#RedHatOSD

RED HAT FORUMS

OpenShift and Contrail Integration

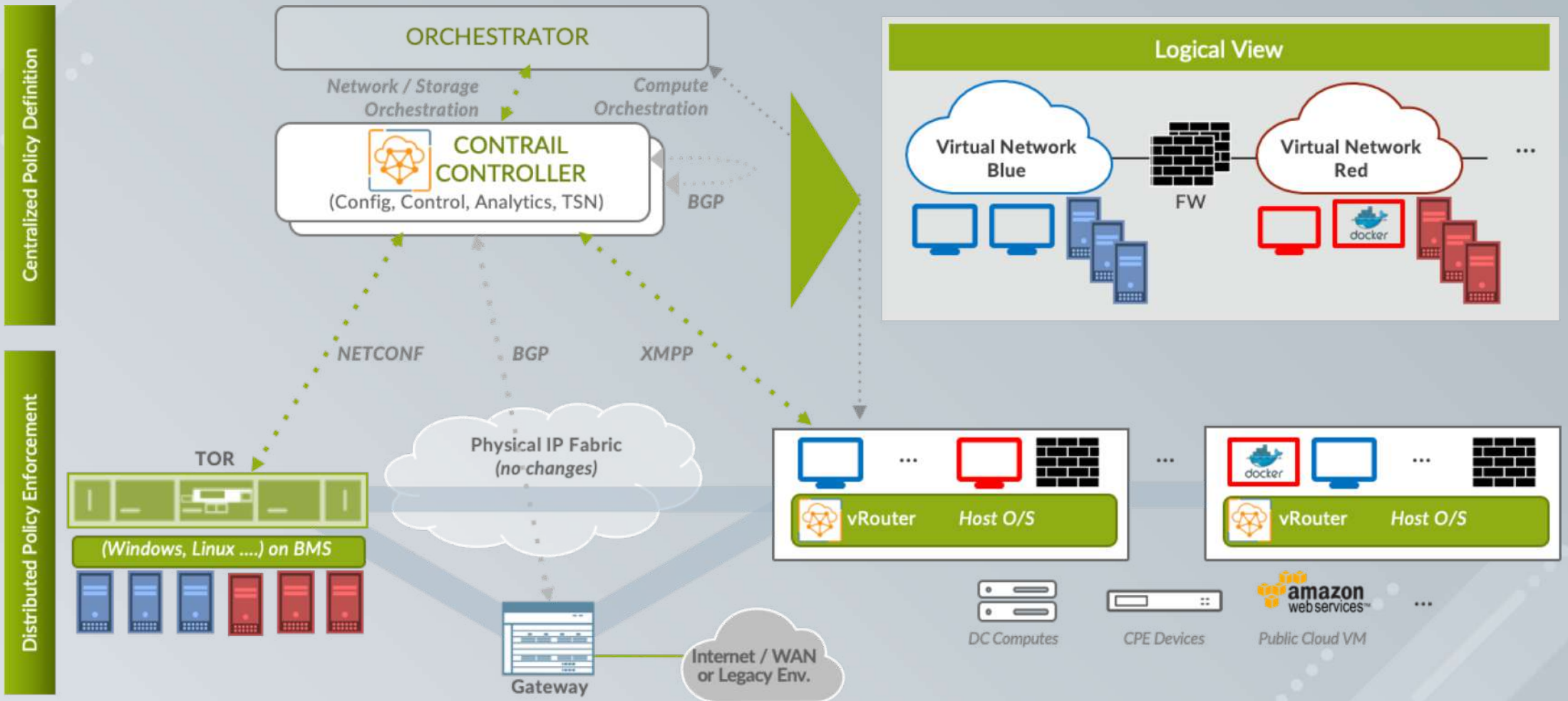
Seamless Networking for Containerized Workload

Luca Tosolini
Professional Service @Juniper Networks
03/12/2019

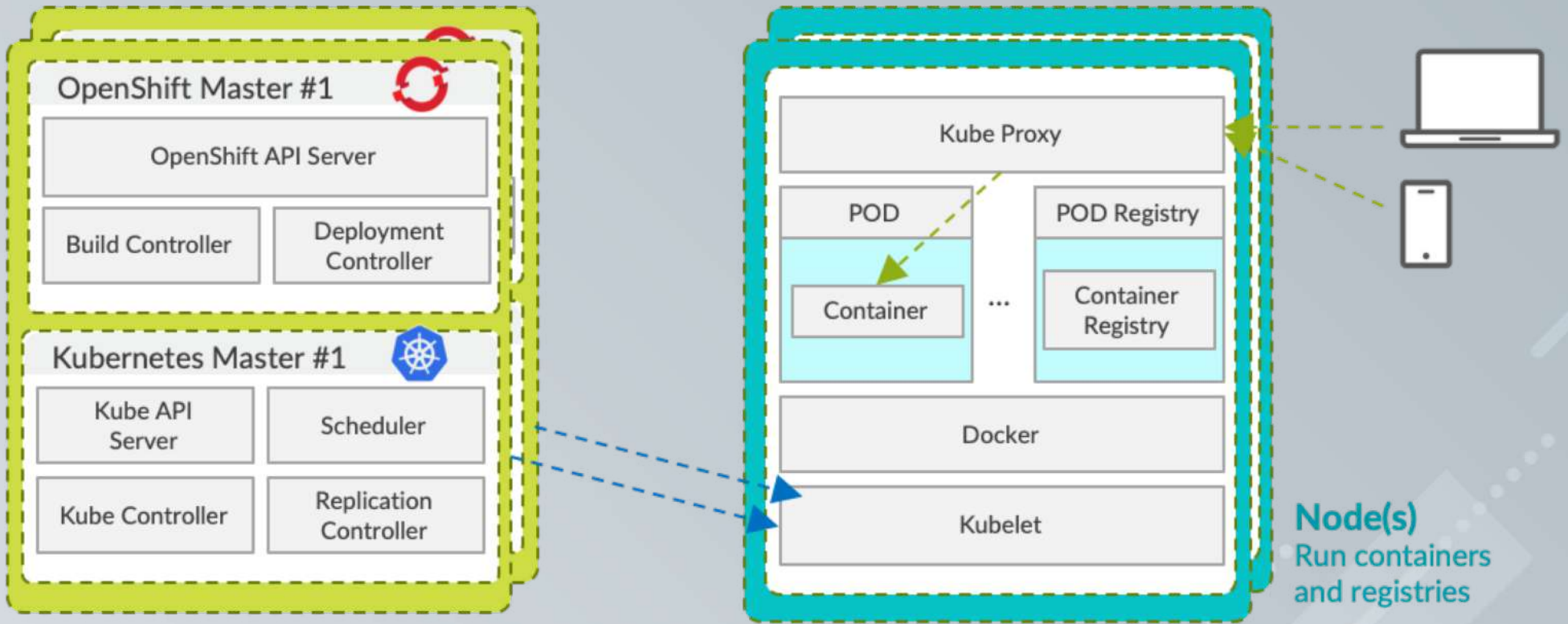
Red Hat | JUNIPER NETWORKS | Engineering Simplicity

