APACHECON
North America
Montréal, Canada
24-27 September, 2018

cloud STACK
COLLABORATION CONFERENCE MONTREAL

CloudStack Virtual Router: Past, Present, Future

Rohit Yadav Software Architect, ShapeBlue

rohit.yadav@shapeblue.com

$ whoami

- **Software Architect** @ ShapeBlue.
- From Gurugram, India.
- Background:
  - Committer and PMC, 6 years and counting!
  - RM for 4.11.0.0, RM and maintainer for minor releases 4.5.x, 4.6.x, 4.7.x, 4.8.x, 4.9.x.
  - Specialize in design and architecture, development work, framework, tooling, APIs, KVM, VR/networking, debugging. Author of cloudmonkey 🐵 and several features in CloudStack.
- Vegetarian, love animals 🐱 and programming.

Topics

- Introduction: SystemVMs and VRs
- Comparison: Past and Present VRs
- Overview of Present VR
  - Building and Patching
  - Networking, Isolation, Network types
  - Network Implementation
  - VR programming
- Future Work
- Q&A

What is Virtual Router? Why we need it?

- **What is CloudStack systemvmtemplate?**
  A single VM disk image that can be used to create service VMs: console proxy, ssvm, router, vpc router, dhcp (basic zone) router, elastic LB VM, internal LB VM.
- Virtual Router is a specially patched VM using the systemvmtemplate, it provides SDN solution to CloudStack.
- To support legacy and cloud-era workloads and their network requirements.

What's past, present, future?

- Ancient: ACS 4.0-4.2, 4.3-4.5
- Past: ACS 4.6-4.10
- Present: ACS 4.11
- Future: ACS 4.12/4.13+

Comparison: Virtual Router Factsheet

Past VR (ACS 4.6-4.9)

- Debian Wheezy 7.x 32/64-bit
- Template disk size: 3.2GB
- Linux kernel 3.2.x + init.d
- Python based code
- Java 1.7.x
- VPN: openswan 1:2.6.37x
- VRRP: keepalived 1.2.2x
- Misc: EOL

Present VR (ACS 4.11)

- Debian Stretch 9.x 64-bit
- Template disk size: 1.8-2.1GB
- Linux kernel 4.9.x + systemd
- Python based *refactored code
- Java 1.8.x
- VPN: strongswan 5.5.1x
- VRRP: keepalived 1.3.2x
- Misc: more secure, stable and tested

Migrate to Debian9 systemvmtemplate: https://github.com/apache/cloudstack/pull/2211

Comparison: Virtual Router Building

Past VR (ACS 4.6-4.9)

- VirtualBox + Linux/Mac + Debian minimal-iso installer
- Export tools: qemu-img, vhd-util*, ovftool
- Archive tools: bzip2, zip,
- Targets: KVM, VMware, XenServer, HyperV, OVM3
- Setup: Difficult

Present VR (ACS 4.11)

- Packer + Linux/Qemu + Debian minimal-iso installer
- Export tools: qemu-img, vhd-util*, ovftool
- Archive tools: bzip2, zip,
- Targets: KVM, VMware, XenServer, HyperV, OVM3
- Setup: Easy, CI/CD Jenkins

Comparison: Virtual Router Patching

Past VR (ACS 4.6-4.9)

- Patch using systemvm.iso
- Single large *cloud-early-config* patching script
- Reboot: 1-3 times, post-patching
- Patching: Slow
- Access: SSH + console proxy + limited serial tty access on xenserver

Present VR (ACS 4.11)

- Patch using systemvm.iso
- Several small patching scripts, very small *cloud-early-config*
- Reboot: zero** post-patching
- Patching: fast + faster bootup
- Access: SSH + console proxy + serial tty access on KVM and XenServer

** conditional reboot in case of vmware

Present VR: Towards near zero-downtime upgrade

ENVIRONMENT	ACS 4.9.3 AVG	ACS 4.11.1 AVG (LOWEST)	REDUCTION AVG (HIGHEST)
VMware 5.5	119s	21s (12s)	82% (90%)
KVM / CentOS7	44s	26s (9s)	40% (80%)
XenServer 7.0	181s	33s (15s)	82% (92%)

Network downtime for isolated non-redundant VR

Behaviour for Non-redundant VR:

Deploy a new VR, program it, destroy old VR, re-program it (for arping)

Behaviour for Redundant VR: (~0s downtime)

Destroy old backup VR, provision new backup VR, destroy old master VR, provision new backup VR. VRRP takes care of arp advertisements.

Pull request:
https://github.com/apache/cloudstack/pull/2508

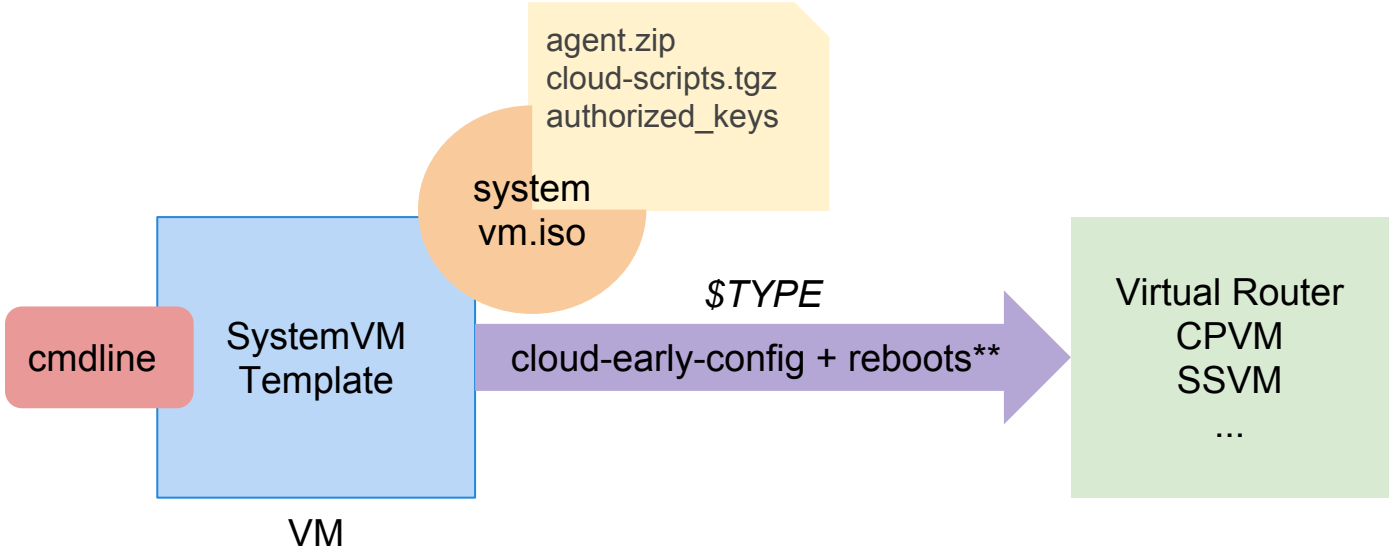
Reference:
https://www.shapeblue.com/working-towards-cloudstack-zero-downtime-upgrades/

Groking the CloudStack VR

Back to the future: Understand the present!

