

# БЕЗОПАСНОСТЬ МИКРОСЕРВИСОВ КОНТЕЙНЕРНЫЙ CSRX

Дмитрий Карякин

Старший системный инженер, JNCIE-DC/ENT

JUNIPER | Summit  
NETWORKS

# ЗАЯВЛЕНИЕ ОБ ОТКАЗЕ ОТ ОТВЕТСТВЕННОСТИ

Juniper может раскрывать информацию, связанную с разработкой и планами выпуска будущих продуктов, функций или улучшений (заявление о направлении развития продукта, «ЗОНРП»). Информация ЗОНРП может быть изменена в любое время без уведомления. За исключением случаев, когда это может быть указано в окончательных соглашениях о потенциальной сделке, Juniper не дает никаких гарантий и не несет никакой ответственности за то, что будущие продукты, функции или улучшения будут представлены. За исключением случаев, когда это может быть указано в окончательных соглашениях о потенциальной сделке, Компания не должна основывать решения о покупке на основании периодов времени или других данных, изложенных в ЗОНРП, поскольку Juniper может задержать или никогда не представить будущих продуктов, функций или улучшений.



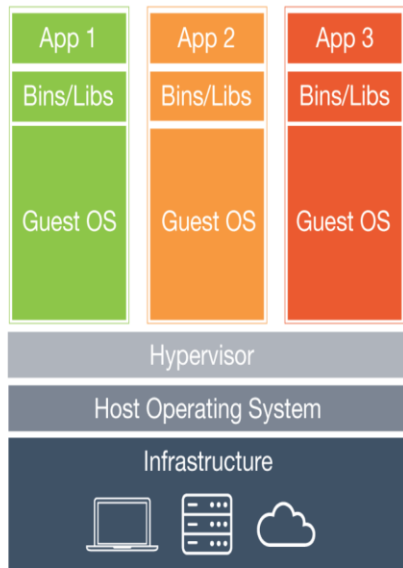


# ПРОГРАММА

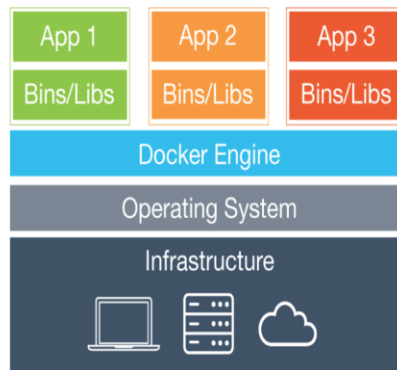
---

- Обзор cSRX
- Применение cSRX
- cSRX в среде Kubernetes и Contrail
- Практика и демонстрация: с чего начать?
  - Docker
  - Kubernetes & Contrail Networking
  - Где взять cSRX?