The Components

Contrail High Level Architecture



OpenShift High Level Architecture

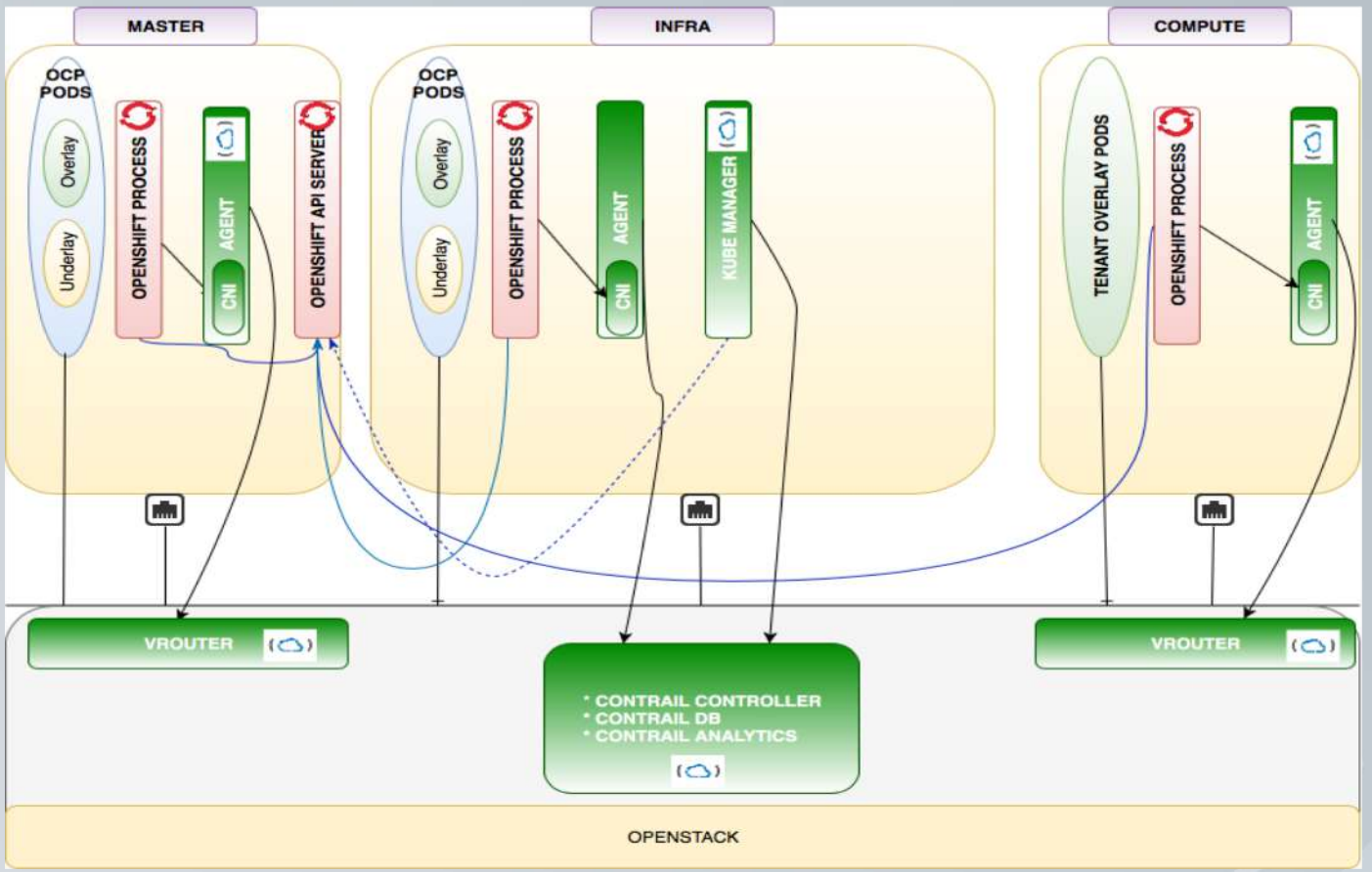


Master(s)
Manage, build, deploy and operate

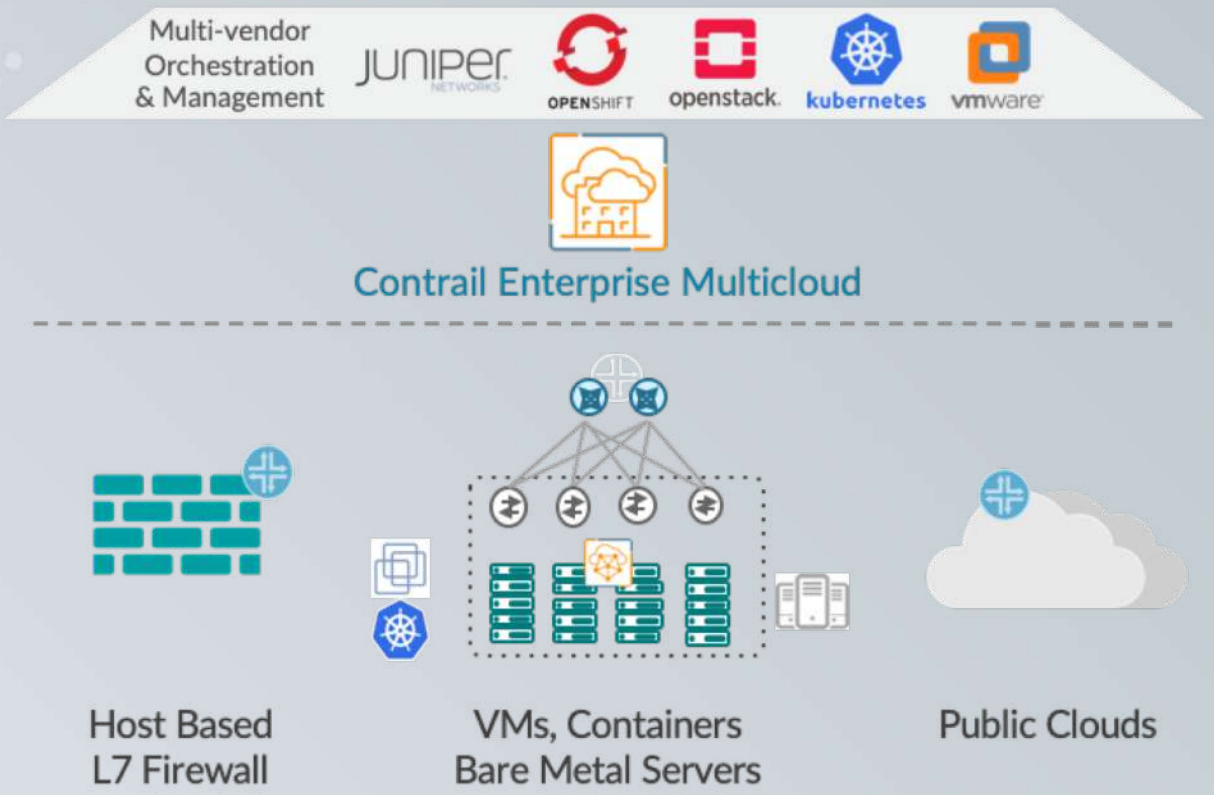
Node(s)
Run containers and registries

Nested Deploy Model

- Openstack + Contrail provide the underlay for OpenShift
- Openshift process never talk directly with Contrail but with the Contrail CNI, CNI then communicates with vRouter and Contrail config.