Survey: Current VR Implementation

How SystemVMs are born?



VR Lifecycle

- SystemVM Template

  Build, Patch, Upgrade
- CloudStack + Networking
- VR programming

- VM DHCP + DNS (dnsmasq)
- VM Password (cloud-password-service)
- VM Metadata (apache2)
- Guest Network (iproute2)
- Network ACLs (iptables: filter)
- Firewall Rules (iptables: filter)
- Forwarding Rules (iptables: nat, filter)
- Static NAT Rules (iptables: nat, iproute2)
- Load Balancer (haproxy)
- VPN: S2S, Remote Access, Users (ipsec:strongswan, xl2tpd, ppp)
- Static Routes (iproute2)
- Redundancy (keepalived, conntrackd)
- Service Monitoring (monitor_service.sh)
- Network Stats (netusage.sh)

Demo and Example: MonkeyBox + KVM

Demo and examples of this talk use CentOS 7 + KVM MonkeyBox:

https://github.com/rhtyd/monkeybox

The *mbx* host has a single ethernet bridge that is used for private, public, management and storage networks.

Building SystemVM Template

$ git clone <repo>

$ cd tools/appliance

$ bash build.sh systemvmtemplate

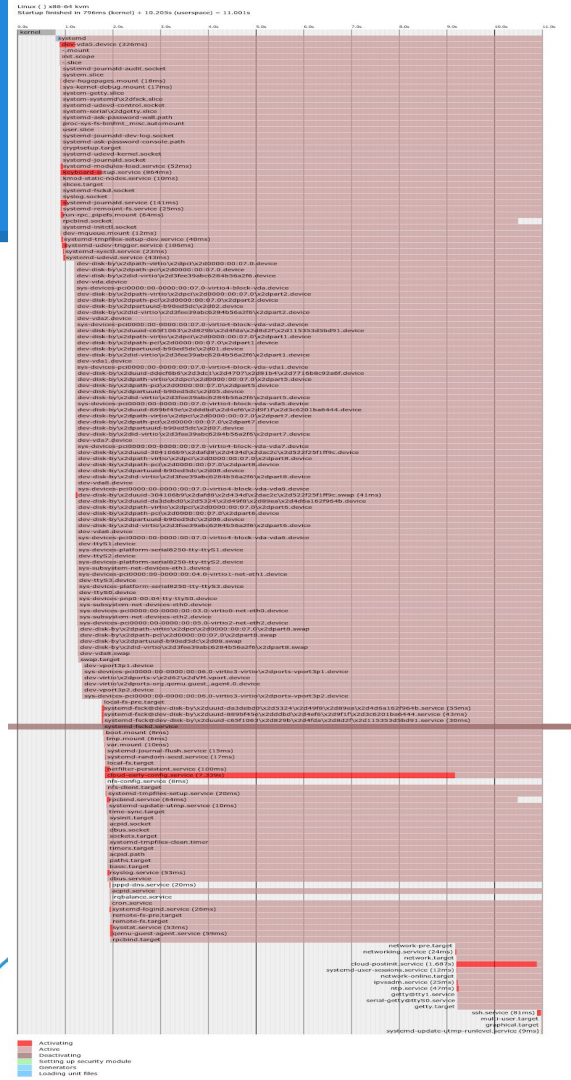
Built artifacts are at: tools/appliance/dist

Reference for packages that are installed:

https://github.com/apache/cloudstack/blob/4.11/tools/appliance/systemvmtemplate/scripts/install_systemvm_packages.sh

ACS 4.11 SystemVM Patching

- Patches without* reboot, systemd enabled and super fast patching!
- **Stage 1**: *cloud-early-config* script patches using attached systemvm.iso. CloudStack passes *cmdline* options via:
  - KVM: serial port (patchviasocket.py)
  - XenServer: kernel cmd params i.e /proc/cmdline or uses xenstore utils
  - VMware: openvm-tools to read
- **Stage 2**: systemvm type based script is executed.
- **Stage 3**: cloud-postinit concludes patching, ejects iso.



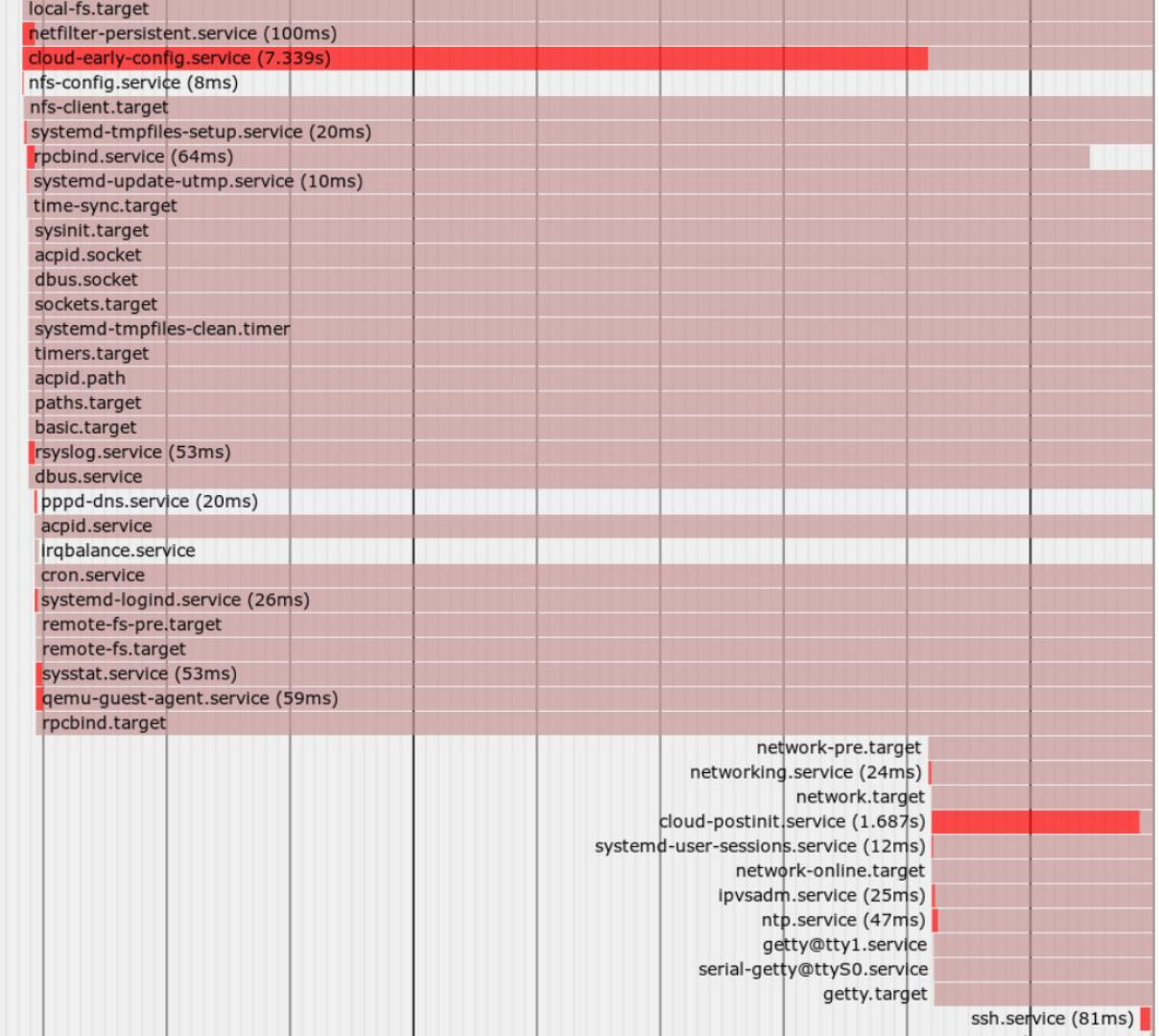
Timeline

Not considered patched until ssh runs!

systemd-analyze

Startup finished in 796ms (kernel) + 10.205s (userspace) = 11.001s

systemd-analyze plot > flame.svg



What is in systemvm.iso?