# КОНТЕЙНЕРЫ И ВИРТУАЛЬНЫЕ МАШИНЫ



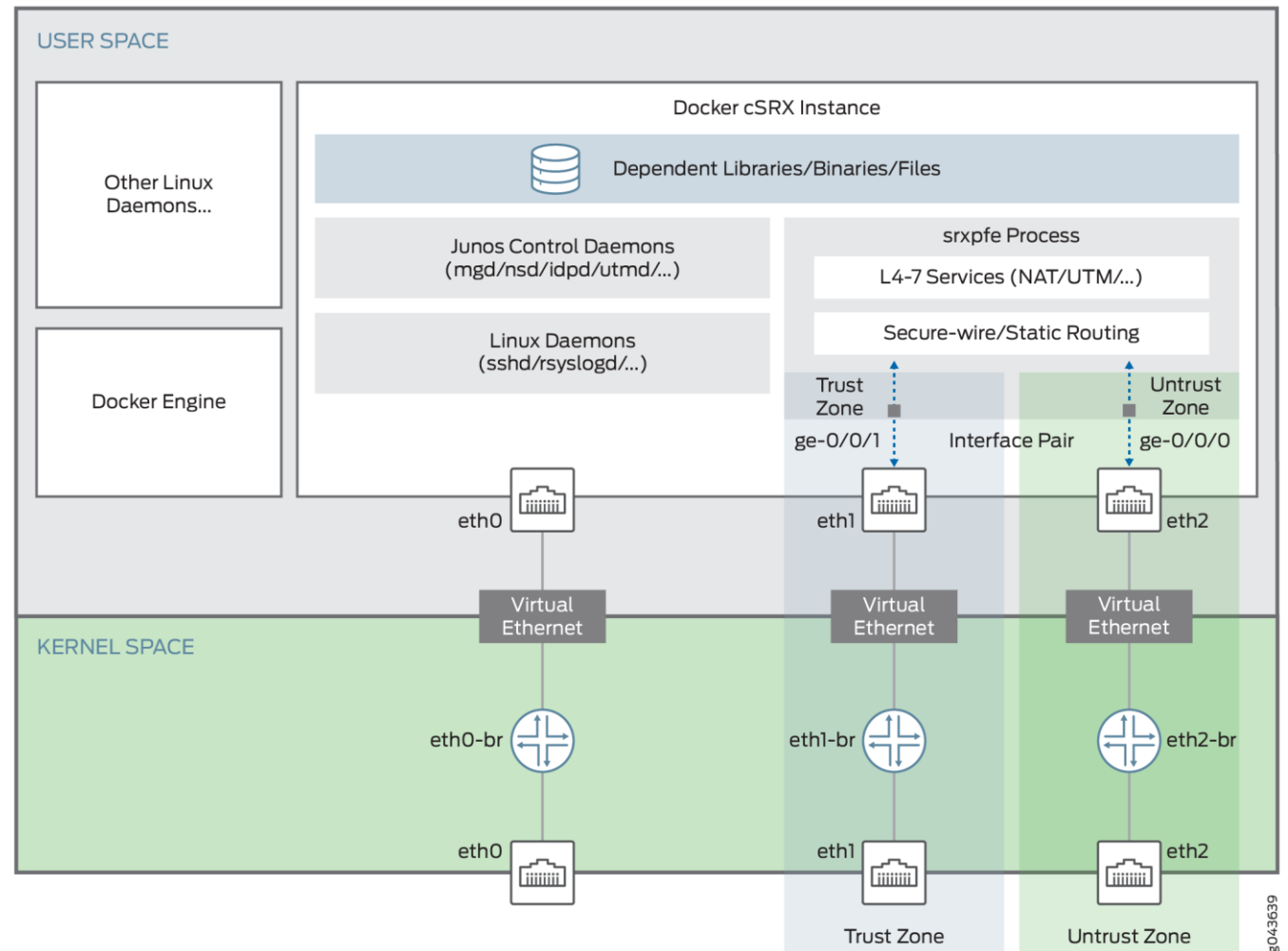
**В контейнере нет гостевой ОС**



	ВИРТУАЛЬНАЯ МАШИНА	КОНТЕЙНЕР
<b>ФИЛОСОФИЯ ДИЗАЙНА</b>	Аппаратная виртуализация	Виртуализация операционной системы
<b>УПРАВЛЕНИЕ РЕСУРСАМИ</b>	Резервируется статически. Минимальное совместное использование.	Нет статического резервирования. Разделяются между контейнерами в пределах хост-системы.
<b>ВРЕМЯ ЗАГРУЗКИ</b>	Занимает минуты	Менее секунды
<b>РАЗМЕР ОБРАЗА</b>	Несколько ГБайт	Менее ГБайта
<b>МАСШТАБИРУЕМОСТЬ</b>	Не более нескольких десятков экземпляров на хост	Может масштабироваться до тысяч на хост
<b>ЗАВИСИМОСТЬ ОТ ОС ХОСТА</b>	ОС хоста может отличаться от гостевой ОС	ОС хоста и ОС образа должна быть одинаковой

# CSRX – NGFW В DOCKER КОНТЕЙНЕРЕ

- Первый в индустрии контейнерный межсетевой экран
- Образ на основе Docker
- Интеграция с Kubernetes, OpenShift, OpenStack и Contrail SDN
- Режимы L2 Transparent (secure wire) и L3 Routing
- Функционал Stateful Firewall, NAT, IPS, UserFW, Application Security (AppFW, AppID, AppTrack), Content Security
- Унифицированные средства управления, возможность использования NetConf (19.2)



## АРХИТЕКТУРА КОНТЕЙНЕРА cSRX

# ПОДДЕРЖИВАЕМЫЙ ФУНКЦИОНАЛ

---

- Прозрачный режим L2 и L3 маршрутизации
- IPv4 и IPv6, Jumbo Frame
- Stateful Firewall с политиками безопасности
- Зоны безопасности
- Screen: защита от DoS и DDoS атак, защита с помощью SYN cookie
- Подавление атак brute force
- Обнаружение вторжений: IDP/IPS
- Детектирование приложений на основе сигнатур – AppSec: AppID, AppFW, UserFW
- UTM: EWF, Sophos AV
- NAT
- NetConf порт 830 (19.2)
- Удаленный Syslog
- Планы
  - Управление с помощью Junos Space Security Director
  - Увеличение количества интерфейсов до 16 (+1 интерфейс управления)

## МАСШТАБИРОВАНИЕ CSRX

ХАРАКТЕРИСТИКА	CSRX: Small 2 vCPU	CSRX: Medium 2 vCPU	CSRX: Large 2 vCPU
Объем памяти	256M	1G	4G
Количество одновременных сессий	8K	64K	512K

Масштабирование cSRX зависит от объема доступной памяти.  
vCPU является общим ресурсом для всех экземпляров, память – нет.

## ПРОИЗВОДИТЕЛЬНОСТЬ CSRX 18.1R1

ХАРАКТЕРИСТИКА	CSRX: Large 2 vCPU
Firewall throughput, large packet (1514B)	1.4 Gbps
Firewall throughput, IMIX	480 Mbps
Application visibility and control	750 Mbps
Intrusion prevention system (IPS) recommended signatures	500 Mbps



## СРАВНЕНИЕ vSRX И cSRX

	vSRX	cSRX
ПРИМЕНЕНИЕ	Интегрированная маршрутизация, безопасность, NAT, VPN, высокая производительность	Безопасность L4-L7 Security, низкое потребление ресурсов
ПРОЦЕССОРНЫЕ РЕСУРСЫ - vCPU	Минимум - 2 со статическим резервированием	Без резервирования Требуется 2 vCPU
РЕСУРСЫ ПАМЯТИ - RAM	Минимум 4GB	4GB для Large, 1 GB без IDP/IPS
NAT	Да	Да
ДИНАМИЧЕСКАЯ МАРШРУТИЗАЦИЯ	Да	Нет
IPSEC VPN	Да	Нет
ВРЕМЯ ЗАГРУЗКИ	~ минуты	<1 сек.
РАЗМЕР ОБРАЗА	~ ГБайты	~ МБайты
ТРЕБОВАНИЯ К ХОСТ-СИСТЕМЕ	Поддержка гипервизоров KVM / VMware / Hyper-V	Поддержка Docker контейнеров