- When a POD is created, a new vif is created on the compute of type sub-interface whose parent is the OpenShift VM eth0 interface.



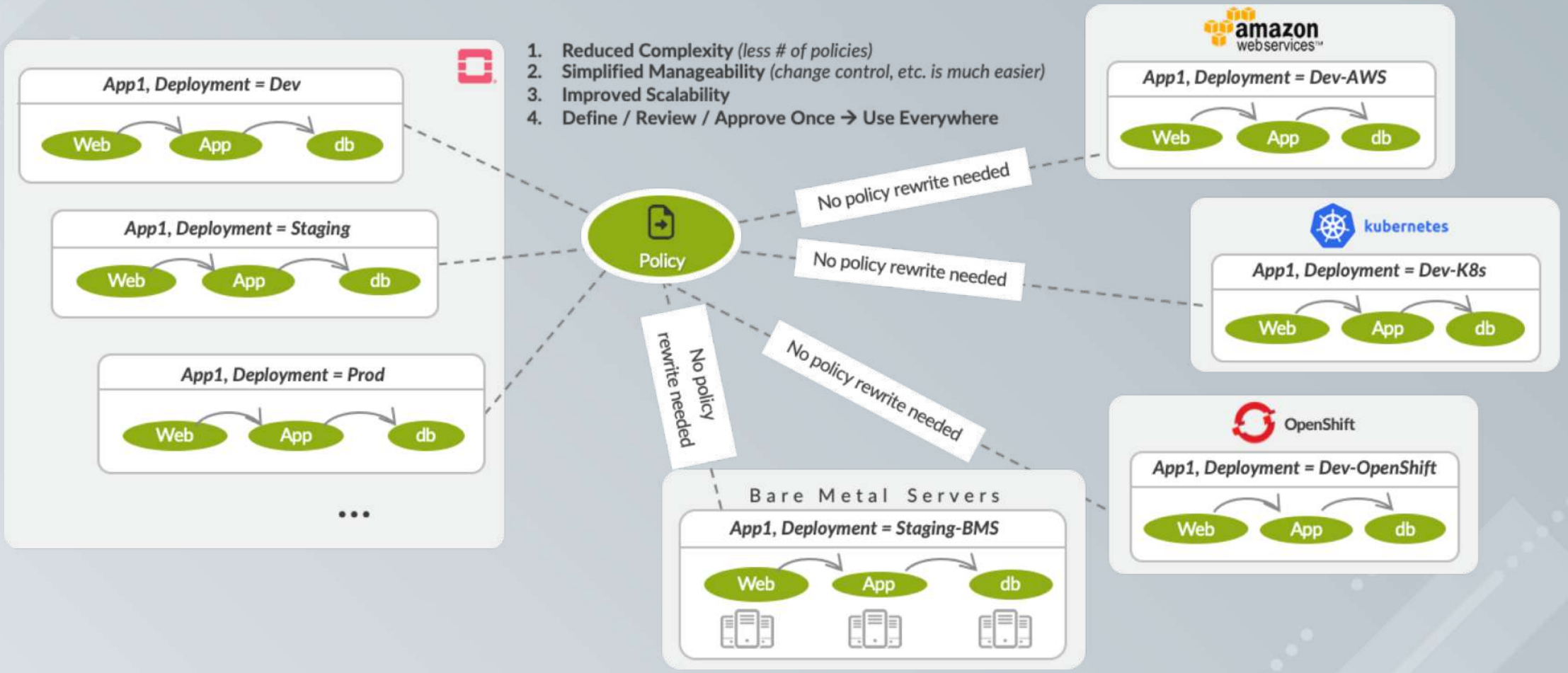
Consistent Security for Multiple Environment



Secure Applications

- Centrally configure and apply fine-grained security policy applied to workloads on every compute
- Enforce security policy with distributed L4 firewalls
- Control the flow of traffic between clouds with centralized policy
- Redirect traffic to a L7 firewall for added protection and visibility