- **authorized_keys:** ssh public key, this gets patched by management server and kvm agent in the iso file.
- **agent.zip:** CPVM, SSVM jars, assets, configs. Installed at /usr/local/cloud/systemvm/.
- **cloud-scripts.tgz:** /etc configs, /opt/cloud/bin/ VR codebase and scripts, /root and /var config/scripts.
- Attached to a systemvm while booting:
  - KVM: kvm agent attaches it on host using: /usr/share/cloudstack-common/vms/systemvm.iso
  - VMware: attached using systemvm.iso at secondary storage.
  - XenServer: attached from /opt/xensource/packages/iso

sysctl: Kernel Parameters

- Allow packet forwarding:

  net.ipv4.ip_forward = 1
- Disable reverse path verification: (source validation)

  net.ipv4.conf.default.rp_filter = 0 (Only enabled for guest network by CsAddress.py)
- Local ARP interactions: (for local interfaces, guests; don't tell world!)

  net.ipv4.conf.default.arp_announce = 2

  net.ipv4.conf.default.arp_ignore = 2
- Allow binding on IPs that don't exist on interface:

  net.ipv4.ip_nonlocal_bind = 1
- Kernel crash handling: (panic immediately, reboot VR)

  kernel.panic = 10

  kernel.panic_on_oops = 1

  vm.panic_on_oom = 1

References:https://github.com/apache/cloudstack/blob/4.11/systemvm/debian/etc/sysctl.conf

https://www.kernel.org/doc/Documentation/sysctl/kernel.txt

Post-patching

- Access via ssh via port 3922: https://github.com/apache/cloudstack/blob/4.11/systemvm/debian/etc/ssh/sshd_config#L13
- Access IP:
  - KVM: link-local (control)
  - XenServer: link-local (control)
  - VMware: private IP
- ***CheckSshCommand*** is executed post system VM orchestration to check SSH accessibility.

Troubleshooting and Debugging

The old wiki:

https://cwiki.apache.org/confluence/display/CLOUDSTACK/SSVM%2C+templates%2C+Secondary+storage+troubleshooting

- SSH via access IP and port 3922 from host (KVM, XenServer) or management server (VMware).
- Or, access via virsh console <domain> or xe console <vm id>
- Run *systemd-analyze* to check if patching/boot completed.

```
# systemd-analyze
# systemd-analyze critical-chain
# systemctl status --all
```

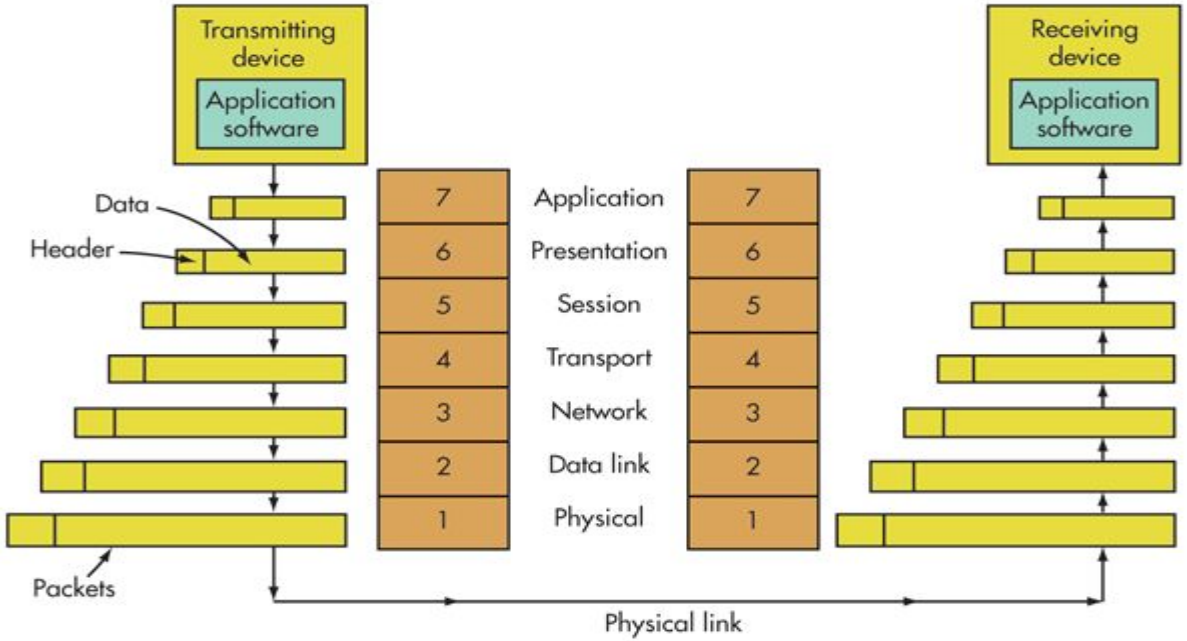
CloudStack Network Models

- Flat Network (switched, shared, L2...)
- NAT-ed Network (isolated, multi-tier/vpc)
- Routed Network: CloudStack does not support ‘em yet (PoC OSPF+quagga exists)

Get your Layers straight!

