This statement of 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.

This presentation contains proprietary roadmap information and should not be discussed or shared without a signed non-disclosure agreement (NDA).
AGENDA

- Background and Evolution of Technology and Ecosystem
- Fundamentals of Containers - what and why
- Microservices and Container Orchestration
- Contrail Networking for Containers
CONTAINERS
THE RISE OF CONTAINERS & MICROSERVICES

Containers:
- Standardized frame for all services.
- Simplified a painful integration process in a heterogeneous infrastructure world.
- Docker - revolution in how developers build and deploy applications w/Containers.

APIs:
- The rapid adoption of APIs has created a standardized format for communications.

Scalable cloud infrastructure:
- Private or public, delivers the resources needed on demand to scale, and operate services effectively.
- Cloud is the new computer!

https://www.sequoiacap.com/article/build-us-microservices/
WHAT ARE CONTAINERS?

- Processes isolated from the host and (optionally) other containers
- Share the same underlying Kernel

Control Groups (cgroups):
Virtualize by sharing and limiting access:
- CPU,
- Memory,
- Disk I/O,
- N/w I/O

Namespaces:
Virtualize by isolating:
- User IDs
- Process IDs & tree
- Filesystem mounts
- Network interfaces

Security:
- SELinux policy and enforcement control over all resources
- AppArmor to restrict a program’s abilities
- Linux capabilities etc.

- Virtual network interfaces / addresses (maybe host NAT’d)
- Files and optional (shared) mounts from the host filesystem
VIRTUAL MACHINES VS CONTAINERS

Host OS + Hypervisor

Infrastructure

Host OS

Infrastructure

App A

Binaries / Libraries

Guest OS

App B

Binaries / Libraries

Guest OS

App C

Binaries / Libraries

Guest OS

Container Engine
THE PROMISE OF DOCKER

Build Once, Run Anywhere
DOCKER NETWORKING

- Bridged (docker0)
- Bridged MacVLAN L2

- Host mode
- L3 mode
DOCKER NETWORKING

- Docker provides:
  - Default networking
  - Mac-vlan
  - VXLAN Overlay
  - “multi-host” networking ;)

- Plugins

MICROSERVICE CONCEPT

MICROSERVICE CONCEPT

Containers are the perfect vehicle for microservices.

Containers orchestration platforms are suited to handle the deployment and scaling of these microservices.
CONTAINERS & MICROSERVICES ORCHESTRATION
The name?
Greek for “pilot” or “helmsman of a ship”

The project?
Open Source project originally designed by Google (Project Borg) and donated to the CNCF

“Open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts, providing container-centric infrastructure”

Desired state / Intent-based system
“Tell Kubernetes what you want and it will take care of everything else”
WHAT IS KUBERNETES?

Application Focus
- Deploy / scale / rolling updates / control resources

Portable
- Run everywhere (public/private cloud, bare metal, VMs, laptop)

Self-Healing
- Auto-restart, auto-rescheduling, auto-scaling, auto-replication

Bridge the Dev/Ops gap
- Taking advantage of application containerization and micro services
- CI/CD environments
- Mechanisms to tie the infrastructure and the applications running on it
**KUBERNETES ARCHITECTURE**

**Namespace - A**

- **Service (VIP, Port)**
- **Service - S1**
- **Application 1** (Load balancing across multiple PODs)

**POD 1 (POD1-IP)**

- **Containers**
  - C1
  - C2
  - ...

**POD 2**

**Service (VIP, Port)**

**Service - S2**

**Application 2** (Load balancing across multiple PODs)

**POD 5**

- **Containers**
  - C1
  - C2
  - ...

**POD 6**

**Repl. Ctrl**

**Namespace - B**

**POD 2**

**Service (VIP, Port)**

**Service - S1**

**Application 1** (Load balancing across multiple PODs)

**POD 1 (POD1-IP)**

- **Containers**
  - C1
  - C2
  - ...