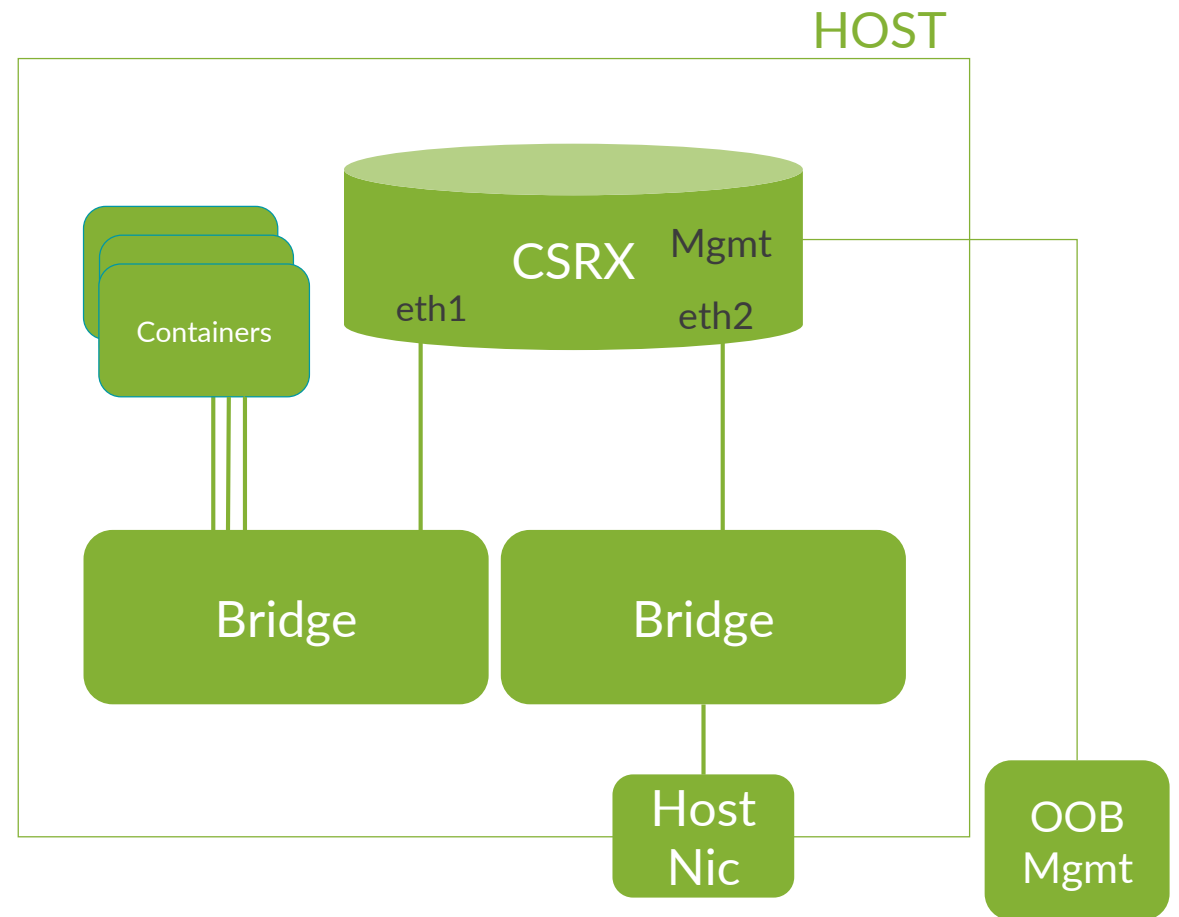
# ПРИМЕНЕНИЕ CSRX

---



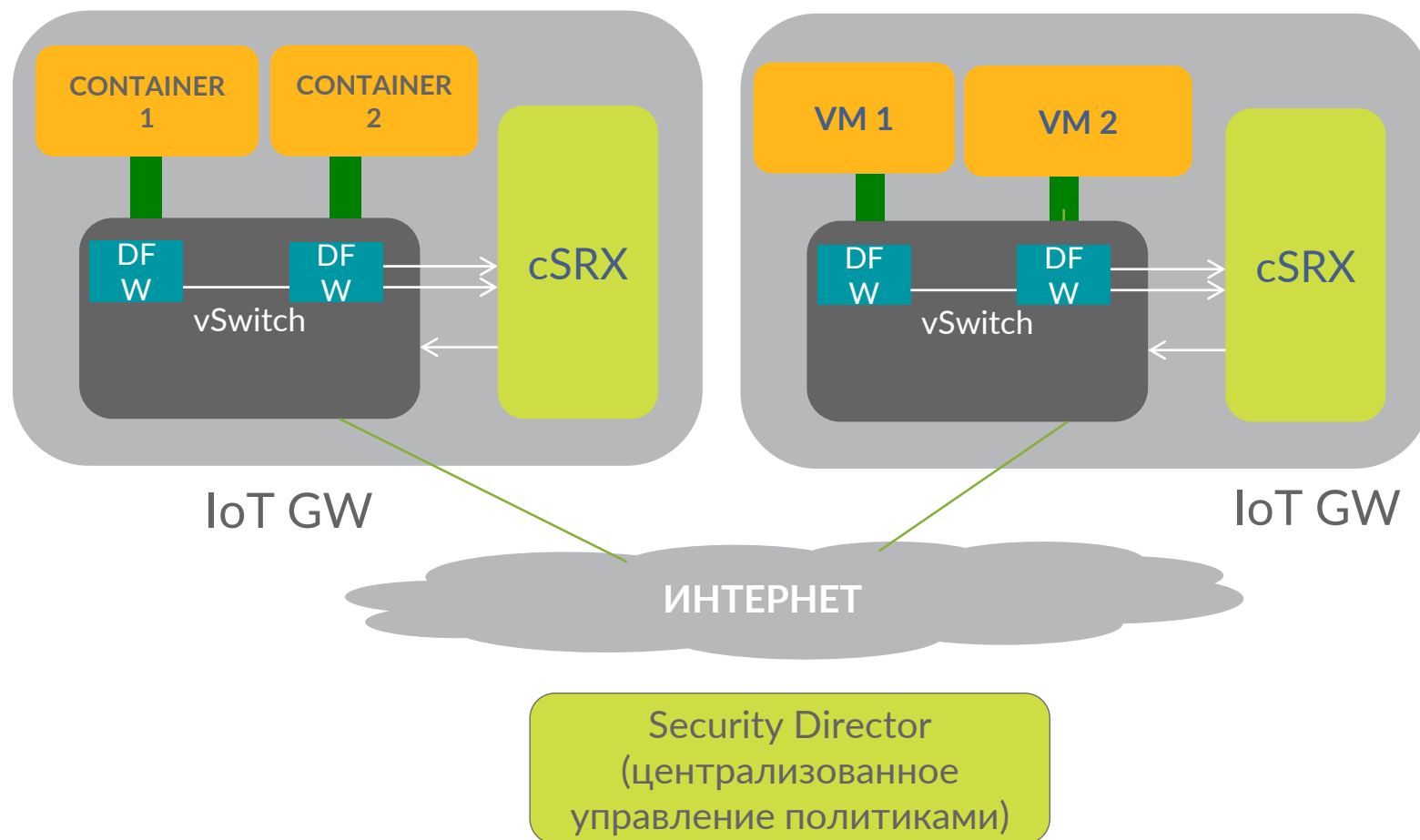
# ЗАЩИТА ПРИЛОЖЕНИЙ

- Инспектирование трафика «North-South»
- Политики безопасности L4-L7 с высокой гранулярностью
- Централизованный провижининг политик безопасности
- Внешняя оркестрация для провижининга cSRX в качестве шлюза трафика «North-South»
- Примеры: IoT шлюзы, приложения, требующие микросегментацию