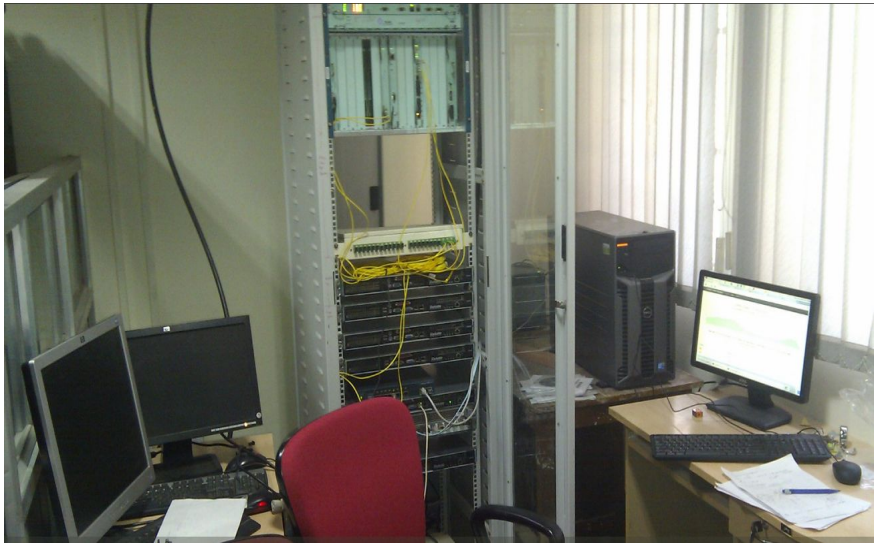
- L2: ARP (MAC address), PPP, FC, FDDI...
- L3: IP (v4+v6), ICMP (v4+v6), IPSec, IGMP, OSPF, RIP…
- L4: TCP, UDP…
- L5: Socket (sessions)
- L6: Presentation (ascii, mime, encoding etc)
- L7: Applications! (http, ntp, dns, dhcp)

OSI Model


Transmitting device
Application software
Receiving device
Application software
Data
Header
Packets
7
6
5
4
3
2
1
Application
Presentation
Session
Transport
Network
Data link
Physical
Physical link

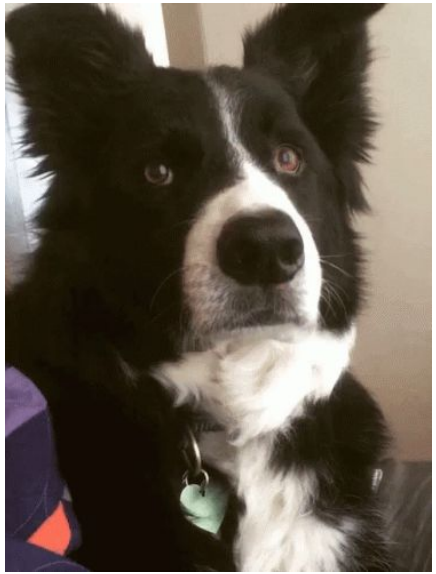
Physical Network

- Switch (L2)
- Trunk + VLANs
- (Core) Routers (L3)
- …



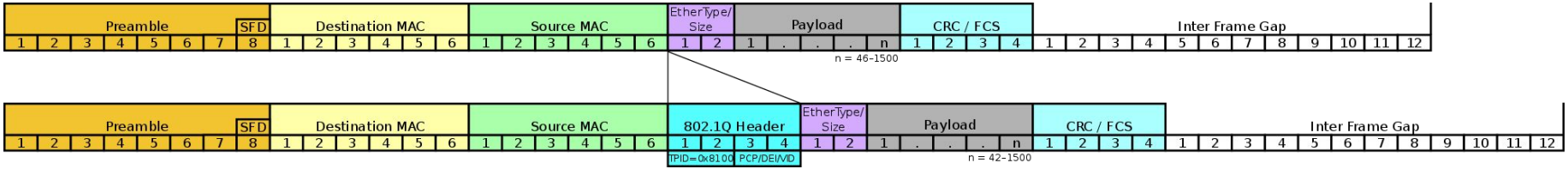
Network Isolation Modes

- **VLAN**
- **VXLAN** (KVM + iproute2)
- GRE (OpenVSwitch)
- STT (Nicira)
- VSP (Nuage)
- L3VPN, ODL ...



** Use-case: Isolate multi-tenant guest network

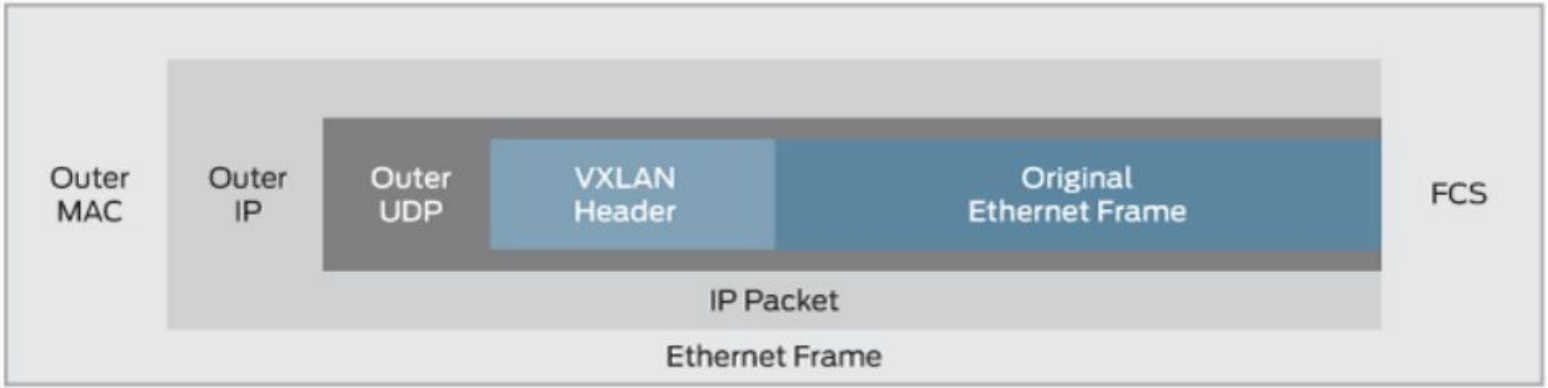
Isolation: VLAN (IEEE 802.1Q)



16 bits	3 bits	1 bit	12 bits
TPID	TCI		
	PCP	DEI	VID

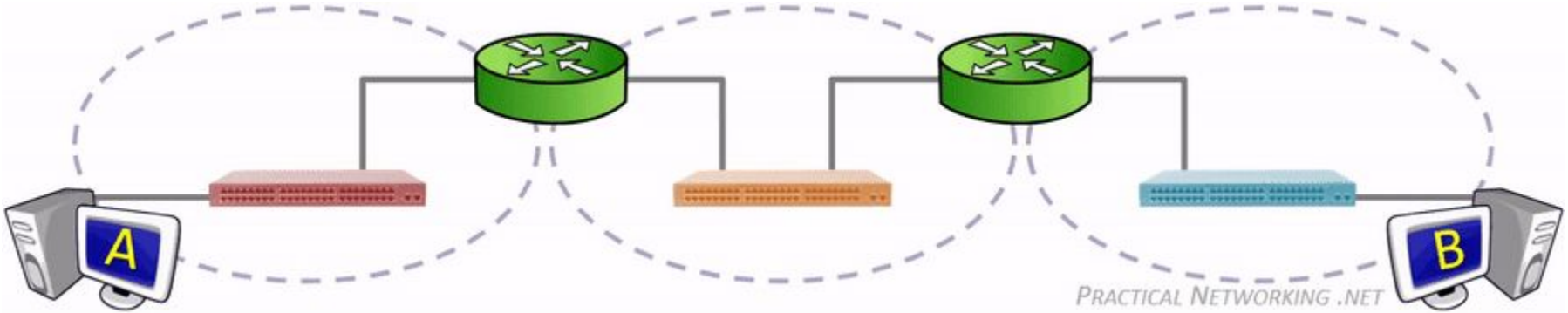
- VLAN ID (VID) - 12 bit. 2^12 ids (4096) ids. (VID 0 & 4095 reserved)
- Load kernel module:
  `modprobe 8021q`
- VLAN config on an interface:
  `vconfig add <intf> <vid>`
  Or, use iproute2:
  `ip link add link <intf> name <intf>.<vid> type vlan id <vid>`
- Add address:
  `ip addr add <ip/mask> dev <intf>.<vid>`
- Enable interface/link:
  `ip link set up <intf>.<vid>`

