Contrail MultiCloud Security and Analytics for Containers
Technical Overview
Marco Chiandusso, SE Cloud Specialist
CONFIDENTIALITY AND LEGAL NOTICE

This material contains information that is confidential and proprietary to Juniper Networks, Inc. Recipient may not distribute, copy, or repeat information in the document.

This statement of product direction sets forth Juniper Networks’ current intention and is subject to change at any time without notice. No purchases are contingent upon Juniper Networks delivering any feature or functionality depicted in this presentation.

Contrail program participants are subject to a license agreement that describes program terms and conditions.
Centralized Network Policy and Security Control

- Connect multiple private and public clouds using an auto-forming mesh of secure tunnels
  - Consume a common network/security interface for all cloud deployments
- Define security policies centrally for all endpoints using dynamic policy generation, and tag based rules
- Manage & Operate cloud networking and security from a single interface
Kube-network-manager listens to K8s API Server and conveys the API request to Contrail Controller.
Contrail provides a collapsed control and data plane in which a single Contrail control plane and a single virtual network stack manage and service both the OpenStack and Kubernetes clusters.

- Single Contrail Controller to provide networking to pods, VMs & baremetal

- Run Kubernetes on OpenStack
  - Launch VMs (K8-master & K8-slave) from OpenStack horizon in a publicly accessible virtual-network
  - Install Kubernetes on the VMs created
  - Create a custom virtual-network. Launch a VM in this network from OpenStack UI & a pod from K8s master node
CONTRAIL INGRESS SERVICE AND SERVICE LB

Namespace - B
External IP (Virtual hosts and URL based routing)

Namespace - A

Service (VIP, Port)... Service IP allocated by K8s IPAM
External IP

Service 1
qa-webservice

Application 1
(Load balancing across multiple PODs done using ECMP-LB)

POD 2

POD 1 (POD1-IP allocated by Contrail)

C1 C2 ...

Containers

Repl. Ctrl

POD 6

POD 5

C1 C2 ...

Containers

Application 2
(dev-webservice)

(Load balancing across multiple PODs using ECMP-LB)

Master Node #2

Master Node #1

API Server
Controller / Replication Mgr
Scheduler
Data Store (etcd)

Contraill Ingress Services

#RedHatOSD
CONTAINER NETWORKING
- DIFFERENT LEVELS OF ISOLATION

**DEFAULT CLUSTER MODE**
- This is how K8s networking works today
- Flat subnet where -- Any workload can talk to any other workload

**NAMESPACE ISOLATION**
- In addition to default cluster, operator can add isolation to different namespaces transparent to the developer

**POD / SERVICE ISOLATION**
- In this mode, each POD is isolated from one another
- Note that all three modes can co-exist

---

**Namespace - A**
- S1
- S2
- POD 1
- POD 5
- POD 9
- POD 13

**Namespace - B**
- S3
- S4
- POD 9
- POD 13

**Namespace - C**
- S5
- S6
- POD 17
- POD 21

**Namespace - D**
- S7
- S8
- POD 25
- POD 29

**Namespace - E**
- S9
- S10
- POD 33
- POD 37

**Namespace - F**
- S11
- S12
- POD 41
- POD 45
Kubernetes Network Policy is a framework for defining firewall rules via the Kubernetes Manifests.

Network Policy defines the rules framework, but not the enforcement. It is up to the CNI to enforce the Network Policy.

Rules can be based on Kubernetes labels, to flexibly define firewall filters, or network/port/protocol to define more rigid matches.

Contrail supports enforcement of Network Policy, extending the label based firewall policy to support OpenStack, VMWare, and Bare Metal.

Contrail maintains flow logs, and policy analytics to provide insight and audit support for connected virtual networks.
INTENT-DRIVEN POLICY OPTIMIZATION
- WRITE ONCE – DEPLOY MANY

Once a set of policies are defined for a particular OpenStack environment, they can easily be re-used for other environments?

1. Reduced Complexity (less # of policies)
2. Simplified Manageability (change control, etc. is much easier)
3. Improved Scalability
4. Define / Review / Approve Once → Use Everywhere

App1, Deployment = Dev-AWS
App1, Deployment = Dev-K8s
App1, Deployment = Dev-OpenShift
App1, Deployment = Dev
App1, Deployment = Staging
App1, Deployment = Staging-BMS
App1, Deployment = Prod

Bare Metal Servers
INTENT-DRIVEN SECURITY POLICY WITH CONTRAIL & K8S

allow tcp 80 tier=web > tier=app match deployment && site
allow tcp 3036 tier=app > tier=db match deployment && site

Namespace = N1 (Finance-Dev)
K8s Pod – finance-dev
"app" : "finance"
"tier" : "frontend"
"deployment" : "dev"

K8s Pod – finance-dev
"app" : "finance"
"tier" : "app"
"deployment" : "dev"

K8s Pod – finance-dev
"app" : "finance"
"tier" : "backend"
"deployment" : "dev"

Namespace = N2 (Finance–Prod)
K8s Pod – finance-prod
"app" : "finance"
"tier" : "frontend"
"deployment" : "prod"

K8s Pod – finance-prod
"app" : "finance"
"tier" : "app"
"deployment" : "prod"

K8s Pod – finance-prod
"app" : "finance"
"tier" : "backend"
"deployment" : "prod"

<table>
<thead>
<tr>
<th>Tags</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>tier</td>
<td>web</td>
</tr>
<tr>
<td>tier</td>
<td>app</td>
</tr>
<tr>
<td>tier</td>
<td>db</td>
</tr>
<tr>
<td>application</td>
<td>finance</td>
</tr>
</tbody>
</table>

Custom Global labels

<table>
<thead>
<tr>
<th>Tags</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>deployment</td>
<td>dev</td>
</tr>
<tr>
<td>deployment</td>
<td>staging</td>
</tr>
<tr>
<td>deployment</td>
<td>prod</td>
</tr>
<tr>
<td>site</td>
<td>US</td>
</tr>
<tr>
<td>site</td>
<td>EMEA</td>
</tr>
</tbody>
</table>
MANAGING THE CLOUD

Software Defined Operations

Stream analysis to monitor SLAs and predict faults

Cross Layer Visibility

Real-time optimizations to improve efficiency and ensure service availability
CONTRAIL VALUE PROP FOR CONTAINERIZED ENVIRONMENTS

- Create secure multi-tenant container environments, with existing application developer workflow
- Offer multiple deployment options (i.e. bare metal server, Private / Public Clouds, etc.)
- Seamless migration & interop of existing Contrail (non-container) environments with a container environment
- Extend all vRouter features (QoS, Floating IP, DDI, etc.) to a container environment
- Allow Operator to modify infra security (& isolation) levels, transparent to app developer
GRAZIE PER L’ATTENZIONE

MARCO CHIANDUSSO – SE CLOUD SPECIALIST