# МИКРОСЕКМЕНТАЦИЯ

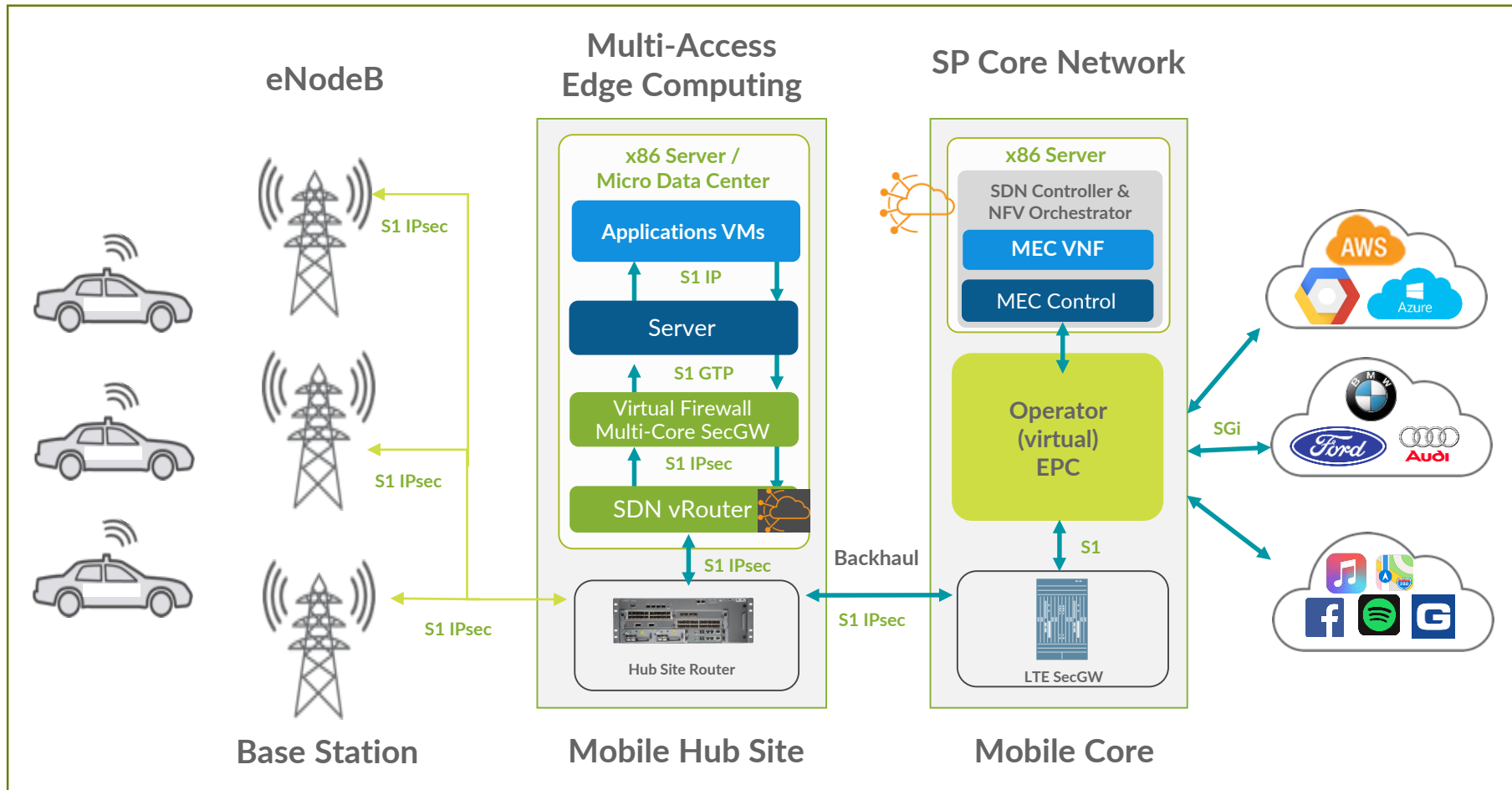
- vSwitch передает трафик «East-West» и «North-South» через cSRX
- Политики безопасности L4-L7 с высокой гранулярностью
- Централизованный провижининг политик безопасности
- Необходим внешний оркестратор для провижининга правил vSwitch перенаправления трафика
- Примеры: IoT шлюзы, приложения, требующие микросегментацию



# БЕЗОПАСНЫЙ ШЛЮЗ ДЛЯ АВТОМОБИЛЕЙ



# MOBILE EDGE CLOUD СВЯЗНОСТИ АВТОМОБИЛЕЙ

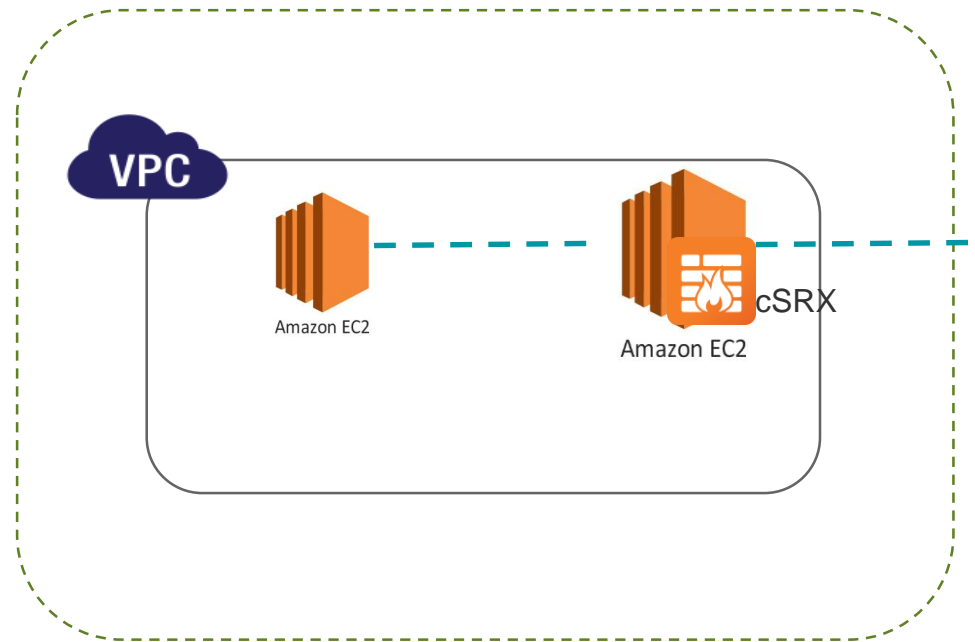


- Требование к низкой задержке
- Мобильная сотовая связь
- Развертывание Mobile Hub Site
- Использование MEC (Multi-Access Edge Computing)
- Виртуальный фаерволл на границе
- Политика безопасности: “Атаки с инфицированных мобильных устройств должны блокироваться на *Mobile Hub site*”
- Использование SDN контроллера и NFV оркестратора (MANO)

# ЛЕГКИЙ МЕЖСЕТЕВОЙ ЭКРАН ДЛЯ ПУБЛИЧНОГО ОБЛАКА

Small footprint firewall (T3.small) – межсетевой экран с базовым функционалом Stateful Firewall, NAT, Screen, для защиты VPC в публичных облаках