Isolation: VXLAN



- Encapsulates L2 ethernet *frames* using L4 UDP datagrams (wrapper) which includes the VXLAN header + original ethernet frame.
- Cloud-era isolation: 2^24, or 16M virtual network/isolation.
- KVM only for CloudStack, see http://docs.cloudstack.apache.org/en/latest/networking/vxlan.html

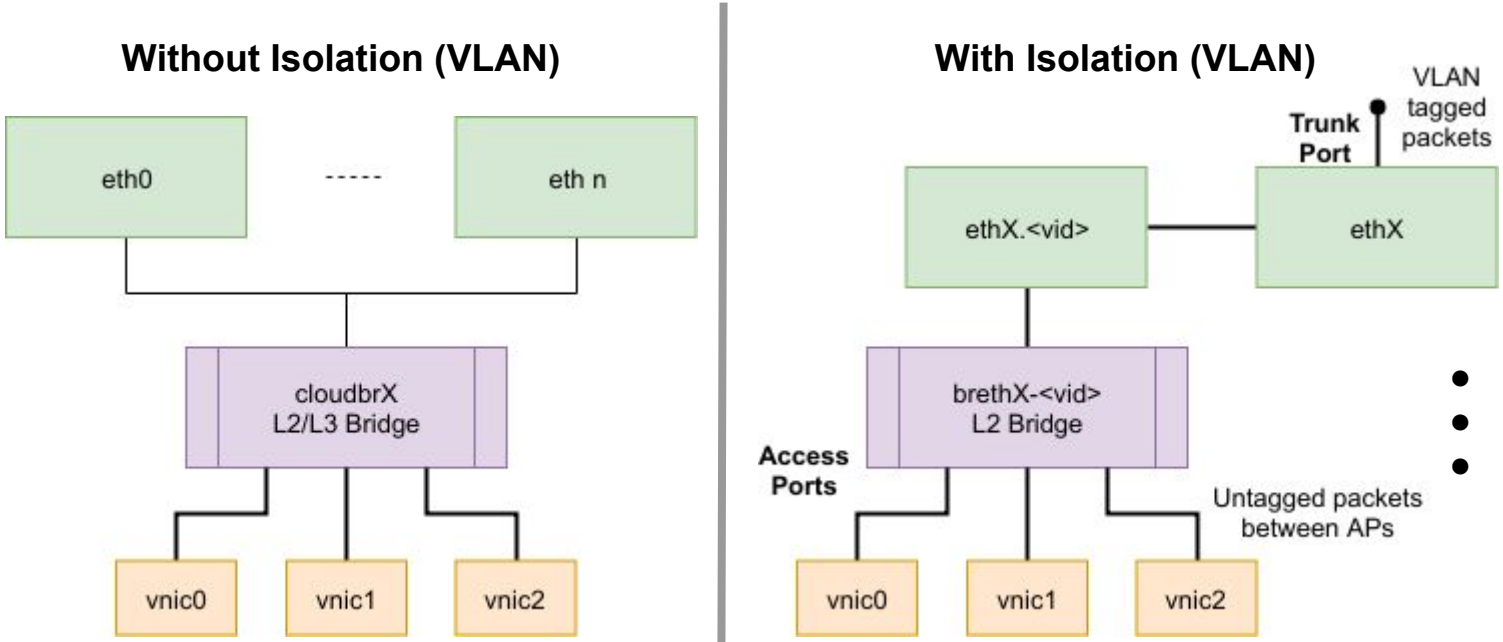
VLAN: What is it again?



- 3 Physical switches, 2 Routers.
- A router routes packets between networks, switches facilitate communication within network.
- A VLAN allows you to take one physical switch, and break it up into smaller mini-switches.

Reference: http://www.practicalnetworking.net/stand-alone/vlans/#tagged-untagged

Typical Bridge Networking (KVM)



- AP ⇉ TP => Tag
- Any ⇉ AP => Untag
- Trunk with native VLAN considered to tag if VID not already present

Reference: Experiment in KVM

- List VMs: `virsh list`
- List interfaces: `ip a[ddr]`
- List bridges: `brctl show`
- Dump xml: `virsh dumpxml <domain>`
- Dump iptables: `iptables-save (or iptables -S)`
- Dump ebtables: `ebtables-save (or ebtables -S)`
- Debug: `tcpdump, traceroute, ping...`

CloudStack Network Types

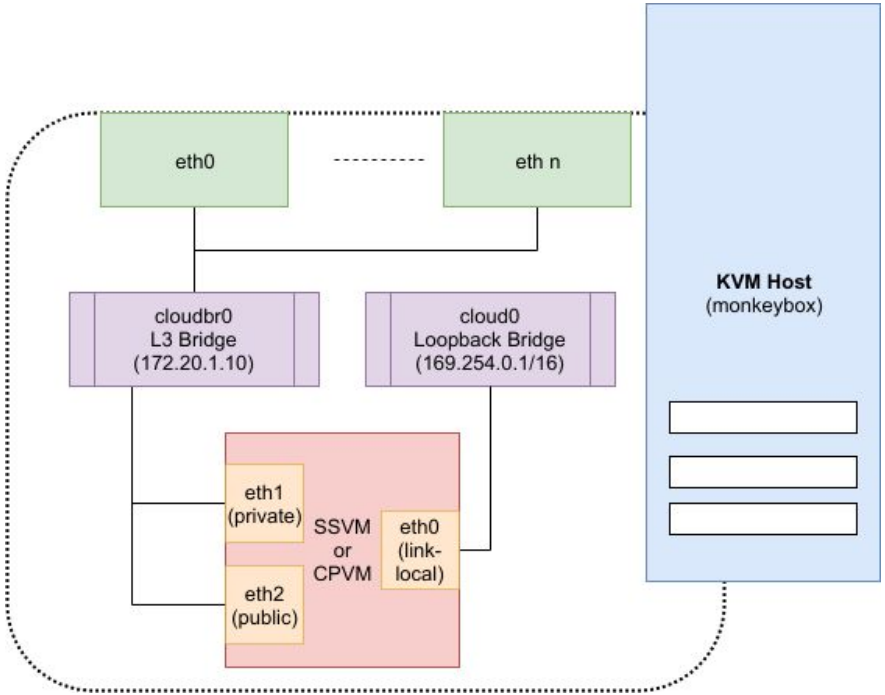
- L2 Network
- Isolated:
  - Isolated Network (Single tier/cidr)
  - VPC Network (Multi-tier/cidrs)
- Shared:
  - Basic Zone + Adv Zone
  - Optional with Security Groups (L2/bridge on host, only supported on XenServer and KVM)

VR Programming

- Orchestration: *VirtualRoutingResource*, *VirtualRouterDeployer*
- Executable scripts at **/opt/cloud/bin/** in VR
- Executable scripts run via **router_proxy.sh** or directly in the /opt/cloud/bin path
- Commands sent as json saved at **/var/cache/cloud/** and updated in VR by **update_config.py**. On updation, they are moved and gzip-ed at /var/cache/cloud/processed.
- VR Config (VR-<uuid>.cfg) file has aggregated file+contents and commands in a custom xml format, processed by vr_cfg.sh.
- VR config jsons are stored at **/etc/cloudstack/** which is used to compare existing vs new config and only diffs (changes) are applied that are calculated by per-command type *databag* handlers (in cs_*.py, merge.py).
- Details!