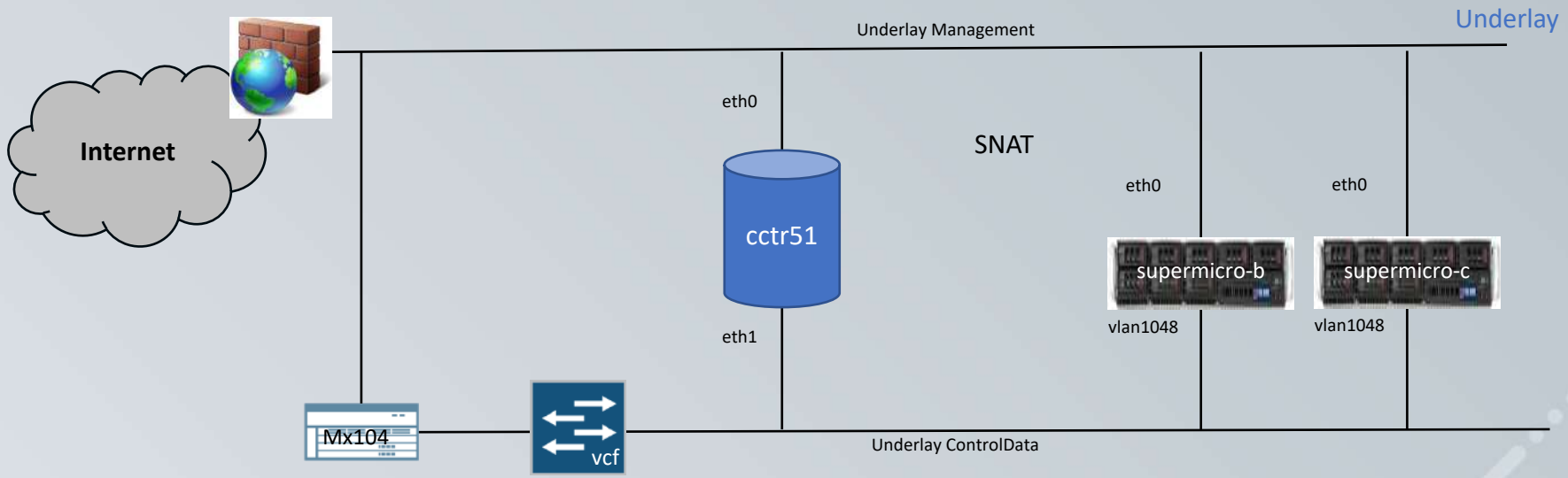
Consistent Network Policy



Deploy

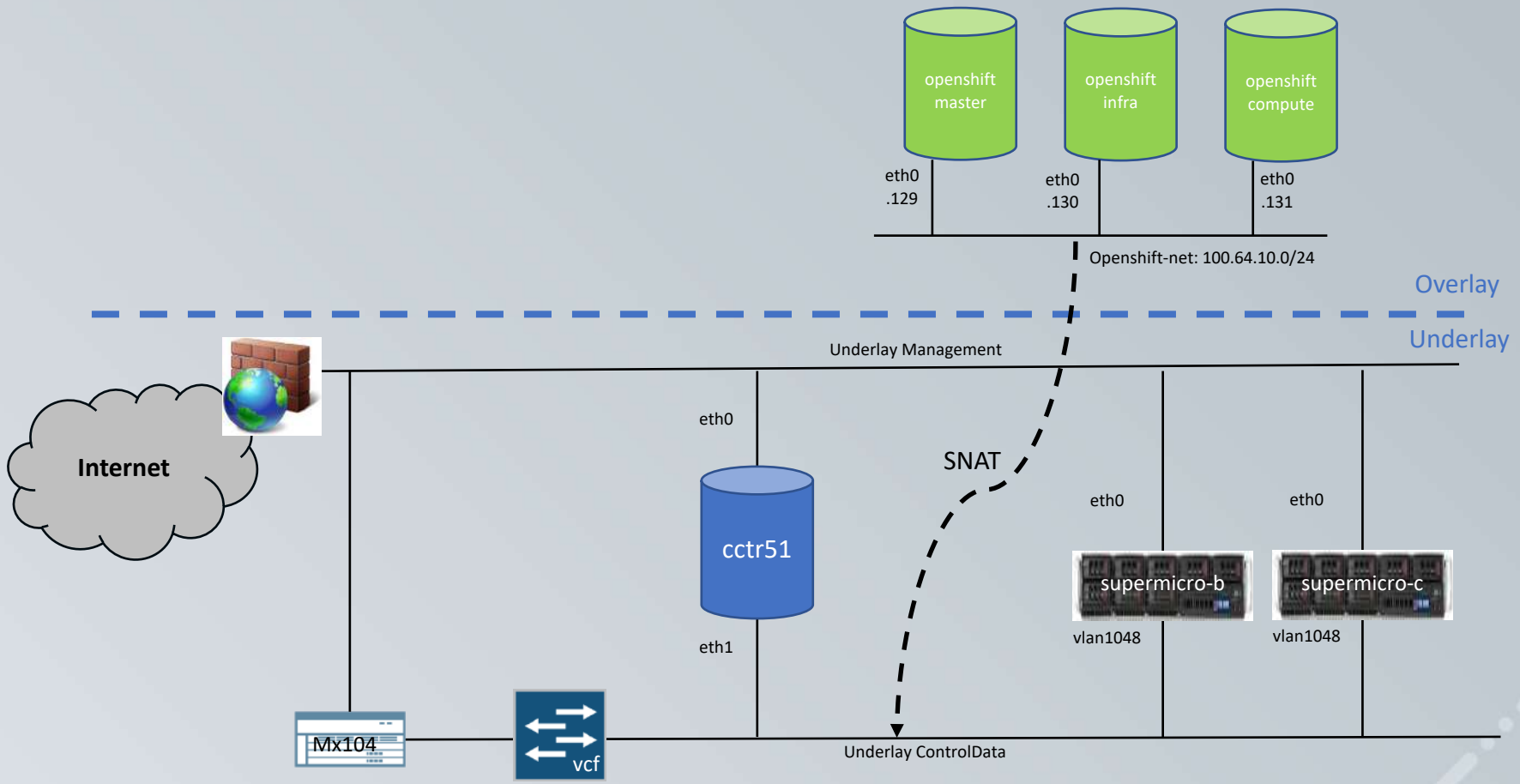
Underlay

Contrail 5.1 + Openstack queen
Fabric + DCGW
Internet access, NAT is ok



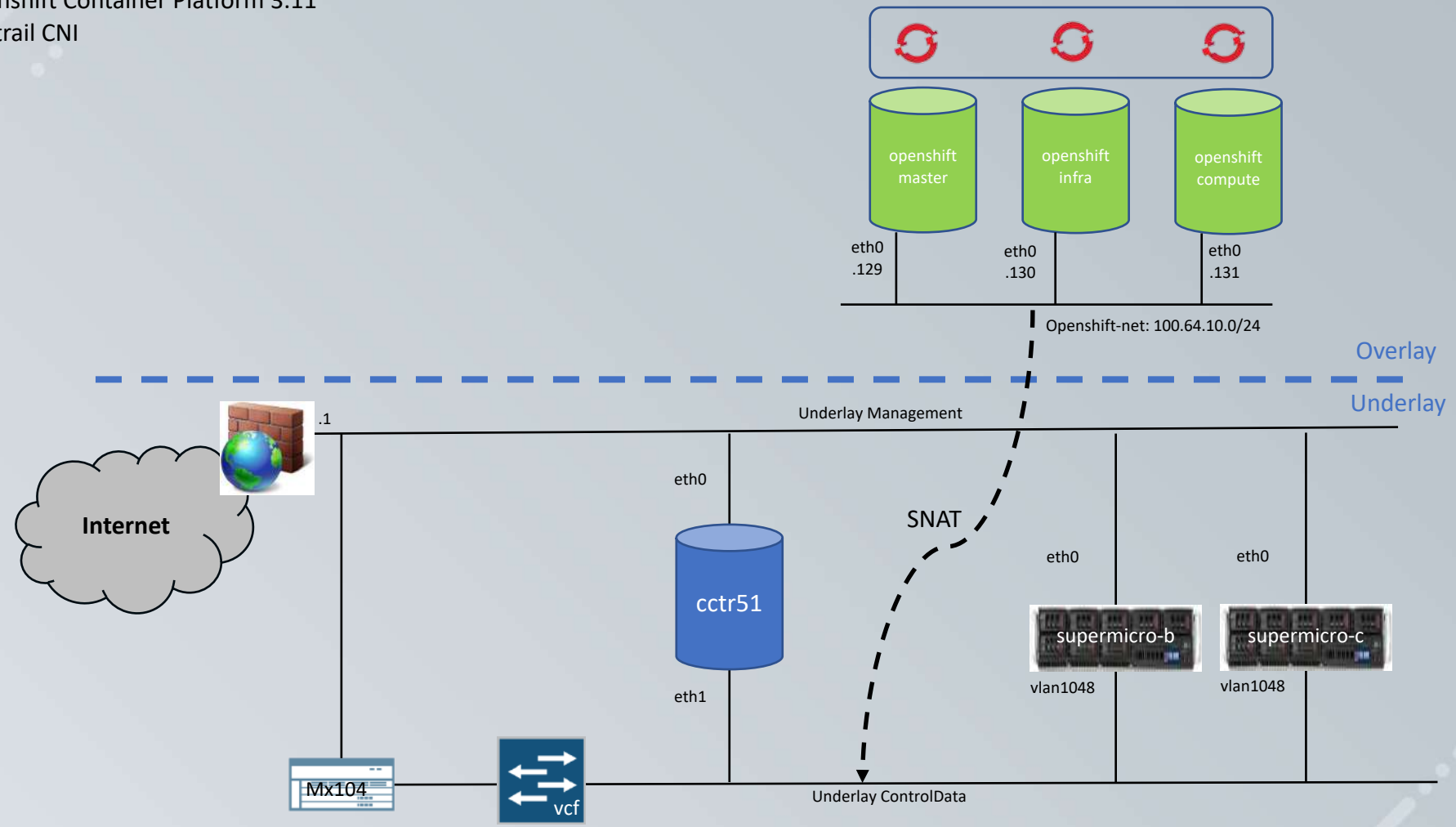
Overlay

3x rhel75 virtual-machine
Cluster-net + SNAT



OpenShift

Openshift Container Platform 3.11
Contrail CNI



Nested Deploy Example

On ansible-deployer:

```
tar zxvf contrail-openshift-deployer-5.1.0-0.38.tgz
cp ose-install openshift-ansible/inventory/
ssh-keygen -q -f ~/.ssh/id_rsa -N ""
ssh-copy-id openshift-master
ssh-copy-id openshift-infra
ssh-copy-id openshift-compute
cd openshift-ansible/
ansible-playbook -i inventory/ose-install playbooks/prerequisites.yml
ansible-playbook -i inventory/ose-install playbooks/deploy_cluster.yml
```

