Tungsten Fabric Microservice Architecture & Role in Edge Computing

Qasim Arham

Nov 6, 2018 (OSN Day Dallas)

Tungsten Fabric Linux Foundation Project

https://tungsten.io/

COMMUNITY MEMBERS

Networking for Edge Computing

Networking is most overlooked and underestimated component in any stack

Networking is focal point for most of the security and scalability issues

Tungsten Fabric is fully distributed and Microservices based SDN controller addressing security, scale and advance networking services

Production grade networking stack for Data Center and Public & Edge cloud

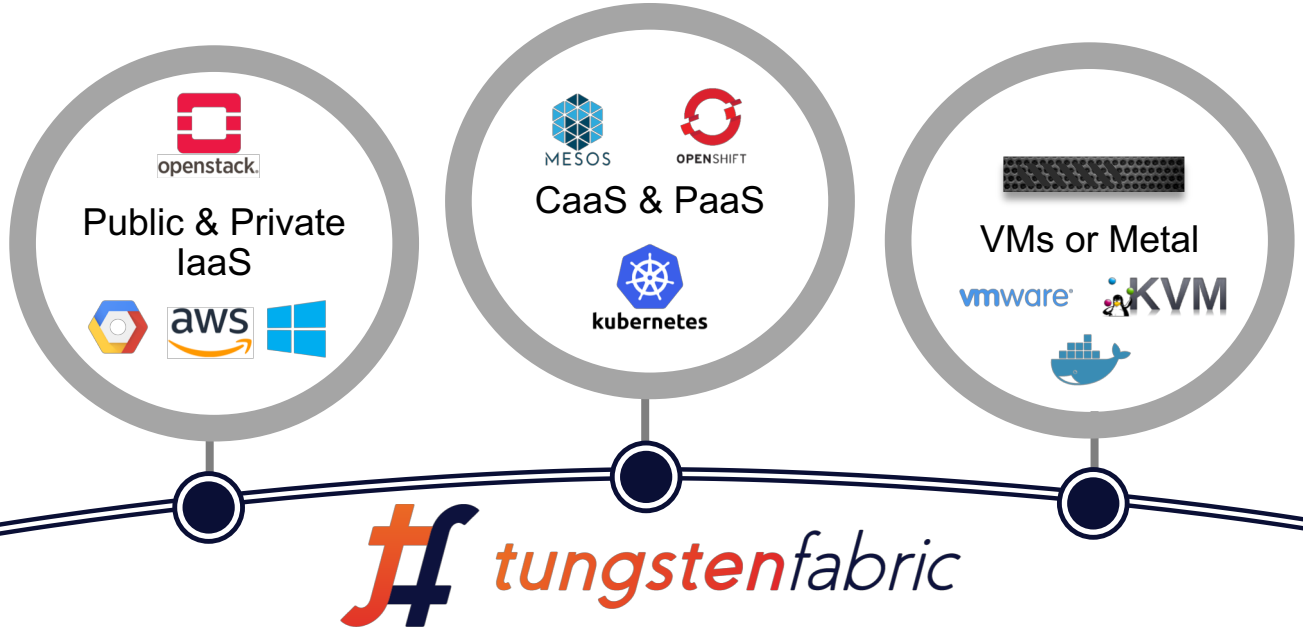
Highly available and ISSU (In Service Software Upgrade) support