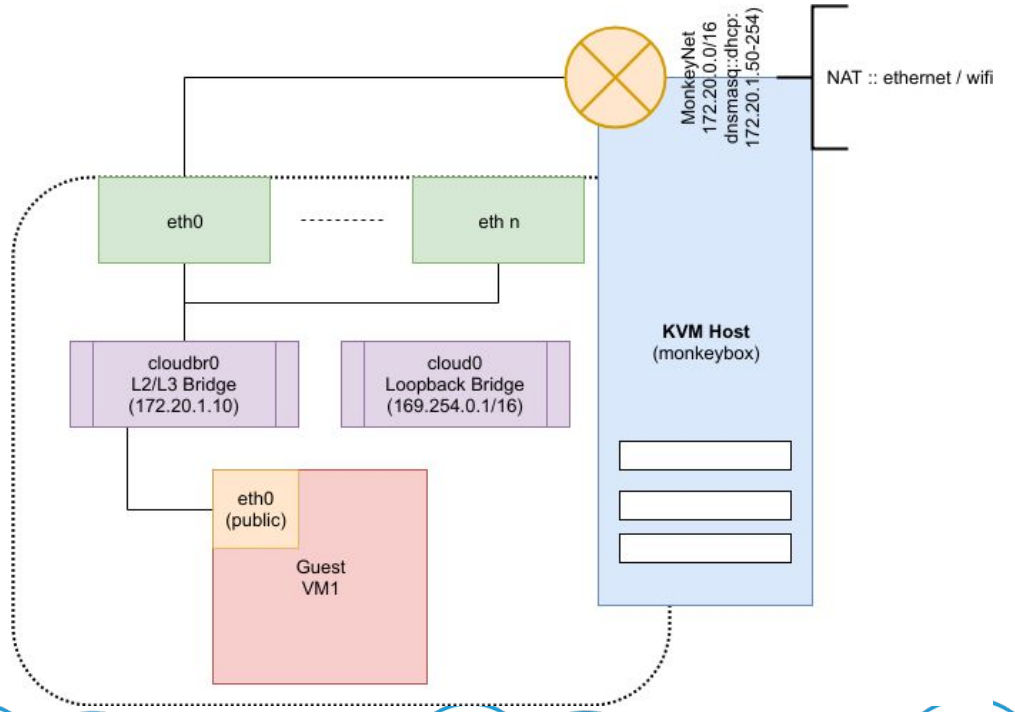
OOM! Kernel Panic! Stop!

SSVM, CPVM Network Setup



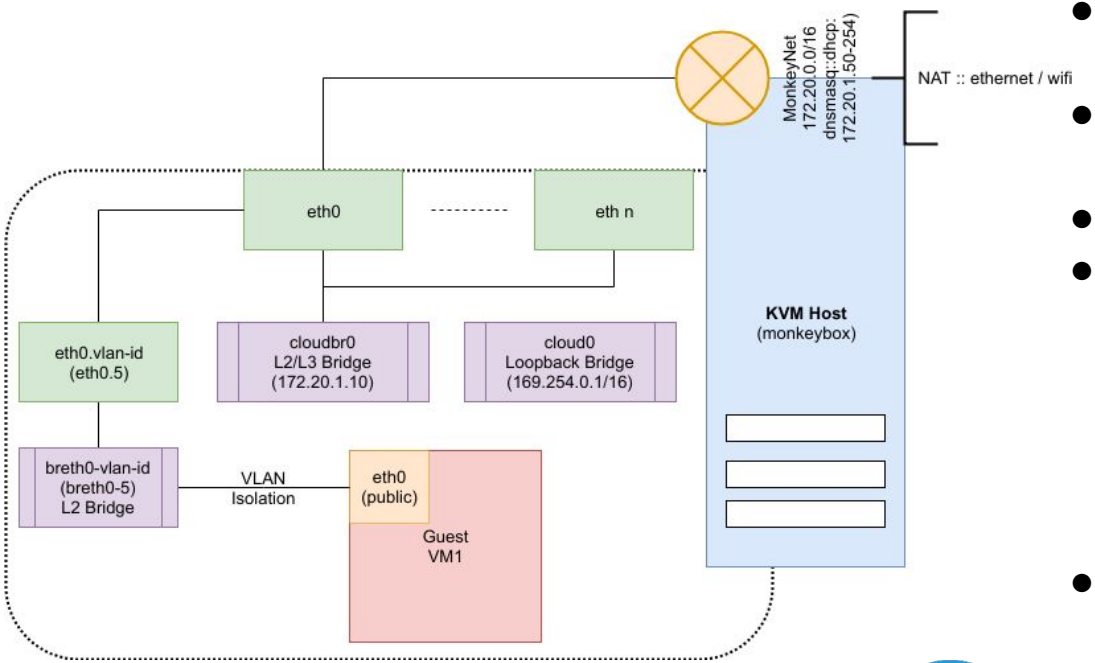
- Agent connects to mgmt server over private network, eth1.
- **eth0** is used as SSH access/control IP to program systemvm for XenServer and KVM.
- **eth1**, private nic/IP is used to access and program systemvm in case of VMware from management server host.