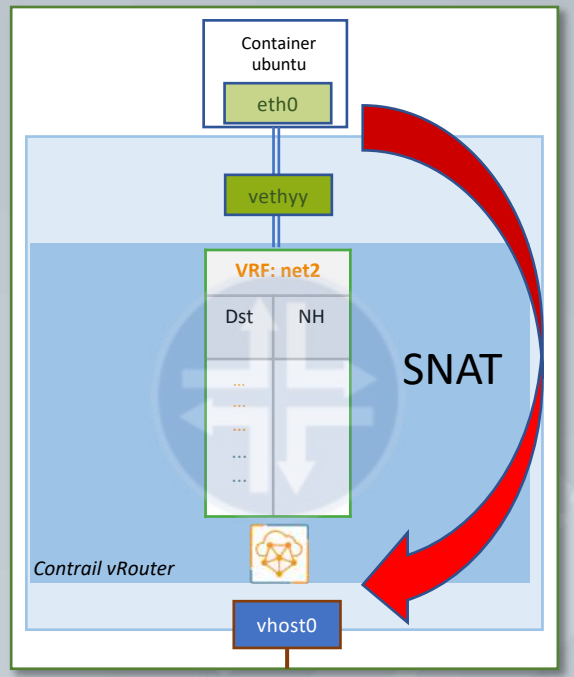
```
PLAY RECAP *********************************************************************************************************************************************************************************************
localhost                  : ok=12   changed=0    unreachable=0    failed=0
openshift-compute          : ok=108  changed=19   unreachable=0    failed=0
openshift-infra            : ok=107  changed=19   unreachable=0    failed=0
openshift-master           : ok=952  changed=250  unreachable=0    failed=0

INSTALLER STATUS ***************************************************************************************************************************************************************************************
Initialization               : Complete (0:00:28)
Health Check                 : Complete (0:00:01)
Node Bootstrap Preparation   : Complete (0:34:19)
etcd Install                 : Complete (0:01:28)
NFS Install                  : Complete (0:00:26)
Master Install               : Complete (0:05:05)
Master Additional Install    : Complete (0:01:24)
Node Join                    : Complete (0:35:38)
Hosted Install               : Complete (0:00:54)
        The use of NFS for the core OpenShift Container Platform components is not recommended, as NFS (and the NFS Protocol) does not provide the proper consistency needed for the applications that make up the OpenShift Container Platform infrastructure.
Cluster Monitoring Operator  : Complete (0:02:47)
Web Console Install          : Complete (0:01:03)
Console Install              : Complete (0:00:39)
Metrics Install              : Complete (0:01:59)
metrics-server Install       : Complete (0:00:44)
Service Catalog Install      : Complete (0:06:53)
Friday 21 June 2019  09:53:35 +0200 (0:00:00.045)       1:34:11.759 **********
===============================================================================
openshift_manage_node : Wait for sync DS to set annotations on all nodes ------------------------------------------------------------------------------------------------------------------------ 1727.49s
openshift_node : Check status of node image pre-pull -------------------------------------------------------------------------------------------------------------------------------------------- 604.28s
openshift_node : Check status of node pod image pre-pull ---------------------------------------------------------------------------------------------------------------------------------------- 604.12s
Waiting for pods to come up --------------------------------------------------------------------------------------------------------------------------------------------------------------------- 360.68s
openshift_node : Create credentials for registry auth ------------------------------------------------------------------------------------------------------------------------------------------- 261.64s
openshift_node : install needed rpm(s) ---------------------------------------------------------------------------------------------------------------------------------------------------------- 200.54s
openshift_service_catalog : Wait for API Server rollout success --------------------------------------------------------------------------------------------------------------------------------- 186.49s
openshift_cluster_monitoring_operator : Wait for the ServiceMonitor CRD to be created ----------------------------------------------------------------------------------------------------- 153.21s
openshift_node : Install node, clients, and conntrack packages ---------------------------------------------------------------------------------------------------------------------------------- 94.89s
template_service_broker : Verify that TSB is running -------------------------------------------------------------------------------------------------------------------------------------------- 65.42s
openshift_service_catalog : Wait for Controller Manager rollout success ------------------------------------------------------------------------------------------------------------------------- 65.24s
openshift_web_console : Verify that the console is running -------------------------------------------------------------------------------------------------------------------------------------- 53.47s
cockpit : Install cockpit-ws -------------------------------------------------------------------------------------------------------------------------------------------------------------------- 49.23s
openshift_excluder : Install docker excluder - yum ---------------------------------------------------------------------------------------------------------------------------------------------- 46.92s
openshift_cli : Install clients ----------------------------------------------------------------------------------------------------------------------------------------------------------------- 37.69s
openshift_node : Install iSCSI storage plugin dependencies -------------------------------------------------------------------------------------------------------------------------------------- 32.48s
openshift_excluder : Install openshift excluder - yum ------------------------------------------------------------------------------------------------------------------------------------------- 31.89s
openshift_ca : Install the base package for admin tooling --------------------------------------------------------------------------------------------------------------------------------------- 30.03s
openshift_console : Waiting for console rollout to complete ------------------------------------------------------------------------------------------------------------------------------------- 29.62s
nickhammond.logrotate : nickhammond.logrotate | Install logrotate ------------------------------------------------------------------------------------------------------------------------------- 24.61s
```

Use Cases

Outbound Connectivity

During cluster deploy and later during operations, PODs require Internet access; this is achieved by means distributed SNAT on the k8s-default-pod-network. With such configuration the vRouter translates source address of containers into its own controldata address which should be a public routable address; if it is not, like in our lab, it should be further natted to a public address.
In case of isolated namespace, a new virtual-network is created per namespace; if Internet access is required, distributed SNAT should be enabled on it.
Furthermore, SNAT must be enabled on the openshift-net to allow kube-manager and cni to communicate with contrail services.