Планы – AWS и Azure



# CSRX В СРЕДЕ KUBERNETES И CONTRAIL

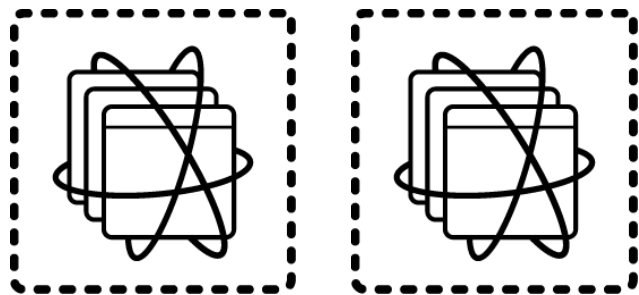
---

Kubernetes Multi-Interface POD

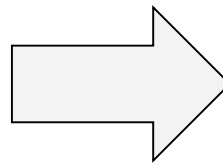




# ПОЧЕМУ КОНТЕЙНЕРЫ ТРЕБУЮТ ОРКЕСТРАЦИИ?



**КОНТЕЙНЕРНЫЕ ПРИЛОЖЕНИЯ**



## kubernetes

БЕЗОПАСНОЕ УПРАВЛЕНИЕ  
КОНТЕЙНЕРАМИ

ОСУЩЕСТВЛЯТЬ МАСШТАБИРОВАНИЕ  
КОНТЕЙНЕРНЫХ ПРИЛОЖЕНИЙ

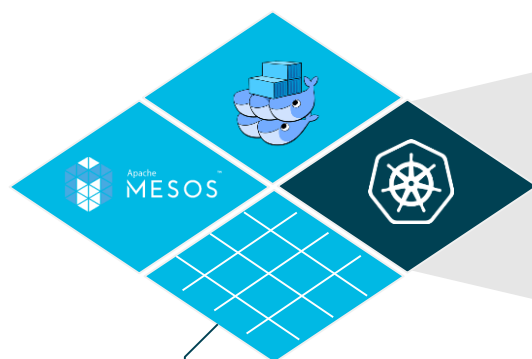
ИНТЕГРАЦИЯ В ЭКСПЛУАТАЦИЮ IT

ИСПОЛЬЗОВАНИЕ ГИБРИДНЫХ  
ОБЛАКОВ

# ЭВОЛЮЦИЯ KUBERNETES

## ДВА ГОДА НАЗАД

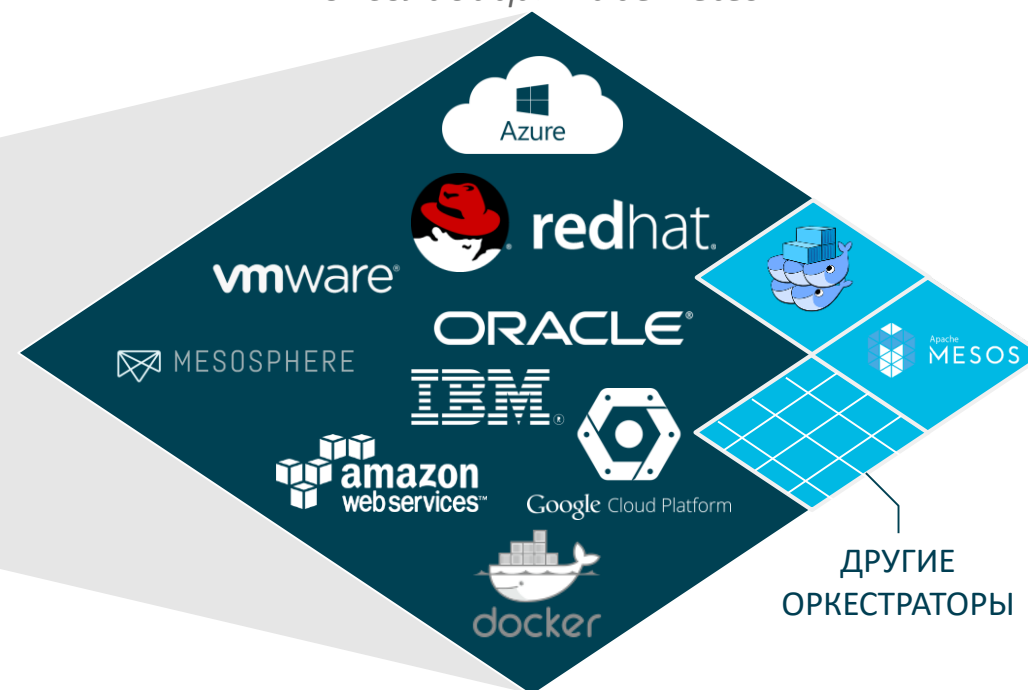
*Разрозненный ландшафт*



ДРУГИЕ ОРКЕСТРАТОРЫ  
(Cloud Foundry Diego, Nomad,  
Blox, etc.)

## СЕГОДНЯ

*Консолидация Kubernetes*

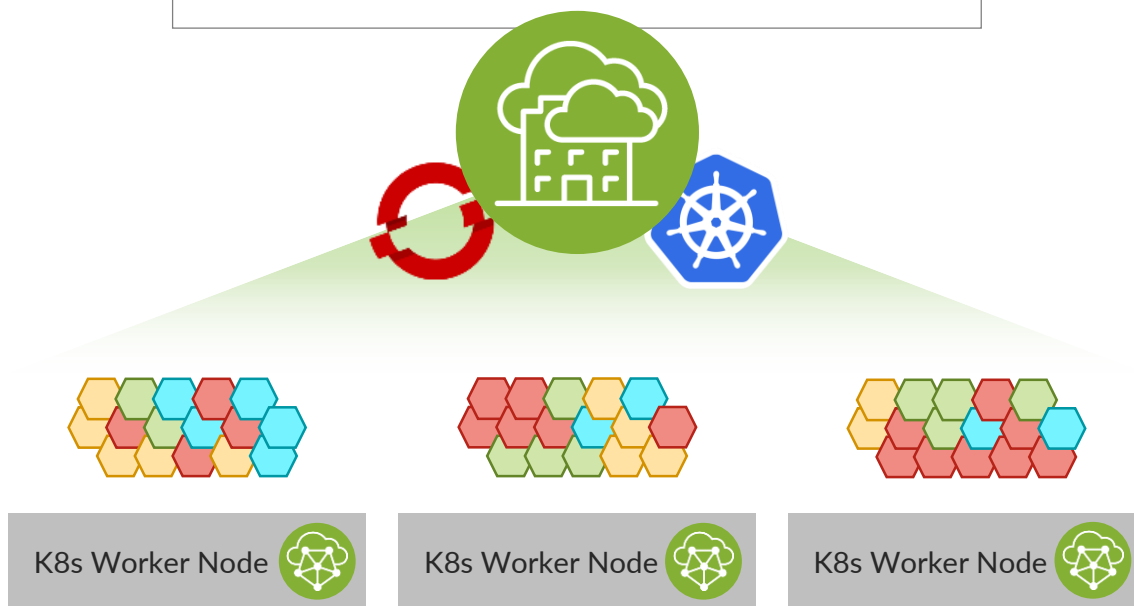


ДРУГИЕ  
ОРКЕСТРАТОРЫ

Red Hat достаточно рано сделал ставку на Kubernetes  
Теперь он стал доминирующей экосистемой оркестрации