Full Fabric Management – Overlay & Underlay Networks

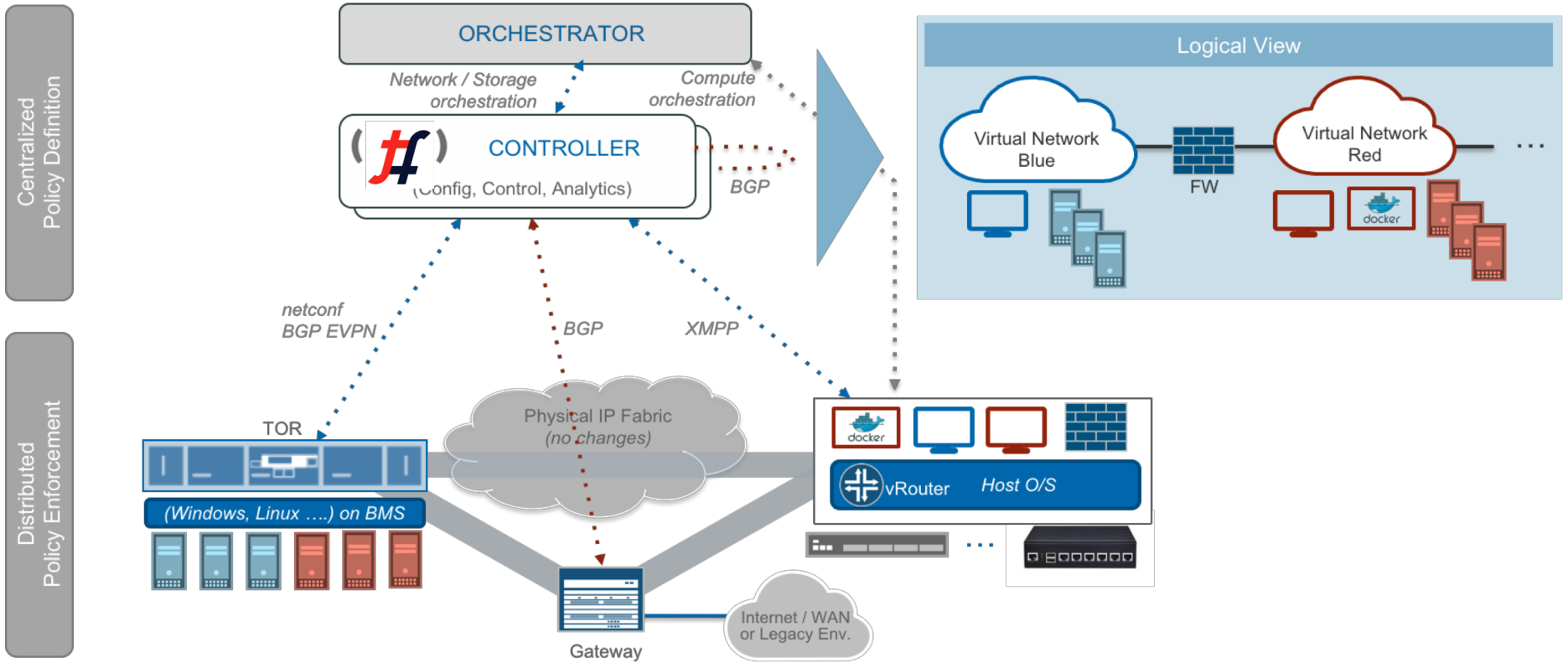
Tungsten Fabric as SDN Controller

RULE THEM ALL WITH ONE

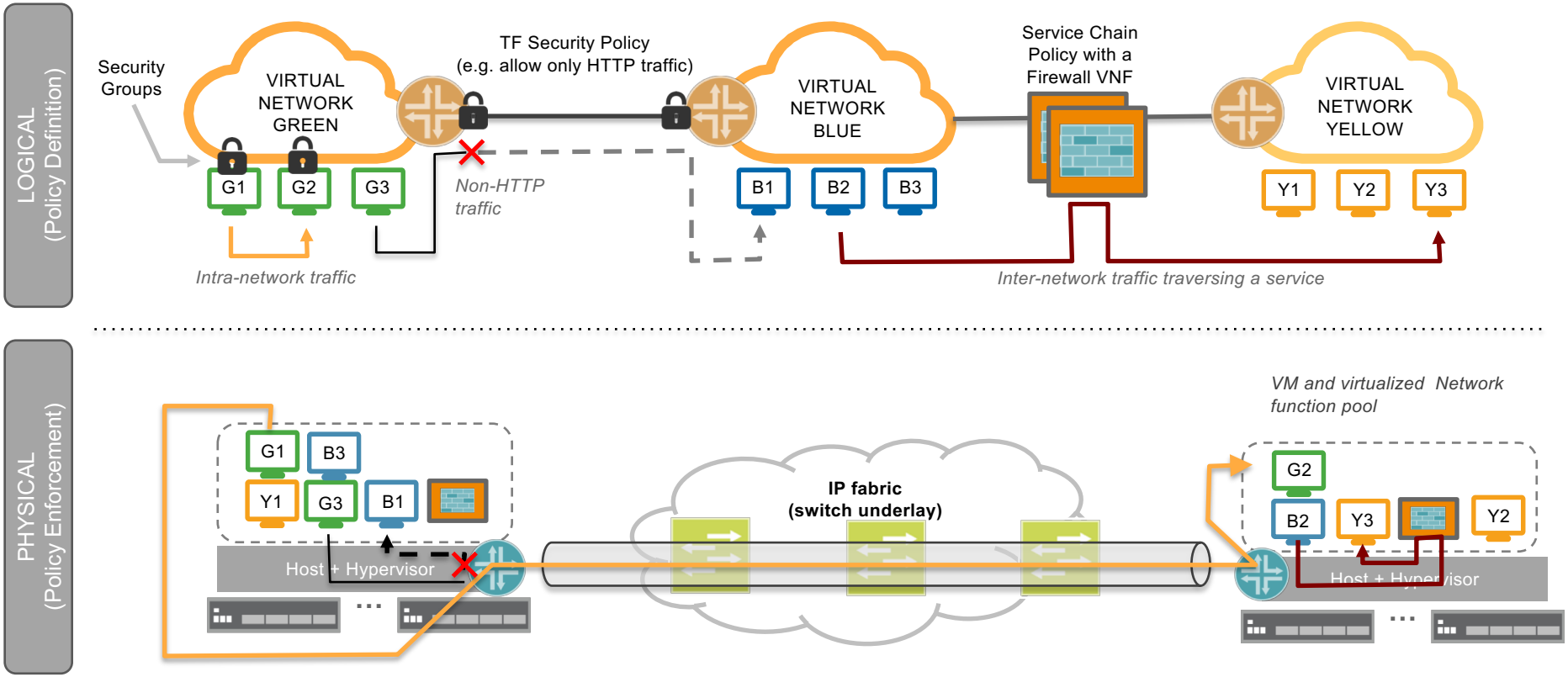
automated secure open SDN Controller



Architecture Overview

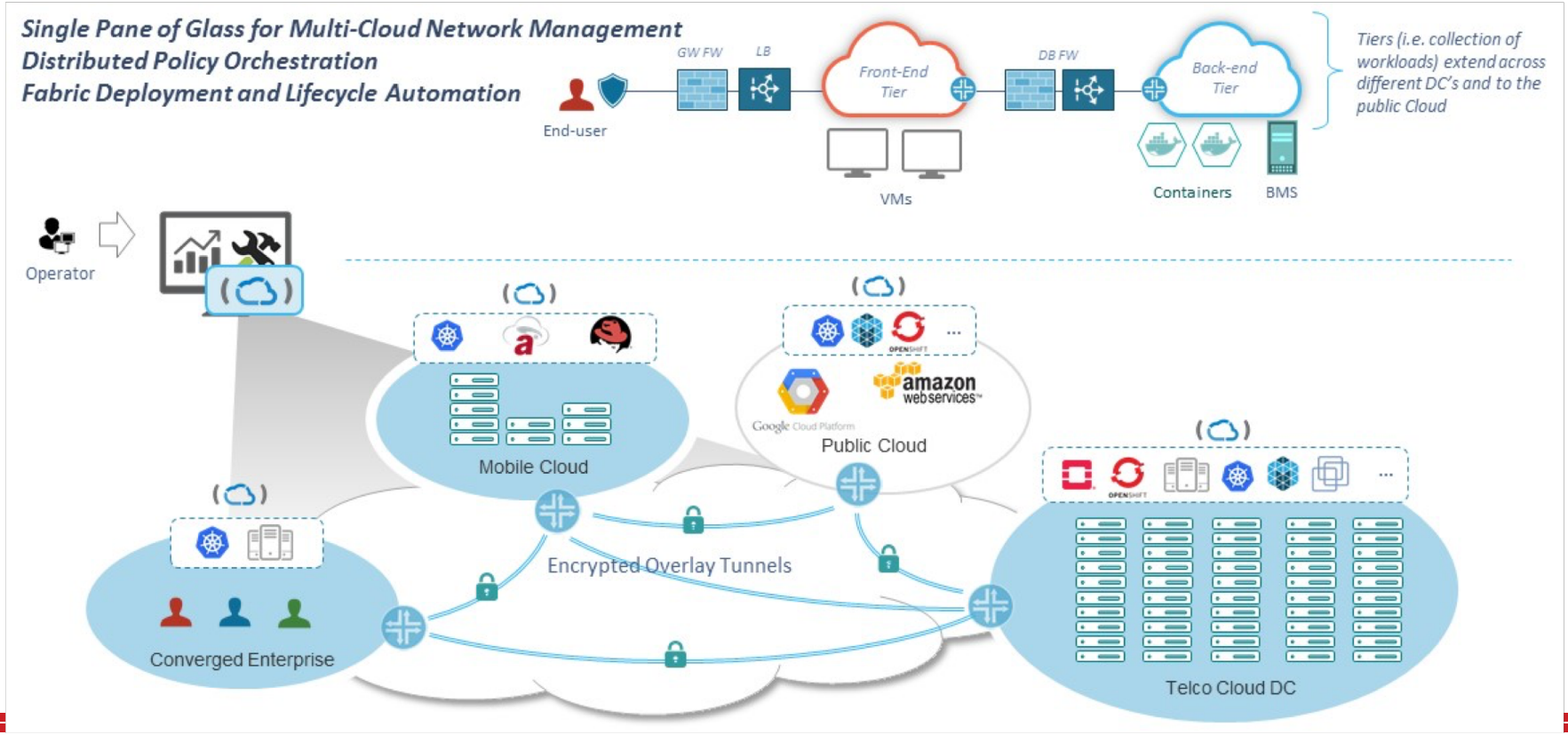


Visualizing Tungsten Fabric's Operational Effects

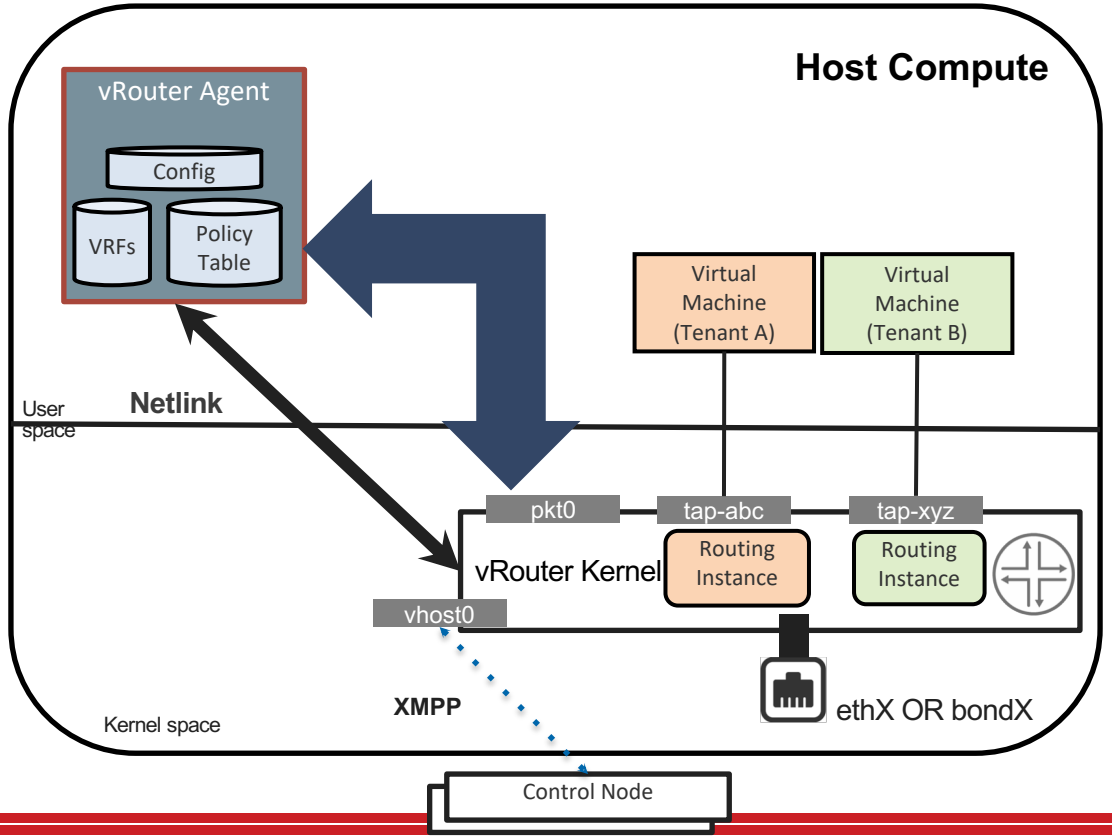


Tungsten Fabric Multi Cloud

Multi-Cloud Networking for Converged Operators



Tungsten Fabric vRouter Architecture & Overview



vRouter Agent

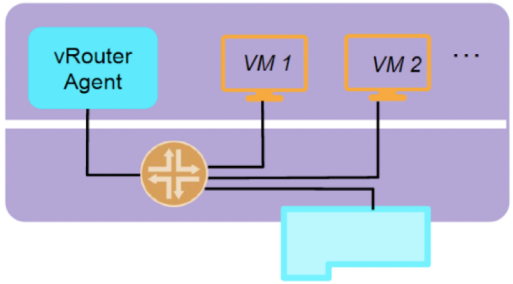
- Exchanging control state such as routes with the Control nodes using XMPP.
- Receiving low-level configuration state such as routing instances and forwarding policy from the Control nodes using XMPP
- Reporting analytics state such as logs, statistics, and events to the analytics nodes.
- Installing forwarding state into the forwarding plane
- Discovering the existence and attributes of VMs in cooperation with the Nova agent.
- Applying forwarding policy for the first packet of each new flow and installing a flow entry in the flow table of the forwarding plane.
- Proxying DHCP, ARP, DNS

vRouter Kernel/DPDK

- Encapsulating packets sent from the overlay network and de-capsulating packets received for the overlay network.
- Packets received from the overlay network are assigned to a routing instance based on the MPLS label or Virtual Network Identifier (VNI).
- Doing a lookup of the destination address of the in the Forwarding Information Base (FIB) and forwarding the packet to the correct destination. The routes may be layer-3 IP prefixes or layer-2 MAC addresses.
- Doing RPF check before sending Virtual machine traffic to destination. This is configurable.

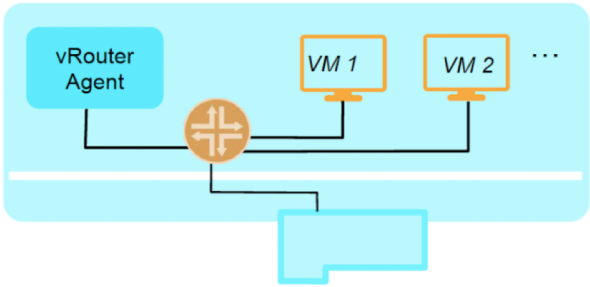
TF VROUTER DEPLOYMENT MODELS

KERNEL VROUTER



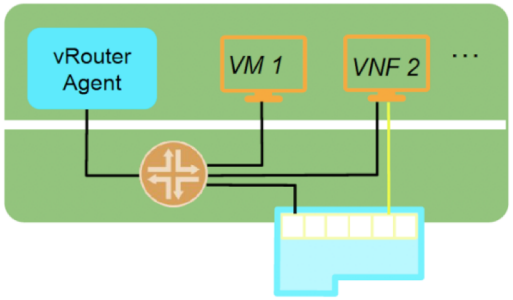