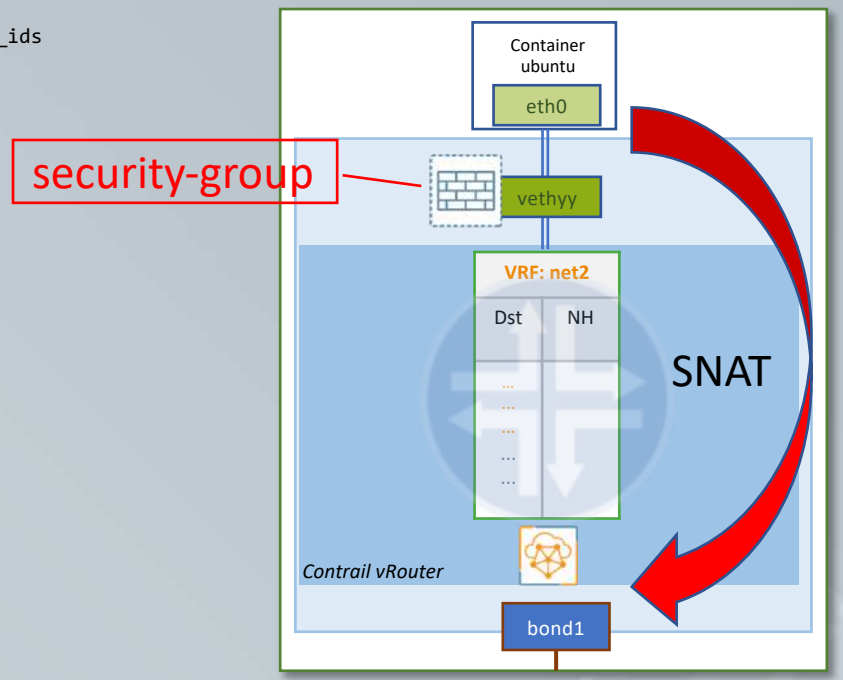
```
[root@openshift-master ~]# oc create -f ubuntu-pod.yml
pod/ubuntu created
[root@openshift-master ~]# oc exec -it ubuntu bash
root@ubuntu:/# ping -c 2 fritz.inet6.biz
PING fritz.inet6.biz (87.0.151.163) 56(84) bytes of data.
64 bytes from host163-151-dynamic.0-87-r.retail.telecomitalia.it (87.0.151.163): icmp_seq=1 ttl=50 time=12.2 ms
64 bytes from host163-151-dynamic.0-87-r.retail.telecomitalia.it (87.0.151.163): icmp_seq=2 ttl=50 time=11.3 ms

[root@supermicro-b ~]# ip -4 -o address
1: lo    inet 127.0.0.1/8 scope host lo\       valid_lft forever preferred_lft forever
2: eth0    inet 172.30.124.34/24 brd 172.30.124.255 scope global eth0\       valid_lft forever preferred_lft forever
12: vhost0    inet 163.162.230.137/26 brd 163.162.230.191 scope global vhost0\       valid_lft forever preferred_lft forever
13: docker0    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0\       valid_lft forever preferred_lft forever

[root@supermicro-b ~]# ip address show dev vlan1048
8: vlan1048@bond1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc noqueue state UP group default qlen 1000
    link/ether 0c:c4:7a:58:a7:ea brd ff:ff:ff:ff:ff:ff

[root@supermicro-b ~]# tcpdump -ni vlan1048 icmp or \( udp port 53 \)
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on vlan1048, link-type EN10MB (Ethernet), capture size 262144 bytes
11:10:42.081899 IP 163.162.230.137 > 87.0.151.163: ICMP echo request, id 27, seq 1, length 64
11:10:42.093465 IP 87.0.151.163 > 163.162.230.137: ICMP echo reply, id 27, seq 1, length 64
```



per POD / Interface Security

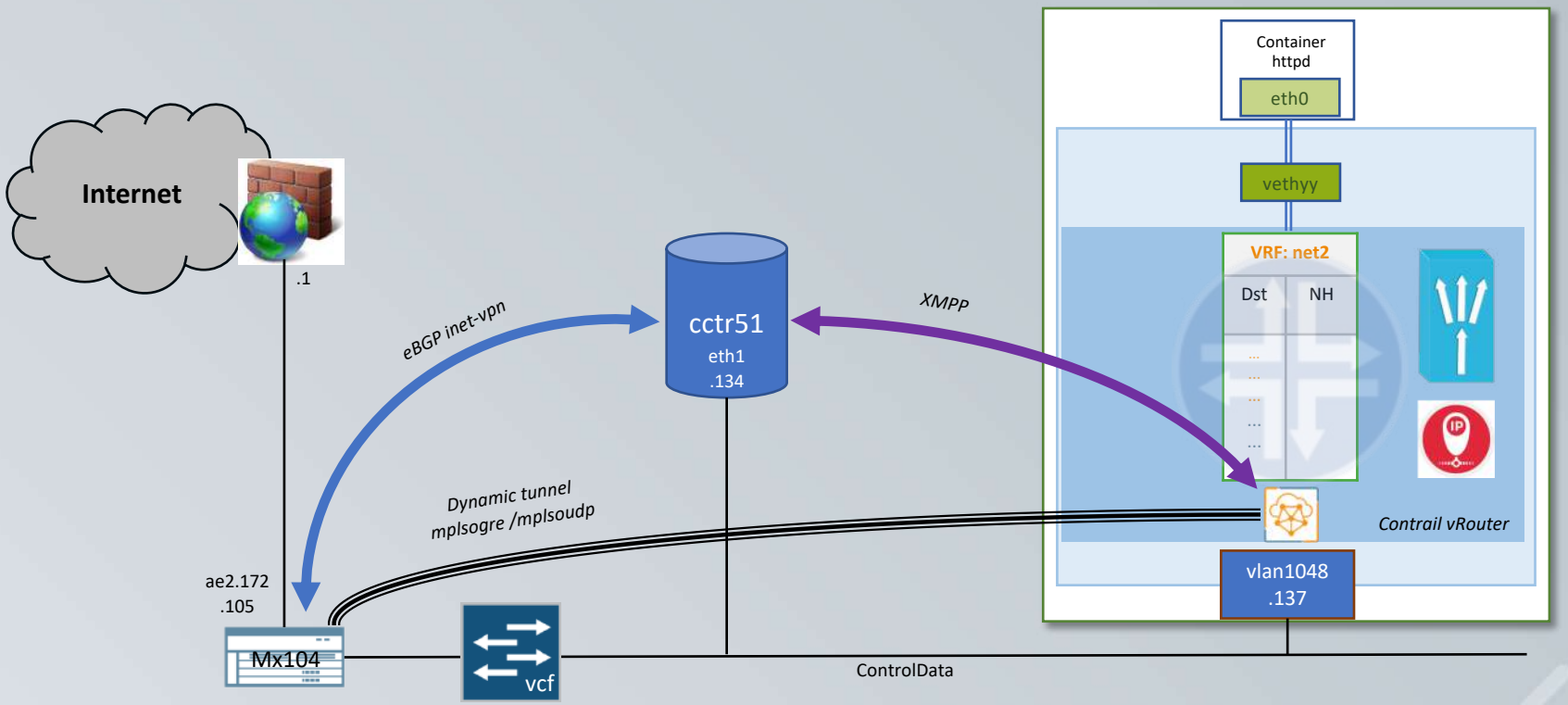
The fact that every POD owns its own interface to the vRouter, allows for fine grained security and is propedeutical to microsegmentation: every port can get a specific security-group.

```
ltosolini@r410:~/OpenShift/OCP/Demo$ openstack port show 7ec2a7d8-9824-11e9-816d-02011c8652da -cname -csecurity_group_ids
+--------------------+---------------------------------------------+
| Field              | Value                                       |
+--------------------+---------------------------------------------+
| name               | ubuntu__7ec2a7d8-9824-11e9-816d-02011c8652da |
| security_group_ids | 4ec375af-255c-492b-94d4-434fdc95ed80        |
+--------------------+---------------------------------------------+
ltosolini@r410:~/OpenShift/OCP/Demo$ openstack security group rule list 4ec375af-255c-492b-94d4-434fdc95ed80
+--------------------------------------+-------------+------------+------------+-----------------------+
| ID                                   | IP Protocol | IP Range   | Port Range | Remote Security Group |
+--------------------------------------+-------------+------------+------------+-----------------------+
| 9fbe4da1-096c-495a-8ef3-cb73da3c8d5b | any         | ::/0       | 0:65535    | None                  |
| be84e255-b867-41ca-bc22-cd08d5e90594 | any         | 0.0.0.0/0  | 0:65535    | None                  |
| 36edba56-3da5-4ffc-9059-d60574dc6c6f | any         | 8.8.8.8/32 | 0:65535    | None                  |
+--------------------------------------+-------------+------------+------------+-----------------------+

[root@openshift-master ~]# oc exec -it ubuntu bash
root@ubuntu:/# ping -c 2 8.8.4.4
PING 8.8.4.4 (8.8.4.4) 56(84) bytes of data.

--- 8.8.4.4 ping statistics ---
2 packets transmitted, 0 received, 100% packet loss, time 1000ms

root@ubuntu:/# ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=47 time=17.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=47 time=16.9 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 16.922/17.198/17.475/0.306 ms
```