L2 Network with untagged VLAN



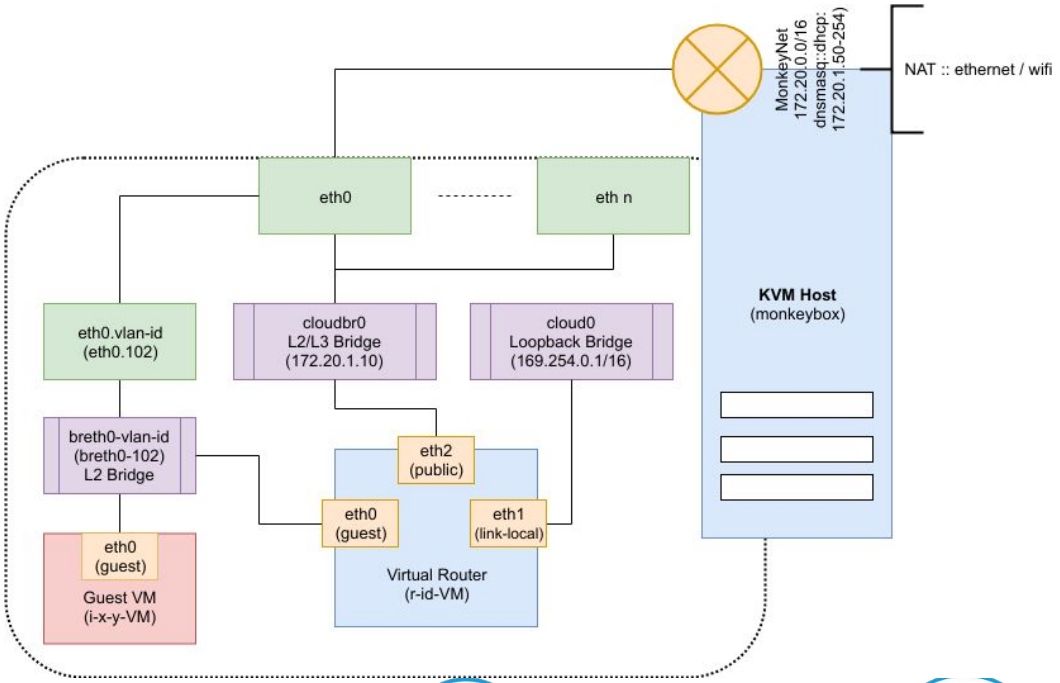
- Gets DHCP/IP from same network as configured host's public L2/L3 bridge.
- No VR needed.
- In the example, guest VM's nic connects to cloudbr0.
- **Use case**: Flat/shared network. Highly scalable, IP/address could be configured manually or config drive based approach.

L2 Network with VLAN



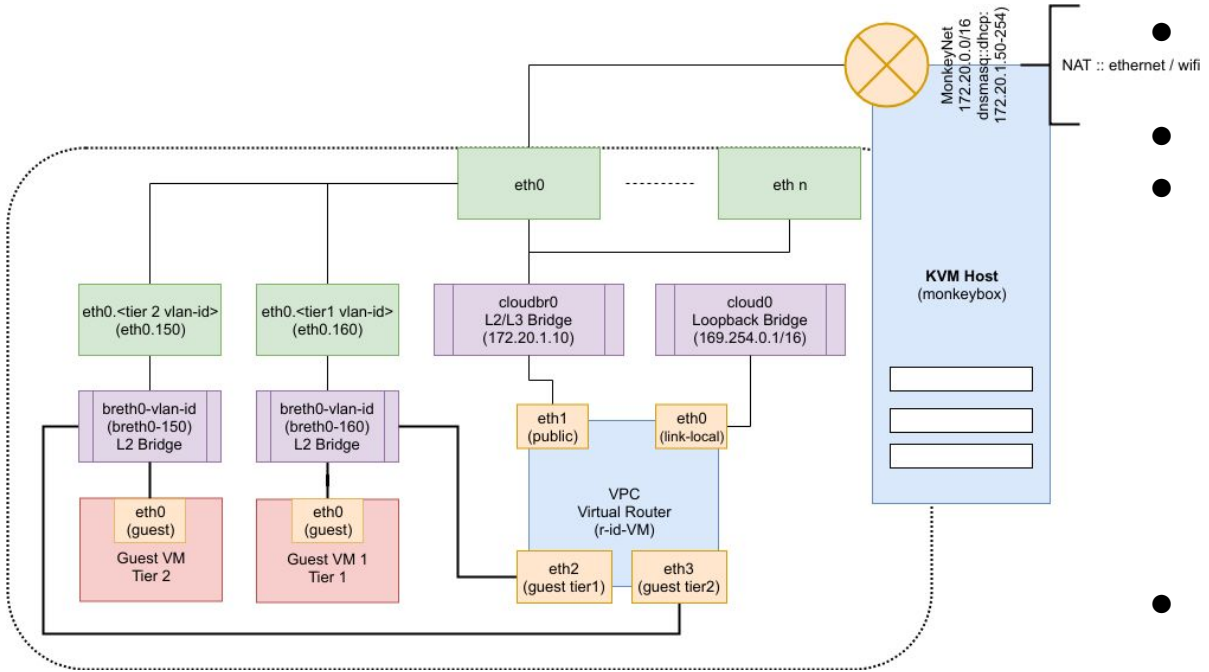
- VLAN based guest network isolation.
- Connects to a L2 bridge for a specific VLAN.
- No VR needed.
- In the example:
  - Guest VM's nic connects to breth0-5.
  - Will not get DHCP response due to VLAN isolation from monkeynet.
- **Use case:** Flat network, no VR but with isolation.

Isolated Network (with VLAN)



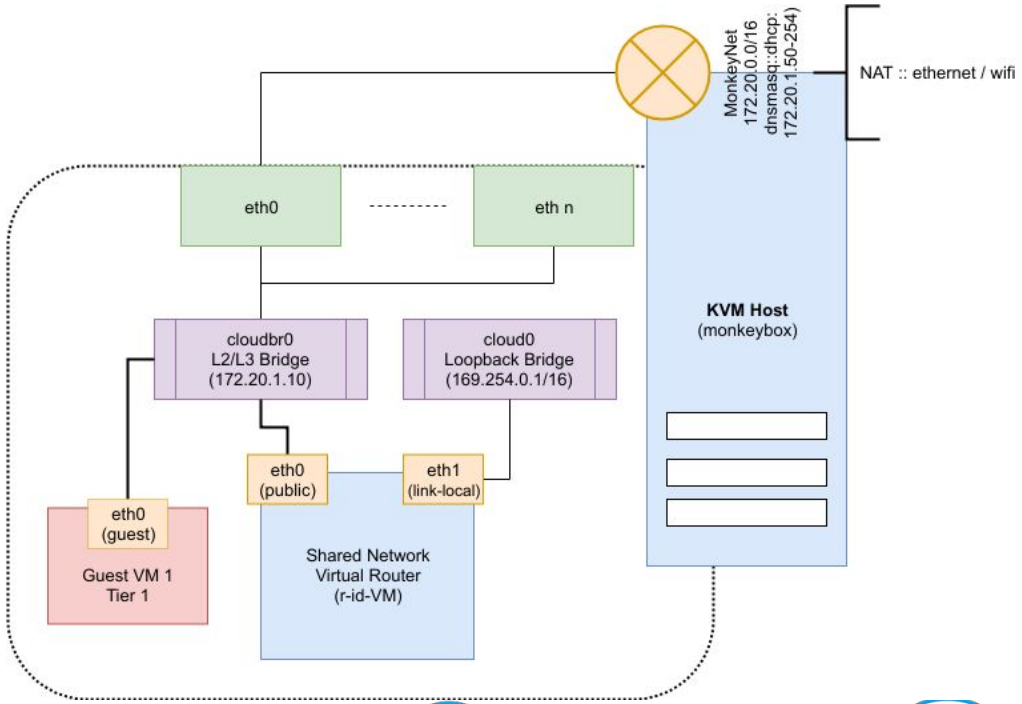
- VLAN based guest network isolation.
- Virtual Router:
  - **eth0** connects to guest network L2-vlan bridge.
  - **eth1** connects to link-local.
  - **eth2** connects to public network bridge.
- **Use-case:** Legacy approach, controlled infra, security boundaries...