- *This the normal operation where fwding plane of vRouter runs in the kernel and are connected to VMs using TAP interface (or veth pair for containers)*
- *vRouter itself is enhanced using other performance related features:*
  - *TSO / LRO*
  - *Multi-Q Virtio*

DPDK VROUTER



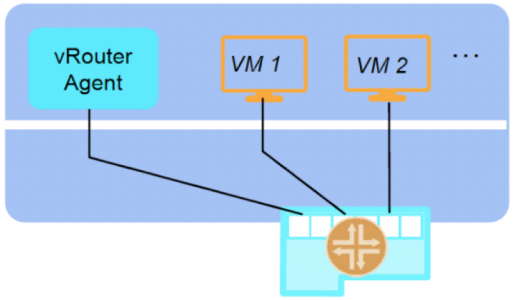
- *vRouter runs as a user space process and uses DPDK for fast path Packet I/O.*
- *Full set of SDN Capabilities Supported*
- *Requires the VMs to have DPDK enabled for performance benefits*

SRIOV/ VROUTER COEXISTENCE



- *Some workloads can directly SRIOV into the NIC, while others go through the vRouter*
- *Sometimes a VNF can have multiple interfaces some of which are SRIOV-ed to the NIC*
- *Interfaces that are SRIOV-ed into NIC don't get the benefits / features of vRouter*

SMARTNIC VROUTER



- *vRouter fwding plane runs within the NIC*
- *Workloads are SRIOV-connected to the NIC*

Tungsten Fabric Evolution to Microservices

- Contrail-Control (5 daemons)
- Contrail-Config (8 daemons)
- Contrail-Analytics (5 daemons)
- Contrail-WebUI (4 daemons)
- Contrail-DB (3 daemons)
- Contrail-vRouter (3 D) + Kernel/DPDK (FP)

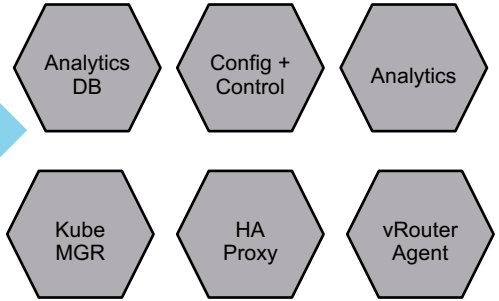
Contrail Controller: 2n+1

OR

BMS

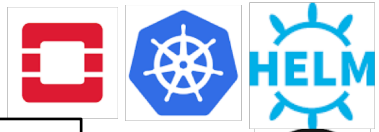
Contrail 1.X/2.X/3.X BMS or VMs base (SDN Controller)

Multiple Process running in one Container (FAT Containers)

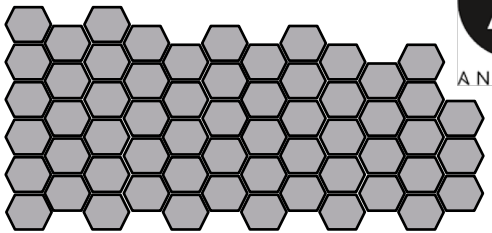


Contrail 4.X (Containers) BMS or VMs base (SDN Controller)

DaemonSet, Ingress Services with Host Networking
with choice of run single or multiple containers per PODs



27-30 Containers Images



Contrail 5.X (Containers) Microservices (SDN Controller)

TF Helm Microservices Architecture *(Helm Charts)*

Ingress
analytics-api
Cluster-SVC-Networking

Service
analytics-ingress
Cluster-SVC-Networking

Service
analytics-api
Cluster-SVC-Networking

POD – DaemonSet (7/7)