External Access to POD

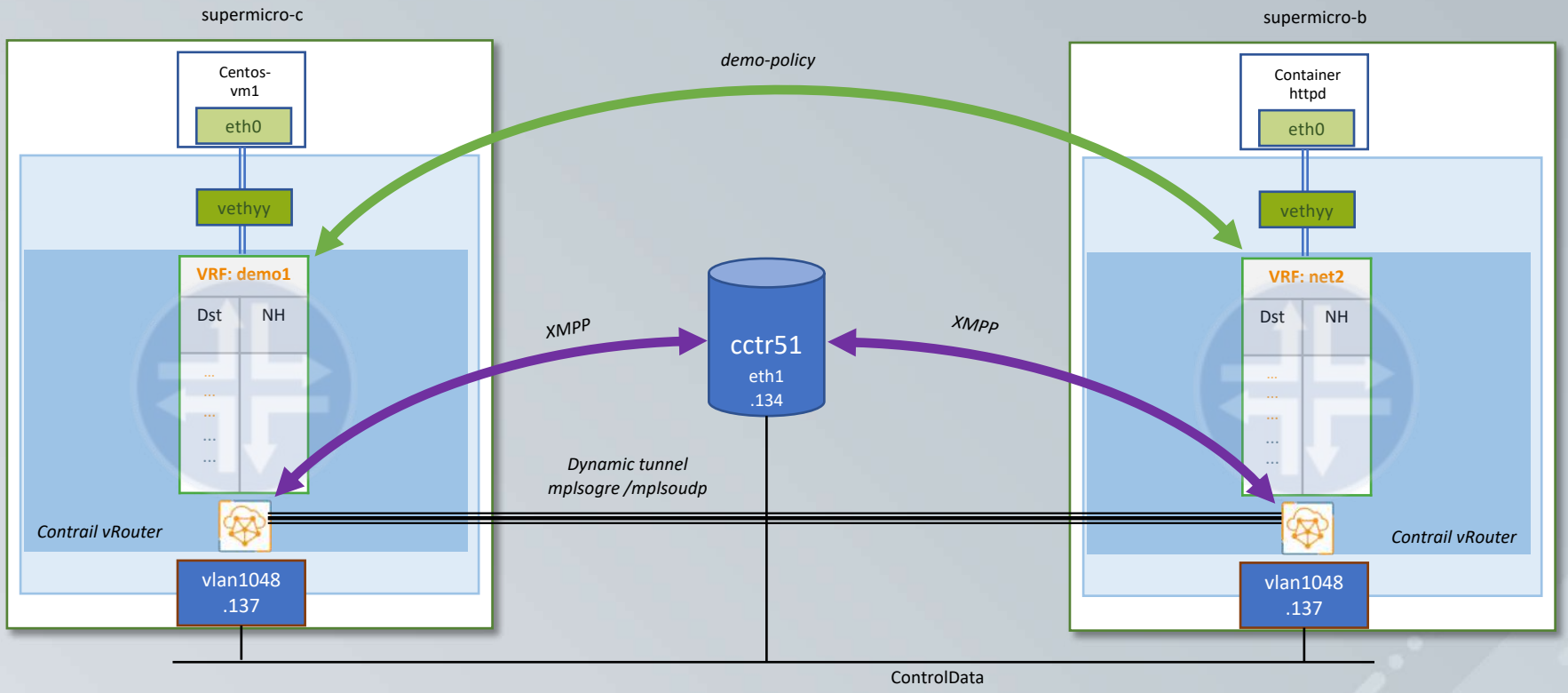
When OpenShift is integrated with Contrail, all the advanced networking capabilities of Contrail become available to OpenShift PODs and services.
Contrail leverages mature and proven technology like l3-mpls-vpn (rfc4364) to extend connectivity between virtual-networks and traditional networks, such as operator IP/MPLS backbone, by mean of a gateway router DCGW which in this setup is represented by Mx104. When a matching route-target community is configured on a virtual-network and on a DCGW vrf, contrail-controller exchanges routes between the two. On the dataplane Mx and vRouter establish dynamic tunnels, MPLSoGRE or MPLSoUDP, to carry overlay traffic.



Interconnect VM - POD

By leveraging advanced SDN capabilities of Contrail, it is possible to seamlessly interconnect OpenStack virtual-machines and OpenShift containers. In this example:

- centos-vm1 on virtual-network demo1_net running on compute supermicro-c is put in communication with
- POD httpd on virtual-network k8s-default-pod-network running on compute supermicro-b
- A contrail routing-policy demo-policy, instructs contrail controller to exchange routes between the two virtual-networks propagating this information to vRouters
- Based on next-hop information, vRouters establish a dynamic tunnel between themselves to carry overlay traffic.



#RedHatOSD

RED HAT FORUMS

THANK YOU

linkedin.com/company/Red-Hat

facebook.com/RedHatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat | JUNIPER NETWORKS | Engineering Simplicity