VPC (with VLAN)



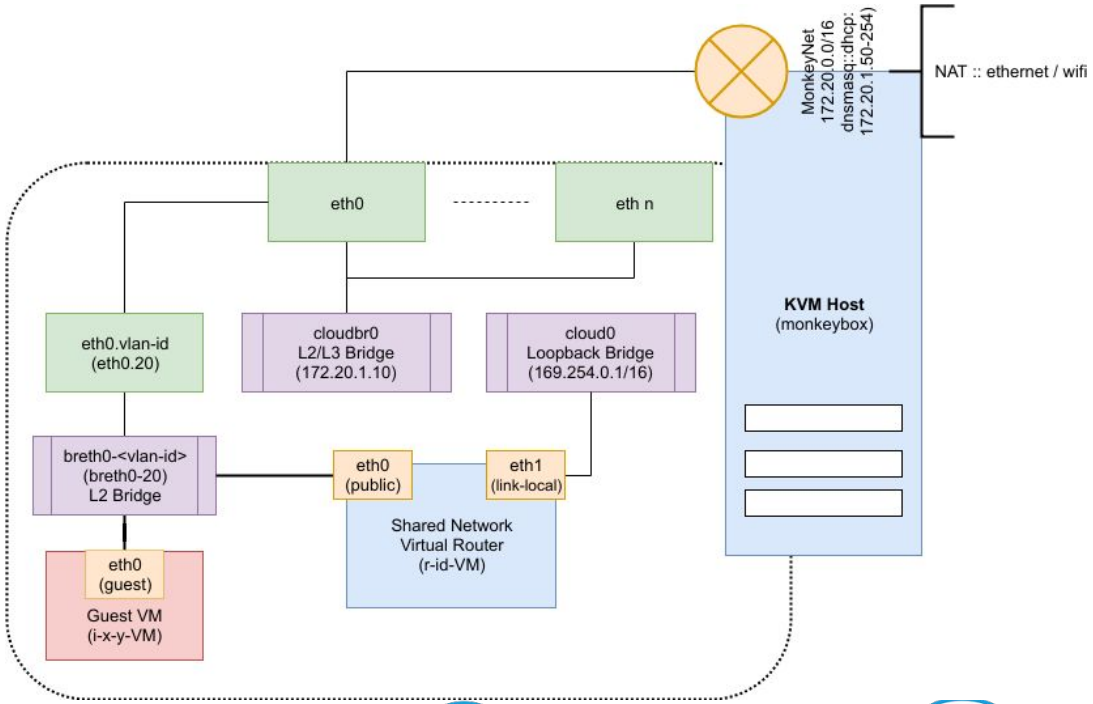
- VLAN based isolation for guest network tiers.
- Lazy programming.
- VPC Virtual Router:
  - **eth0** connects to link-local bridge.
  - **eth1** connects to public network bridge.
  - **eth2** .. **ethn** connects to L2 guest network tier VLAN bridge.
- **Use-case:** same as AWS VPC, n-tier apps, security boundaries...

Shared Network: No Isolation



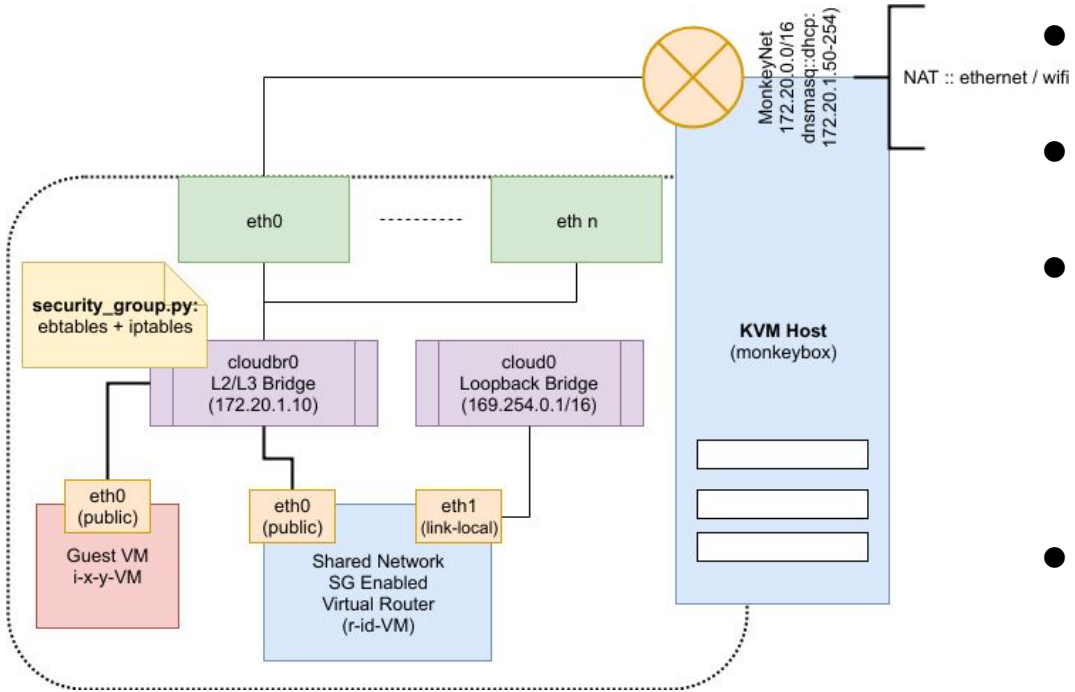
- No isolation.
- VR provides dns and dhcp services using dnsmasq.
- Shared Network VR:
  - **eth0** connects to public network L2 bridge.
  - **eth1** connects to link-local.
- **Use-case:** simplest approach, use dhcp/dns from a physical private/public network.

Shared Network: VLAN



- VLAN guest traffic isolation.
- VR provides dns and dhcp services using dnsmasq.
- Shared Network VR:
  - **eth0** connects to public network VLAN based L2 bridge.
  - **eth1** connects to link-local.
- **Use-case:** Isolation between tenants.

Shared Network: Security Groups



- VLAN guest network isolation available.
- VR provides dhcp+dns service via dnsmasq.
- SG VR:
  - **eth0** connects to public network (VLAN) based L2/L3 bridge.
  - **eth1** connects to link-local.
- **Use-case:** Cloud-era network model, massively scalable, cloud hosting.

Learn from Others

- OpenStack https://twitter.com/rhtyd/status/980923569587834880
- Kubernetes and CNI such as Calico etc.
- Opensource VRs such as Vyos

What have I learnt?

- Build and patching process of a systemvm
- SDN/NFV: Network models, types, topologies and their implementation in CloudStack
- VR is unaware of network isolation
- Non-core services and VR codebase could be abstracted/containerized
- Ideas for future!

Future Work

- Improve SystemVM template upgrade process.
- Redundant capable VRs, state-transfer and handover.
- Improve VR programming (speedup), remove performance bottlenecks.
- Move away from Python 2.x. Migrate to nft + iproute2.
- Optimize SystemVM template size and bootup.
- Testing improvements.
- R&D: Containers? Get rid of VR? Explore other technologies?

Q&A - Thanks!

We're Hiring!

https://www.shapeblue.com/careers/