# CONTRAIL CONTAINER NETWORKING

Contrail – это CNI, DNS и Load Balancer



Может работать на с любой underlay IP сетью,  
на любом расстоянии

Может работать в любой инфраструктуре

Высокопроизводительная безопасная сетевая инфраструктура  
для Kubernetes и OpenShift

## ПОДКЛЮЧЕНИЕ, ВИЗУАЛИЗАЦИЯ, БЕЗОПАСНОСТЬ

### Подключение

- Изоляция Namespaces с применением виртуальных сетей Contrail
- Управление связностью с использованием политик безопасности, политик маршрутизации и политик сетей
- Интегрированные сервисы, DNS и балансировщик входящего трафика обеспечивает портативность приложений
- Аппаратное ускорение сетевых балансировщиков нагрузки (MX/QFX)
- Мультикластерные сети
- Несколько интерфейсов на Pod и контейнерные сервисные цепочки

### Визуализация

- Визуализация всех политик и угроз, включая заблокированный трафик

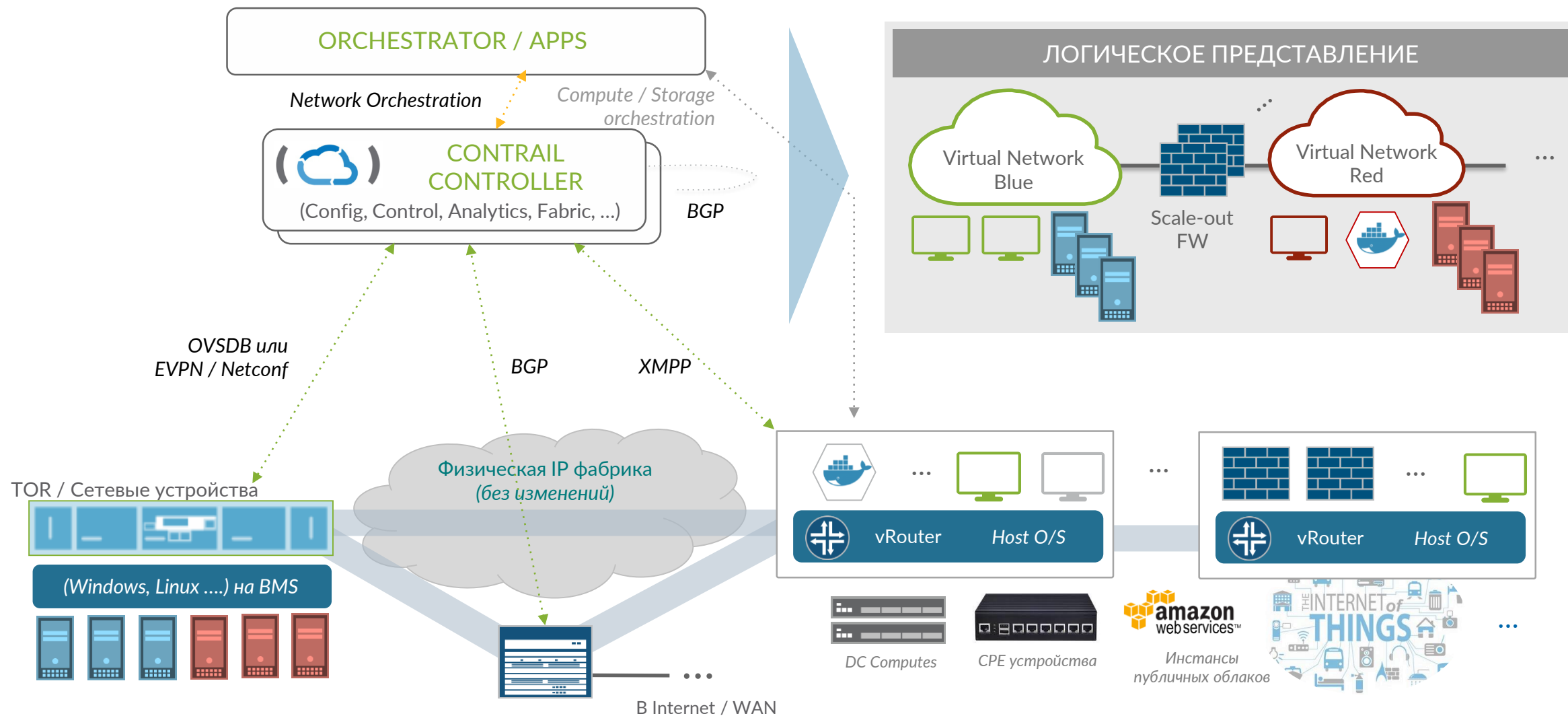
### Безопасность

- Безопасность микросегментов с использованием сетевых и contrail политик
- Contrail может ассоциировать политику с произвольными приложениями на основе объектных тегов K8s
- Автоматическое генерирование политик в режиме наблюдения и обучения (watch-and-learn)

# ЧТО ТАКОЕ CONTRAIL?

Централизованное  
определение политик

Распределенное  
применение политик



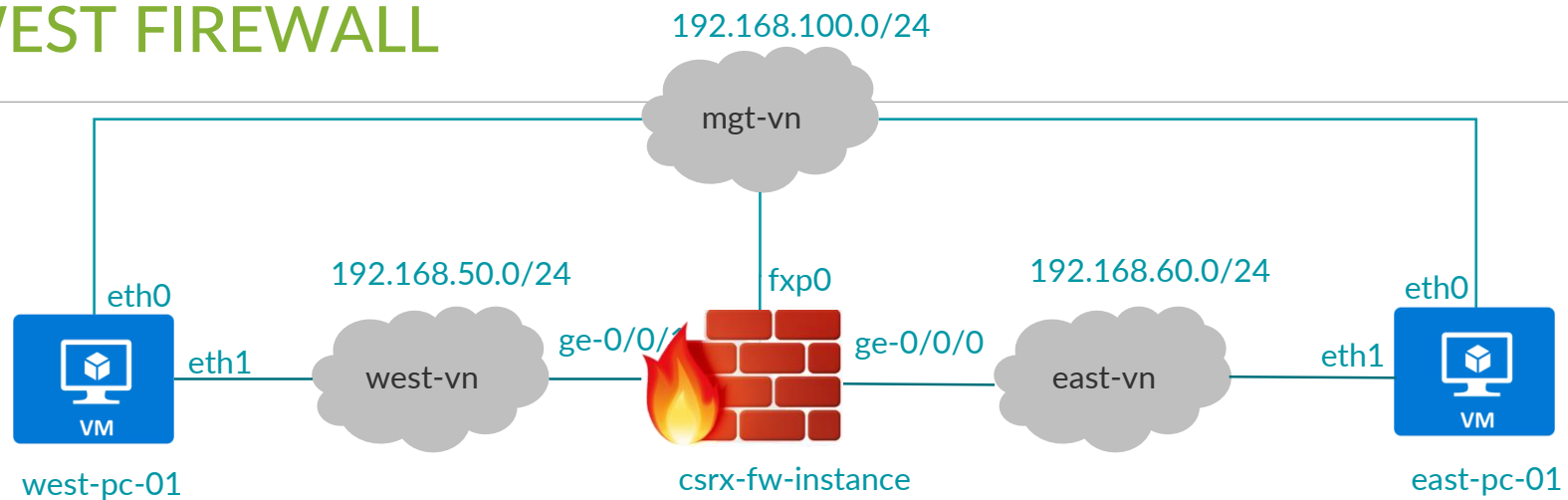
# ЧТО ТАКОЕ СЕРВИСНАЯ ЦЕПОЧКА NFV?