- Container (1/7) contrail-analytics-api — Host-Networking
- Container (2/7) contrail-analytics-nodemgr — Host-Networking
- Container (3/7) contrail-collector — Host-Networking
- Container (4/7) contrail-snmp-collector — Host-Networking
- Container (5/7) contrail-query-engine — Host-Networking
- Container (6/7) Contrail-topology — Host-Networking
- Container (7/7) Contrail-alarm-gen — Host-Networking

Contrail-Analytics

POD – DaemonSet (3/3)

- Container (1/4) contrail-control — Host-Networking
- Container (2/4) contrail-dns — Host-Networking
- Container (3/4) contrail-named — Host-Networking

POD – DaemonSet (5/5)

- Container (1/5) contrail-config-api — Host-Networking
- Container (2/5) contrail-config-nodemgr — Host-Networking
- Container (3/5) contrail-svc-monitor — Host-Networking
- Container (4/5) contrail-schema-transf — Host-Networking
- Container (5/5) contrail-device-mgr — Host-Networking

Container (1/4)
contrail-control-nodemgr
Host-Networking

Ingress
config-api
Cluster-SVC-Networking

Ingress
webui
Cluster-SVC-Networking

Service
config-api
Cluster-SVC-Networking

Service
config-ingress
Cluster-SVC-Networking

Service
web-controller
Cluster-SVC-Networking

Service
web-ingress
Cluster-SVC-Networking

POD – DaemonSet (2/2)

- Container (1/2) contrail-webui — Host-Networking
- Container (2/2) contrail-webui-middleware — Host-Networking

Contrail-Controller

- POD - DaemonSet analyticsdb — Host-Networking
- POD - DaemonSet analyticsdb-nodemgr — Host-Networking
- POD - DaemonSet analytics-zookeeper — Host-Networking
- POD - DaemonSet kafka — Host-Networking
- POD - DaemonSet configdb — Host-Networking
- POD - DaemonSet configdb-nodemgr — Host-Networking
- POD - DaemonSet config-zookeeper — Host-Networking
- POD - DaemonSet redis — Host-Networking

Contrail-Third-Party

Kubernetes Cluster

- contrail-config
- contrail-control
- contrail-webui
- contrail-analytics
- contrail-vrouter

Contrail Helm Toolkit

Other Containers:

- **Contrail-status**
- node-init
- vrouter-init-kernel
- vrouter-init-dpdk

POD DaemonSet (2/2)

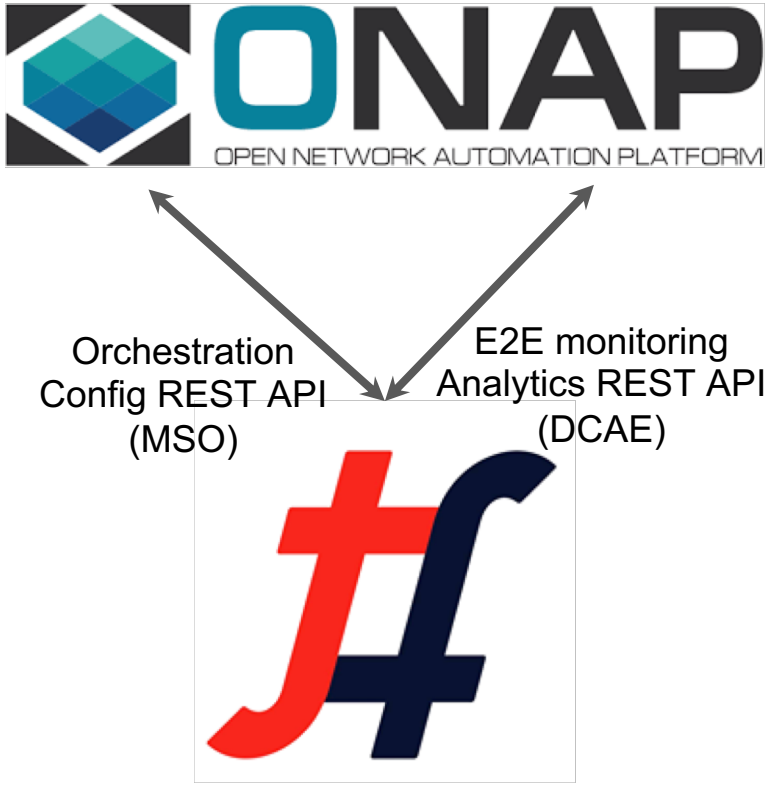
- Container (1/2) contrail-vrouter-agent — Host-Networking
- Container (2/2) contrail-vrouter-nodemgr — Host-Networking

POD DaemonSet (2/2)

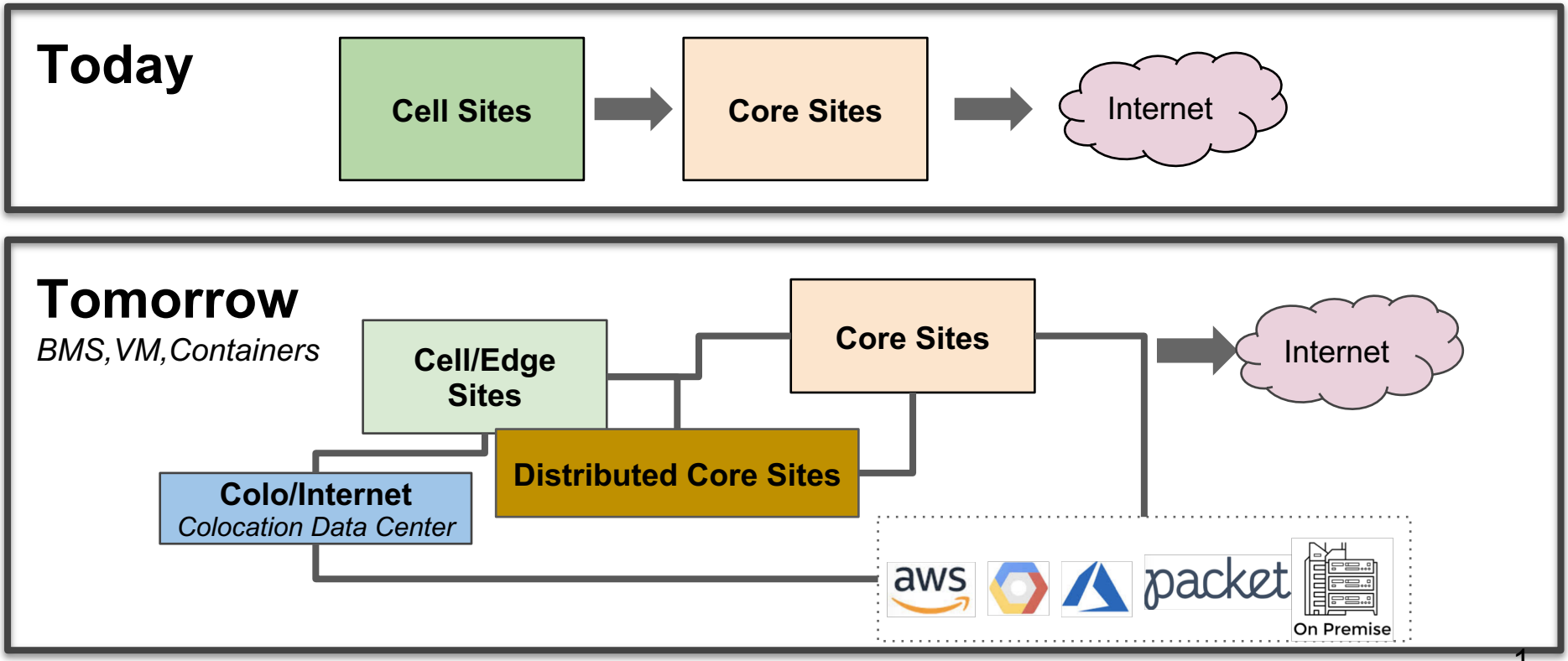
- Container (1/3) contrail-vrouter-agent-dpdk — Host-Networking
- Container (2/3) contrail-vrouter-dpdk — Host-Networking
- Container (3/3) contrail-vrouter-nodemgr — Host-Networking