**POD 5**

**Service (VIP, Port)**

**Service - S2**

**Application 2** (Load balancing across multiple PODs)

**POD 6**

**Repl. Ctrl**

**Master Node #2**

- **kube-apiserver**
- **kube-controller-manager**
- **kube-scheduler**

**Data Store** (etcd)

**Master Node #1**

**Minion / Node**

**Kubelet**

**Minion / Node**

**Kubelet**

**Master**
OPENSHIFT
Red Hat’s Container Application Platform (PaaS)

Self Service
• Templates
• Web Console

Multi-language

Automation
• Build
• Deploy

Collaboration
• DevOps

Security
• NameSpaces
• RBAC

Scaleable
• Integrated Loadbalancer

Open Source

Enterprise Grade
• Authentication
• Web Console
• Central Logging

source: www.redhat.com
CONTRAIL NETWORKING FOR CONTAINERS
CONTRAIL ARCHITECTURE

**Centralized Policy Definition**

- **ORCHESTRATOR**
  - Network / Storage orchestration
  - Compute orchestration

**Distributed Policy Enforcement**

- **CONTRAIL CONTROLLER**
  - (Config, Control, Analytics, Svr Mgmt)
  - OVSDB
  - BGP
  - XMPP

**Physical IP Fabric**

- (Windows, Linux ...) on BMS

**Logical View**

- Virtual Network Blue
- Virtual Network Red
- FW
- Gateway

**Internet / WAN or Legacy Env.**
CONTAINERIZED CONTRAIL

Containerizing Contrail Control Plane – for easier manageability

Docker containers orchestrated using K8s or other orchestration tools

SALIENT ASPECTS

- Multiple personalities of containers:
  - 3 controller container – (Controller, Analytics, Analytics DB) each representing a node
  - LB to enable HA (based on HAProxy) will be provided as container not a mandatory item
  - vRouter Agent on containers

- Containers are deployed using either Contrail Server Manager / K8s / Helm Charts
- Each of the nodes can independently scale (3 x)
- Can be deployed on Bare Metal or VMs
- No change in the role / functionality of the Control / config / analytics nodes

BENEFITS

- LCM is simplified [All dependencies within the container (easy bring up)]
- Accelerate Contrail provisioning
- Integration with 3rd party provisioning tools simplified
CONTRAIL ARCHITECTURE WITH KUBERNETES
CONTRAIL WITH KUBERNETES
CONTRAIL VALUE PROP

What do the users / app team get?
- Distributed LB
- Security / Isolation / multi-tenancy
- External / Public access from within cluster (SNAT) and vice versa (using floating IP)
- Exposing cluster to Enterprise network outside the cluster

Increasing Levels of Isolation

- Define Networks and Security Policies in Contrail
- Use Network/Security Policy per namespace and/or pods using annotations
- Virtual Network per namespace
- Solution per namespace
- All Pods and Services are reachable

User-Defined Isolation
Custom Isolation
Namespace Isolation
Default Cluster Network Mode
KUBERNETES TO CONTRAIL OBJECT MAPPING

Namespace
Pod
Service
Ingress
Network Policy

Single project OR Shared project
Virtual Machine
ECMP Loadbalancer
Haproxy Loadbalancer for URL
Security Groups
CONTRAIL WITH KUBERNETES ON BAREMETAL
CONTRAIL WITH KUBERNETES ON OPENSTACK

- Openstack + Contrail Controller
  - Nova API
  - Glance
  - Keystone
  - Controller
  - Analytics
  - Analytics-DB

- Openstack Compute 1
  - K8s Master
  - K8s API Server
  - Kube Manager
  - POD
  - K8s Kubelet
  - CNI

- Openstack Compute 2
  - K8s Node2
  - Agent
  - Green VM

- Nova API
- Glance
- Keystone
- Controller
- Analytics
- Analytics-DB

- Pod
- K8s Kubelet
- CNI

- Agent
- Green VM

- vrouter

- Green VM
- vrouter

- Agent
CONTRAIL WITH OPENSSHIFT

OpenShift Enterprise
- User Experience
- Enterprise Management & Integration
- Container Development

Kubernetes
- Container orchestration

Docker
- Container runtime environment

Atomic
- Minimal OS

Build
- IDM(LDAP, SSO)
- Web-Console
- JBOSS xPaas images
- Eclipse & Jenkins integrations
- Router (Now Ingress in K8S)
- **OpenShift SDN for Isolation**
- Logging & Metrics

Deploy

Run

Host

Contrail SDN augments OpenShift SDN