- Маршрутизация в data plane (реализуется на Contrail vRouter) обеспечивает связность между сетевыми функциями в заданной последовательности
- Применяется Filter-Based Forwarding и Route Reorigination
- Независимо от положения и форм-фактора сетевой функции
- Привязывается к виртуальным сетям

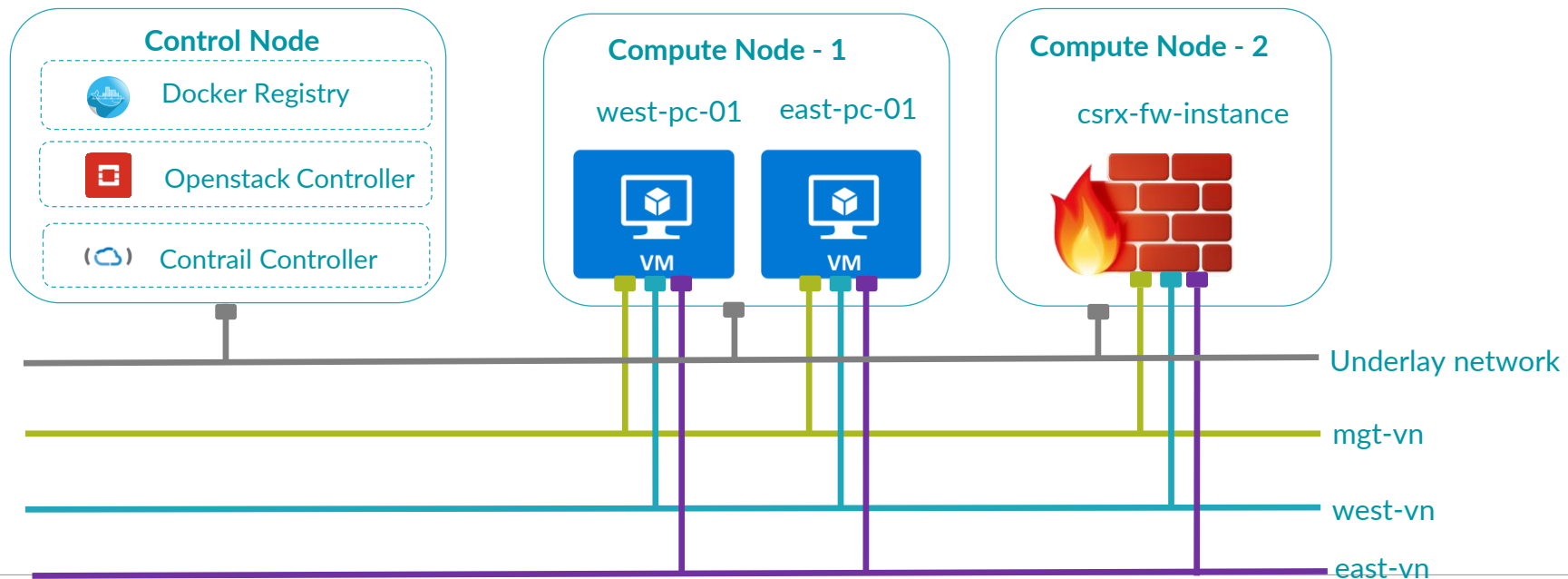


# ПРИМЕР: EAST-WEST FIREWALL

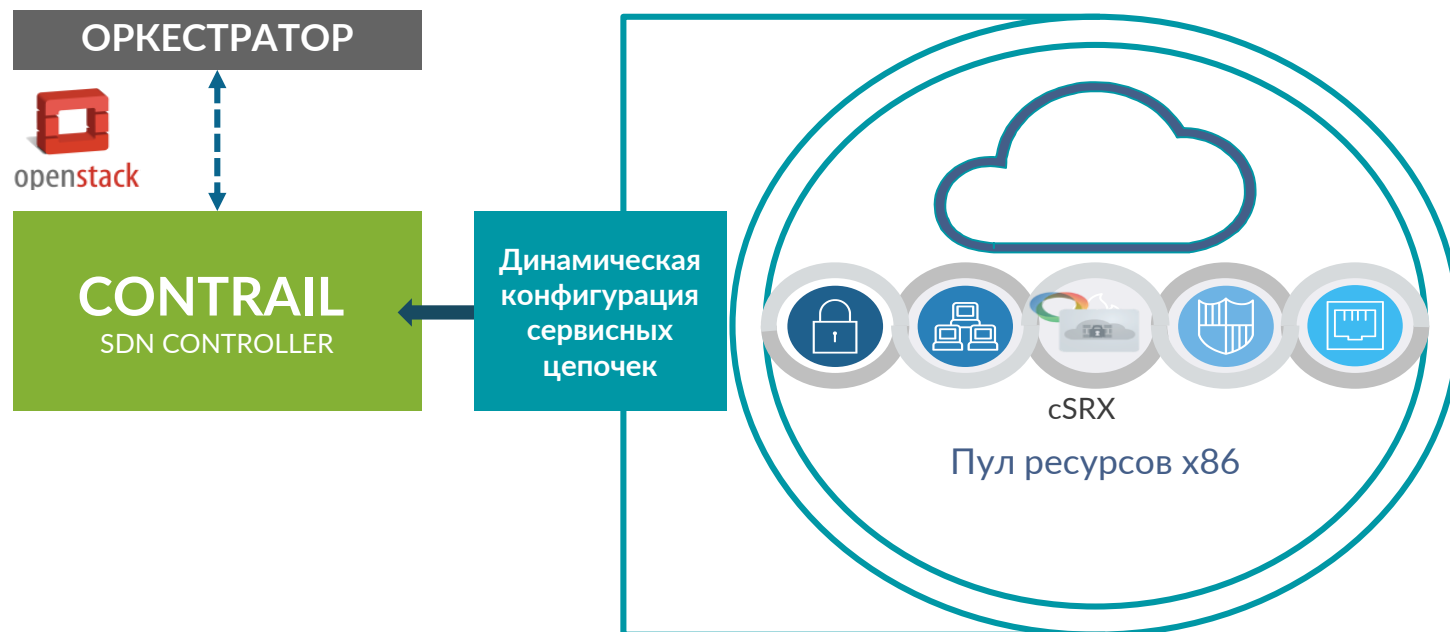
## ЛОГИЧЕСКАЯ СХЕМА



## ФИЗИЧЕСКАЯ СХЕМА



# CSRX: CONTRAIL SERVICE CHAINING



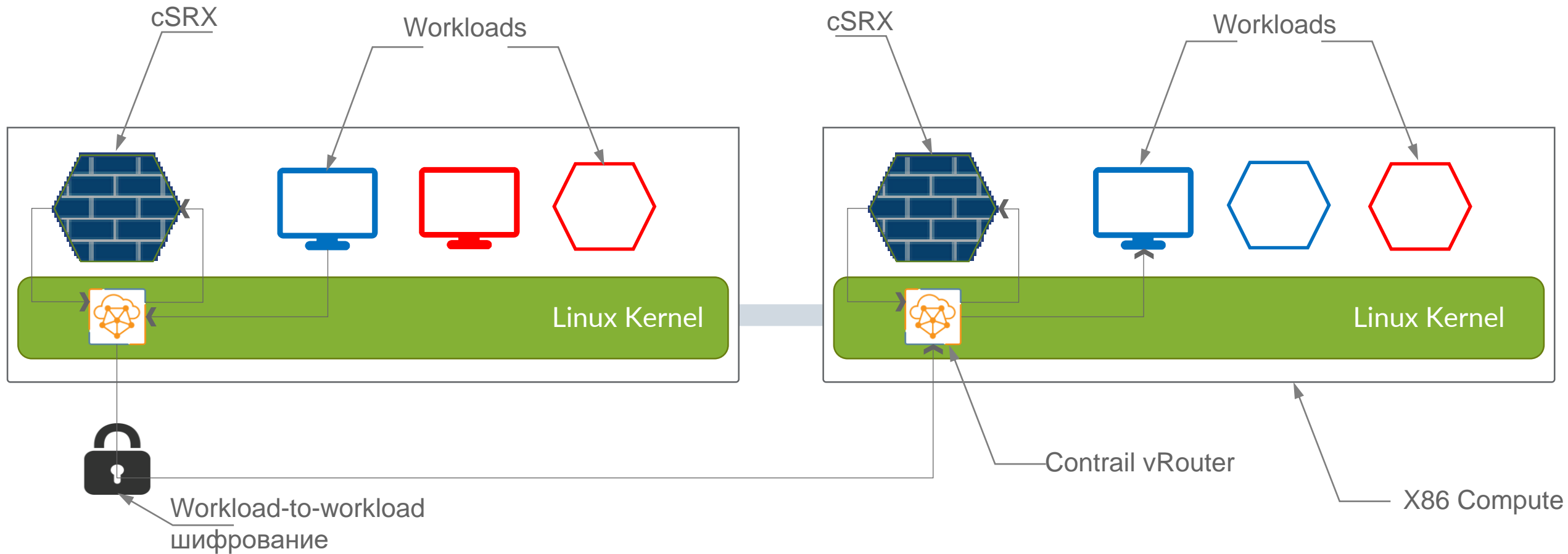
## КЛЮЧЕВЫЕ ОСОБЕННОСТИ

- Развертывание NGFW функционала в контейнере
- Поддержка App Security: AppFW, AppTrack, IPS
- Консистентное управление в Security Director

## ПРЕИМУЩЕСТВА

- Эластичность – cSRX отлично масштабируется благодаря небольшим требованиям и отсутствию статического резервирования ресурсов,
- Быстрота – скорость загрузки/рестарта до 1 секунды
- Консистентная оркестрация VM и контейнеров

# CONTRAIL SECURITY – HOST BASED FW HA OCHOBE CSRX



Контейнерный NGFW на каждом хосте,  
интегрированный в поток данных (out-of-box).





Практика: с чего начать?

# DOCKER [ШАГ 1]: ЗАГРУЗКА ОБРАЗА CSRX

- **Проверка доступных релизов в registry**

```
root@linux-csrx:~# curl -u JNPR-FieldUserXXX -X GET https://hub.juniper.net/v2/security/csrx/tags/list
Enter host password for user 'JNPR-FieldUserXXX':
{"name":"security/csrx","tags":["18.1R1.9","18.2R1.9","19.2R1.8"]}
```

- **Авторизация в Juniper registry**

```
docker login hub.juniper.net -u JNPR-FieldUserXXX -p xxx
```