Contrail-vRouter

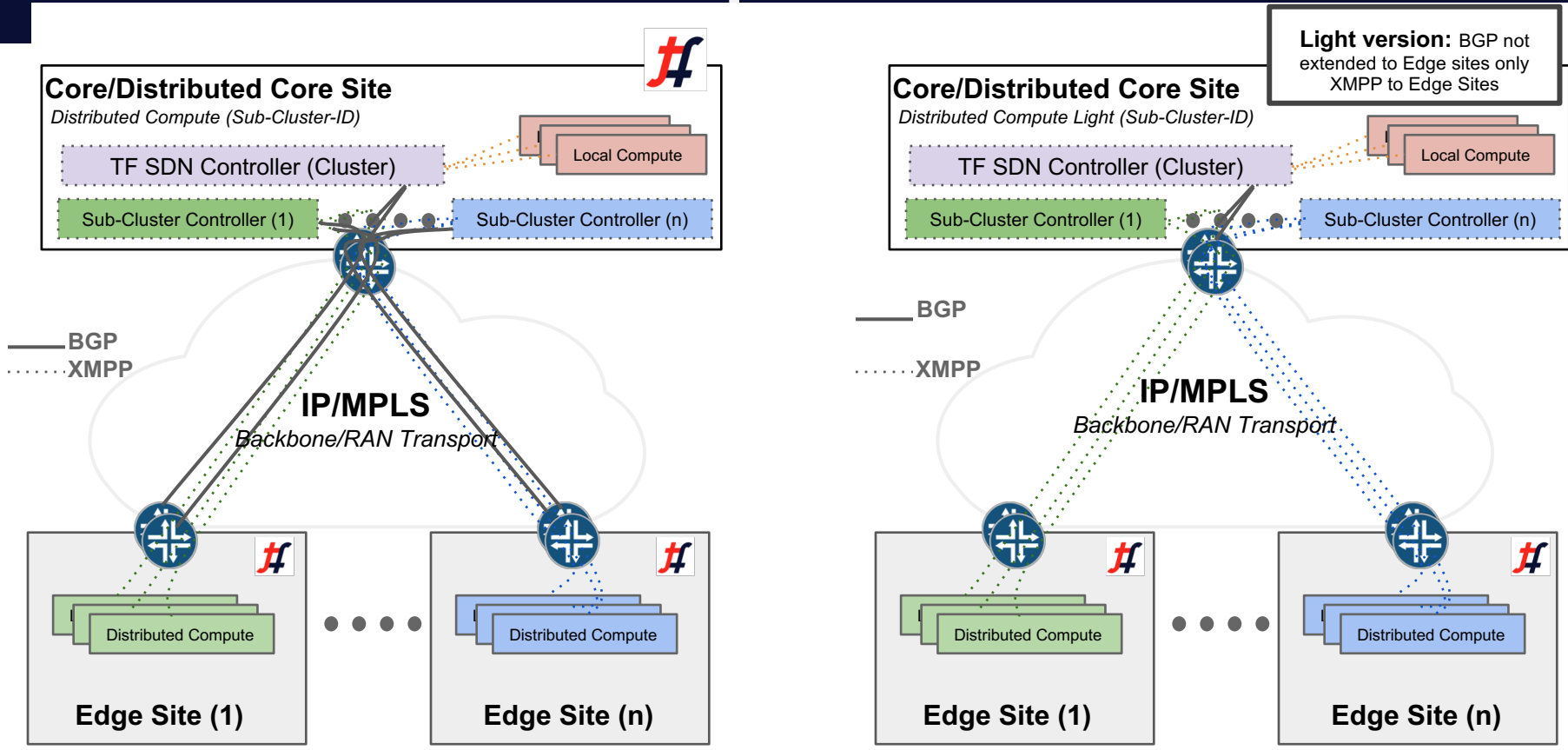
Tungsten Fabric Integration with ONAP



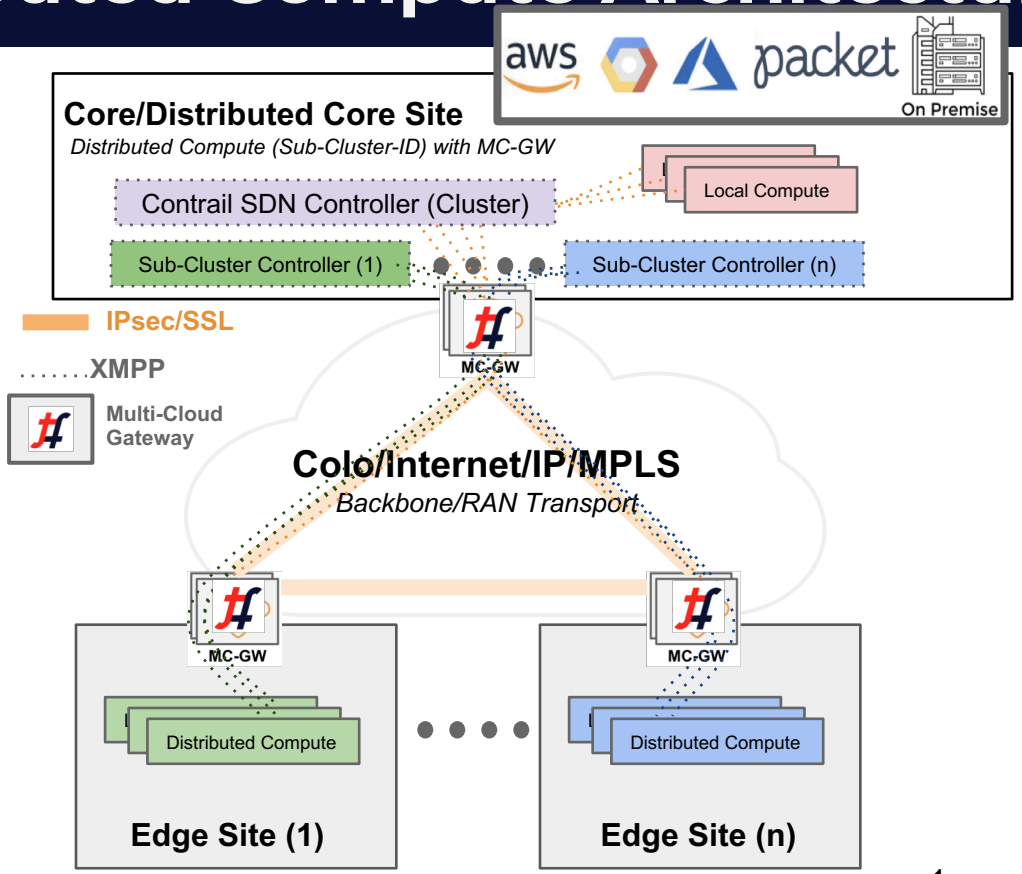
Edge Computing (Today & Tomorrow)



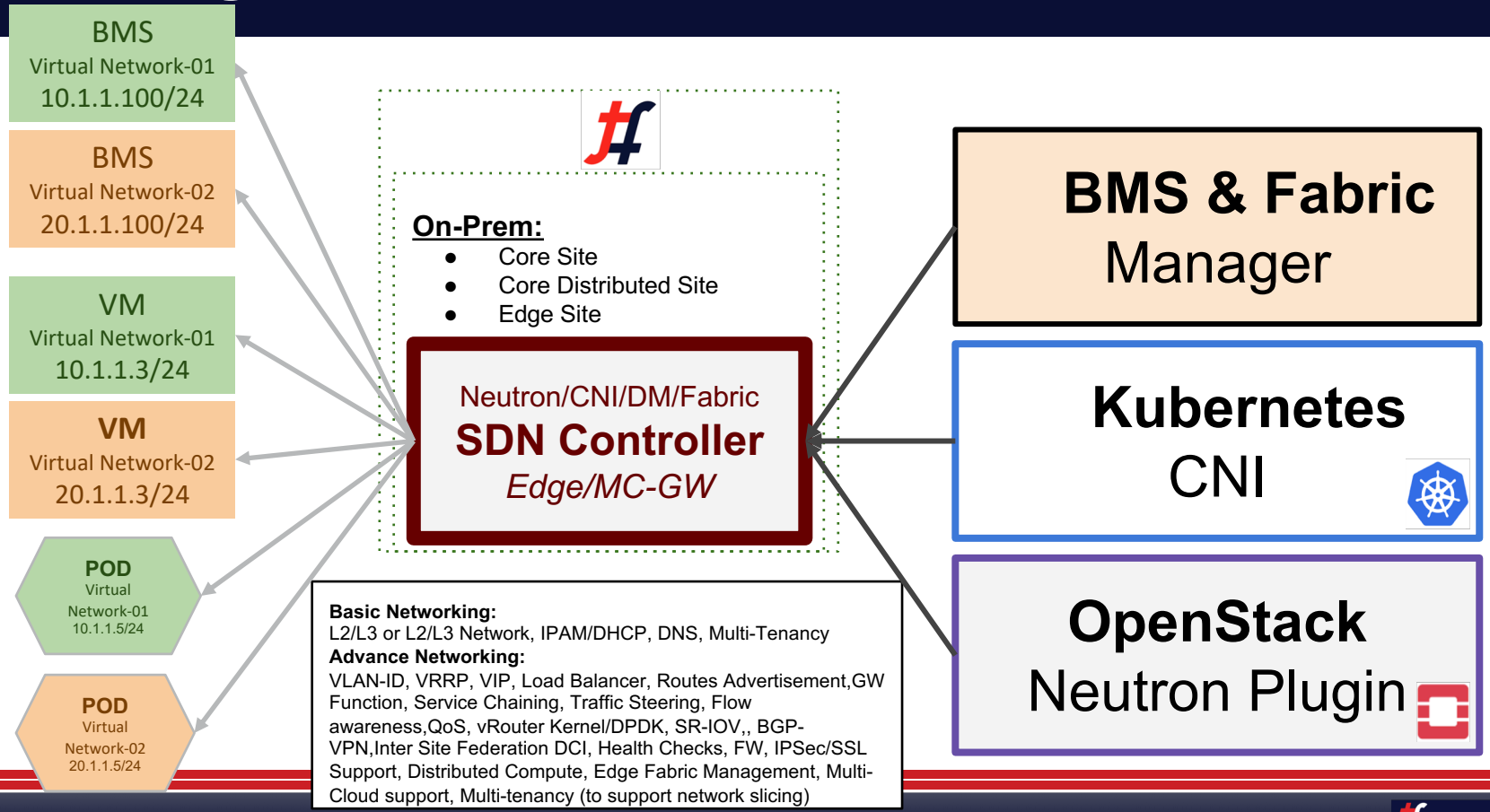
TF Distributed Compute Architecture



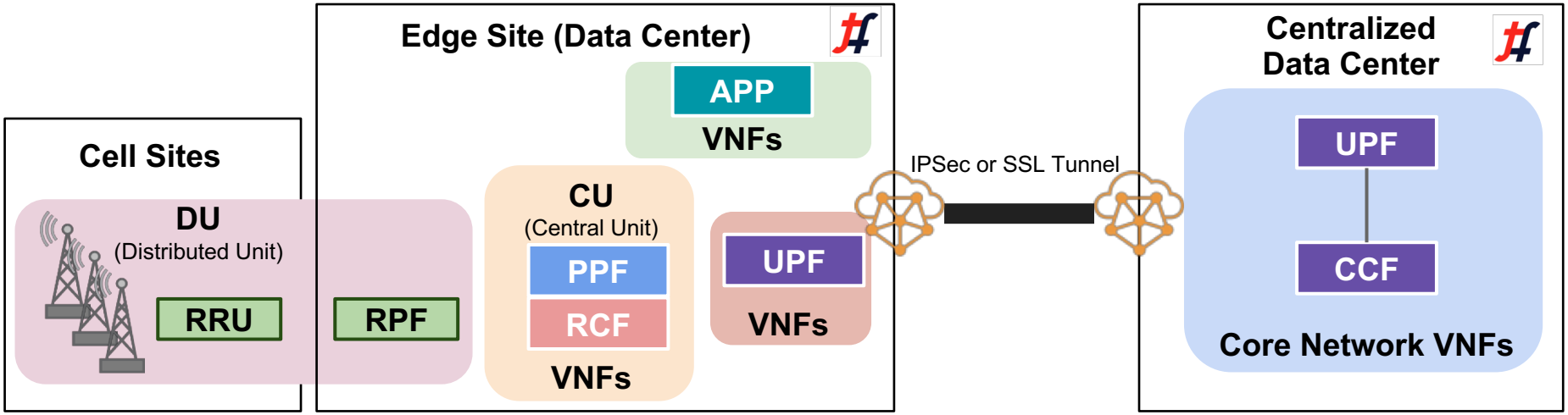
TF Distributed Compute Architecture



TF as Single SDN for VMs, PODs & BMS



5G Edge Computing and Encryption



Secure RAN to CN

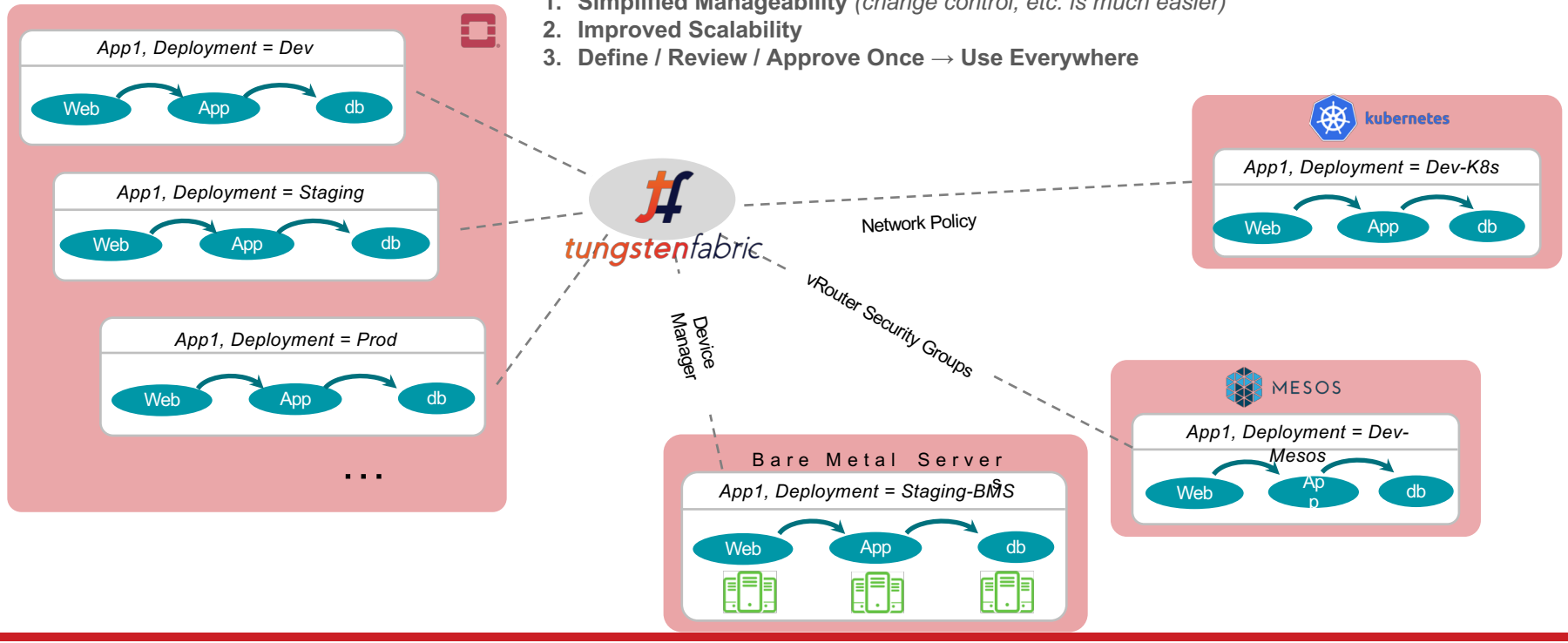
- Use Contrail Encryption to secure Remote Edge and Central DC connection.
- Secure Overlay site to site communication via Contrail encryption support
- Policy based encryption model

APP Application
CCF Core Control Function (Core Network)
UPF User Plane Function (Core Network)
RCF Radio Control Function (RAN)
PPF Packet Processing Function (RAN)
RPF Radio Processing Function (RAN)
RRU Remote Radio Unit (RAN)

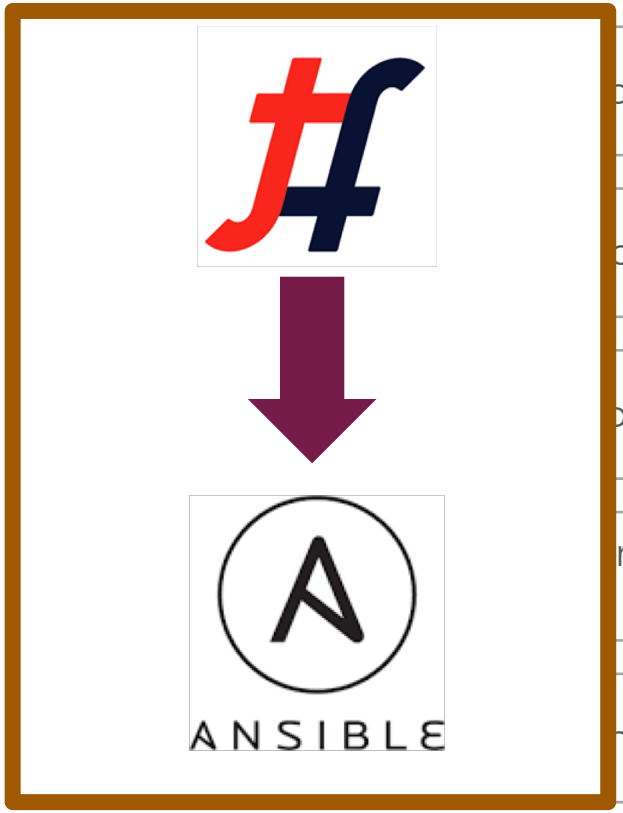
SOFTWARE DEFINED SECURE NETWORKING

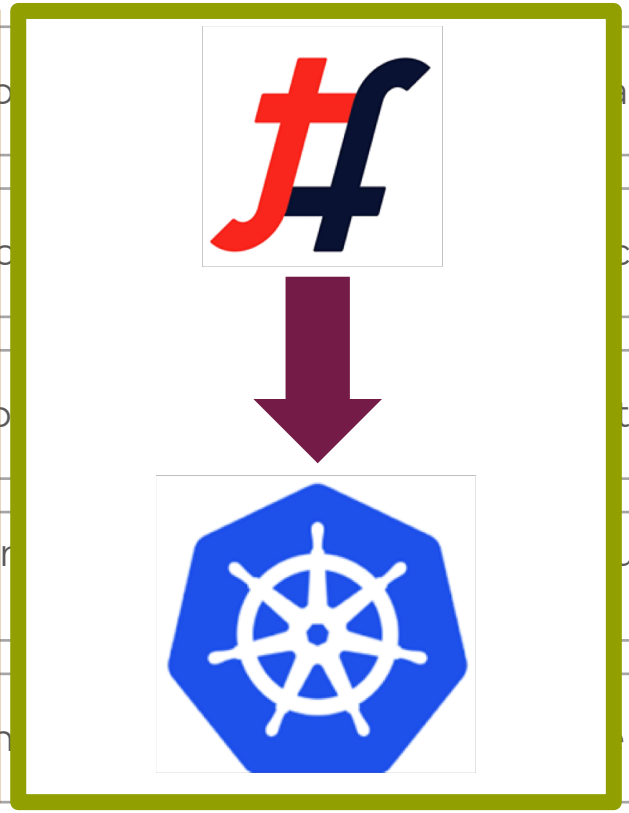
Tungsten fabric provides a rich, consistent set of security policy capabilities across multiple platforms.

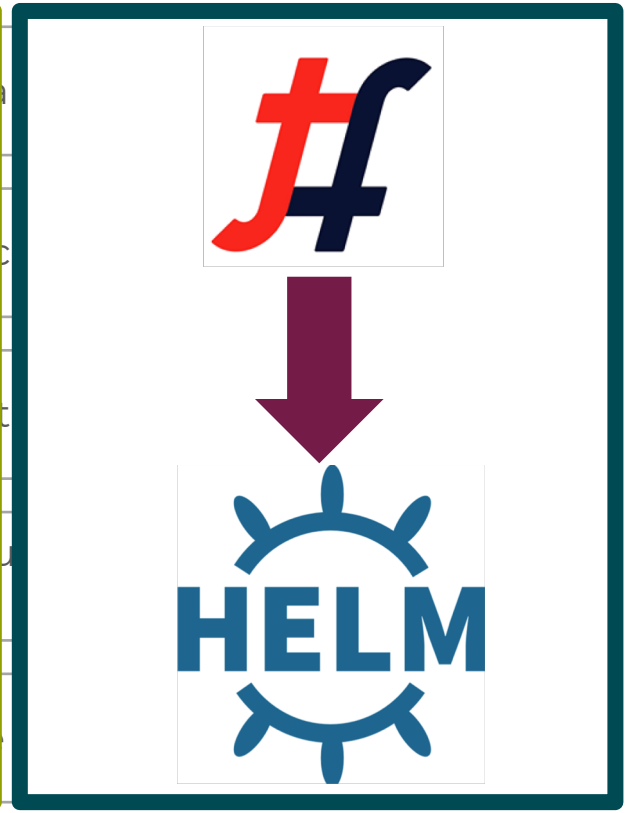
1. **Simplified Manageability** *(change control, etc. is much easier)*
2. **Improved Scalability**
3. **Define / Review / Approve Once → Use Everywhere**



Tungsten Fabric INSTALLATION







Tungsten Fabric K8s CNI (A single YAML Install & CARBIDE)

CentOS Single YAML

Ubuntu Single YAML

tungstenfabric CARBIDE

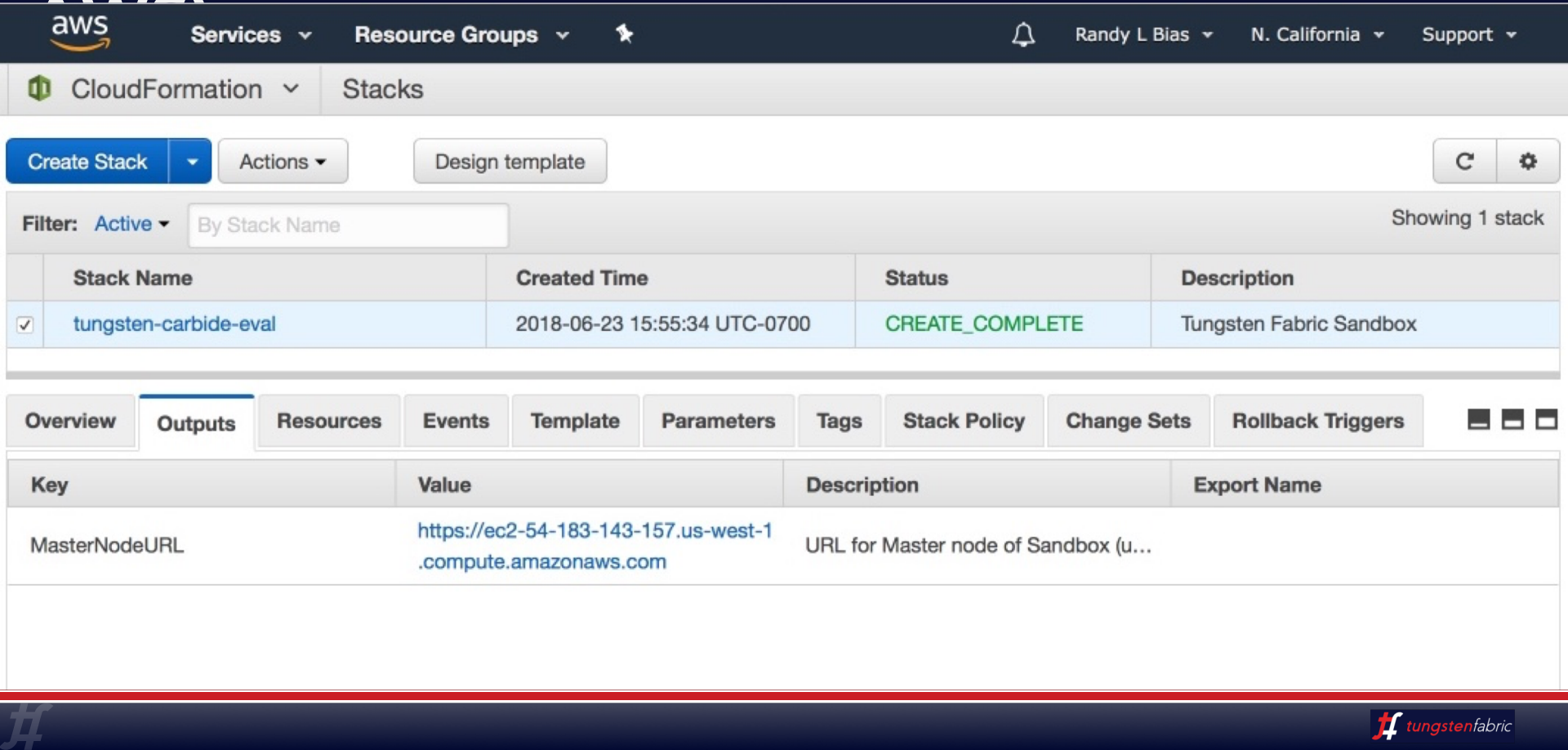
Reference: https://github.com/Juniper/contrail-kubernetes-docs

Carbide Sandbox Environment

Tungsten Fabric + Kubernetes on AWS

https://tungsten.io/start/

0-60 in 15 Minutes Flat w/Carbide (TF+k8s on AWS)



0-60 in 15 Minutes Flat w/Carbide (TF+k8s on AWS)

Carbide Evaluation System

tungstenfabric

Deployment is in progress:

Please wait until the deployment ends.

23:27:46 UTC: 1/6 The control site is being deployed ...
23:28:45 UTC: 2/6 Creating and exporting a key pair ...
23:28:46 UTC: 3/6 Download the repository ...
23:28:46 UTC: 4/6 Provision instances ...
23:31:02 UTC: 5/6 Configure instances ...
23:33:52 UTC: 6/6 Install Kubernetes and Tungsten Fabric ...

0-60 in 15 Minutes Flat w/Carbide (TF+k8s on AWS)

Deployment is completed

Contrail UI: https://ec2-54-215-222-33.us-west-1.compute.amazonaws.com:8143

User name: *admin*
User password: *contrail123*

To use Tungsten Fabric or Kubernetes command line utilities, connect to controller using the key specified during the deployment of CloudFormation stack and **centos** user name.

Example:

```
ssh -i randyb-carbide-test.pem centos@ec2-54-215-222-33.us-west-1.compute.amazonaws.com
```

Accessing the Kubernetes dashboard:

On the controller:

```
kubectl get pods -n kube-system -o wide | grep dashboard
```

Check the IP column. It tells you the private IP address of the compute node where the dashboard POD is running. You need to find out the associated public IP address (it is left to you as an exercise). Once you know it, you can connect to the URL:

```
https://<public-ip>:8443
```

Select the token option. Where can you get the token from? There is one on the controller's file /root/k8s_dashboard_token.txt , but it only allows to visualize. If you want read-write access do the following:

```
kubectl get secret -n contrail | grep kubemanager
kubectl describe secret -n contrail | grep "token:" | awk '{print $2}'
```

Take your time to browse the dashboard. During the next exercises, you can choose to do some tasks on the web instead of (or in addition to) the CLI.

Carbide EC2 Instances overview

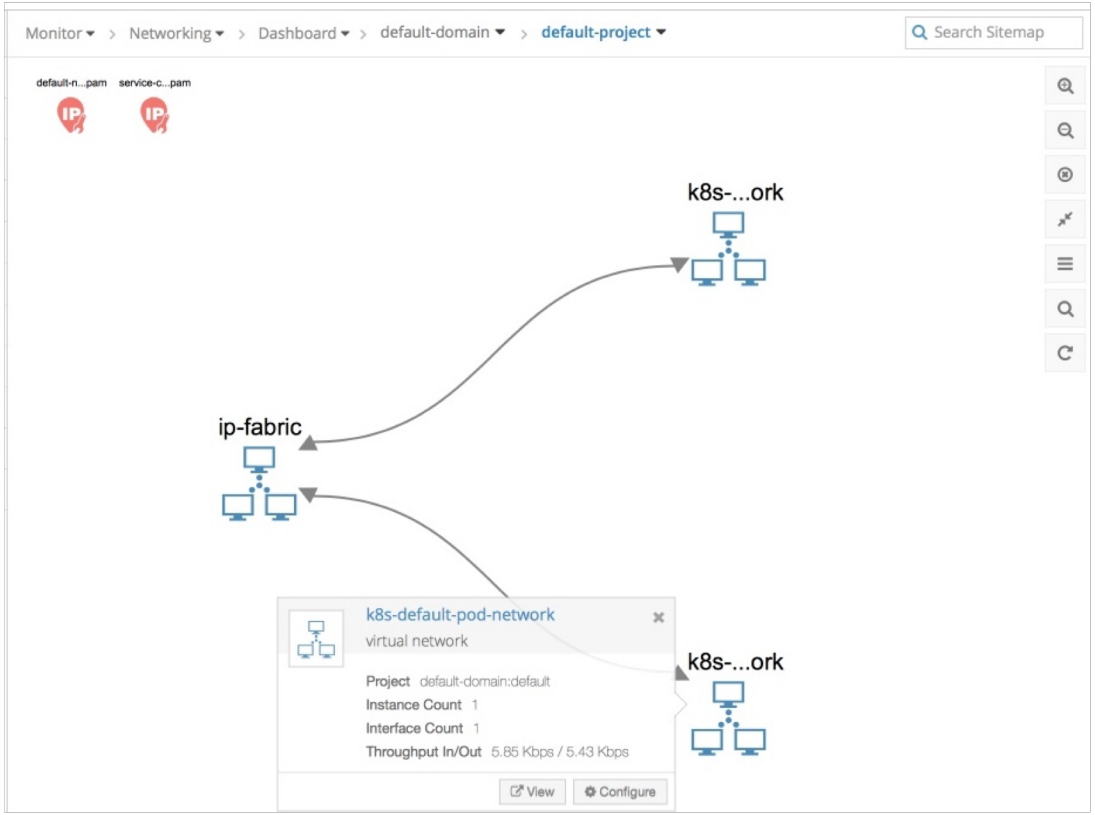
tungstenfabric-k8s-aws-master-node

tungstenfabric-k8s-aws_control1

tungstenfabric-k8s-aws_compute1

tungstenfabric-k8s-aws_compute2

0-60 in 15 Minutes Flat w/Carbide (TF+k8s on AWS)



Try Tungsten Fabric

https://tungstenfabric.github.io/website/Tungsten-Fabric-15-minute-deployment-with-k8s-on-AWS.html