- **Загрузка образа cSRX необходимой версии**

```
docker pull hub.juniper.net/security/csrx:19.2R1.8
```

- **Проверка списка локальных образов контейнеров**

```
root@linux-csrx:~# docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
ubuntu	latest	2ca708c1c9cc	4 days ago	64.2MB
hub.juniper.net/security/csrx	19.2R1.8	9075354bd501	3 months ago	481MB
hello-world	latest	fce289e99eb9	8 months ago	1.84kB

[https://www.juniper.net/documentation/en\\_US/csrx/topics/task/multi-task/security-csrx-linux-server-prep.html](https://www.juniper.net/documentation/en_US/csrx/topics/task/multi-task/security-csrx-linux-server-prep.html)

# DOCKER [ШАГ 2]: ЗАПУСК КОНТЕЙНЕРА И ПОДКЛЮЧЕНИЕ СЕТИ

---

- **Создание сетей для cSRX (linux bridge)**

```
docker network create --driver bridge mgmt_bridge
docker network create --driver bridge left_bridge
docker network create --driver bridge right_bridge
```

- **Запуск контейнера с опциями (сеть управления, root пароль, режим L3, размер)**

```
docker run -d --privileged --network=mgmt_bridge -e CSRX_ROOT_PASSWORD=jun123 -e CSRX_FORWARD_MODE="routing" -e CSRX_SIZE="small" --name=csr01 hub.juniper.net/security/csr:19.2R1.8
```

- **Подключение дополнительных интерфейсов**

```
docker network connect left_bridge csr01
docker network connect right_bridge csr01
```

- **Подключение к локальной консоли**

```
docker exec -it csr01 cli
```

- **Включаем поддержку терминала VT100 (опционально, для работы кнопок вверх/вниз )**

```
root@3e9cc7105585> set cli terminal vt100
```

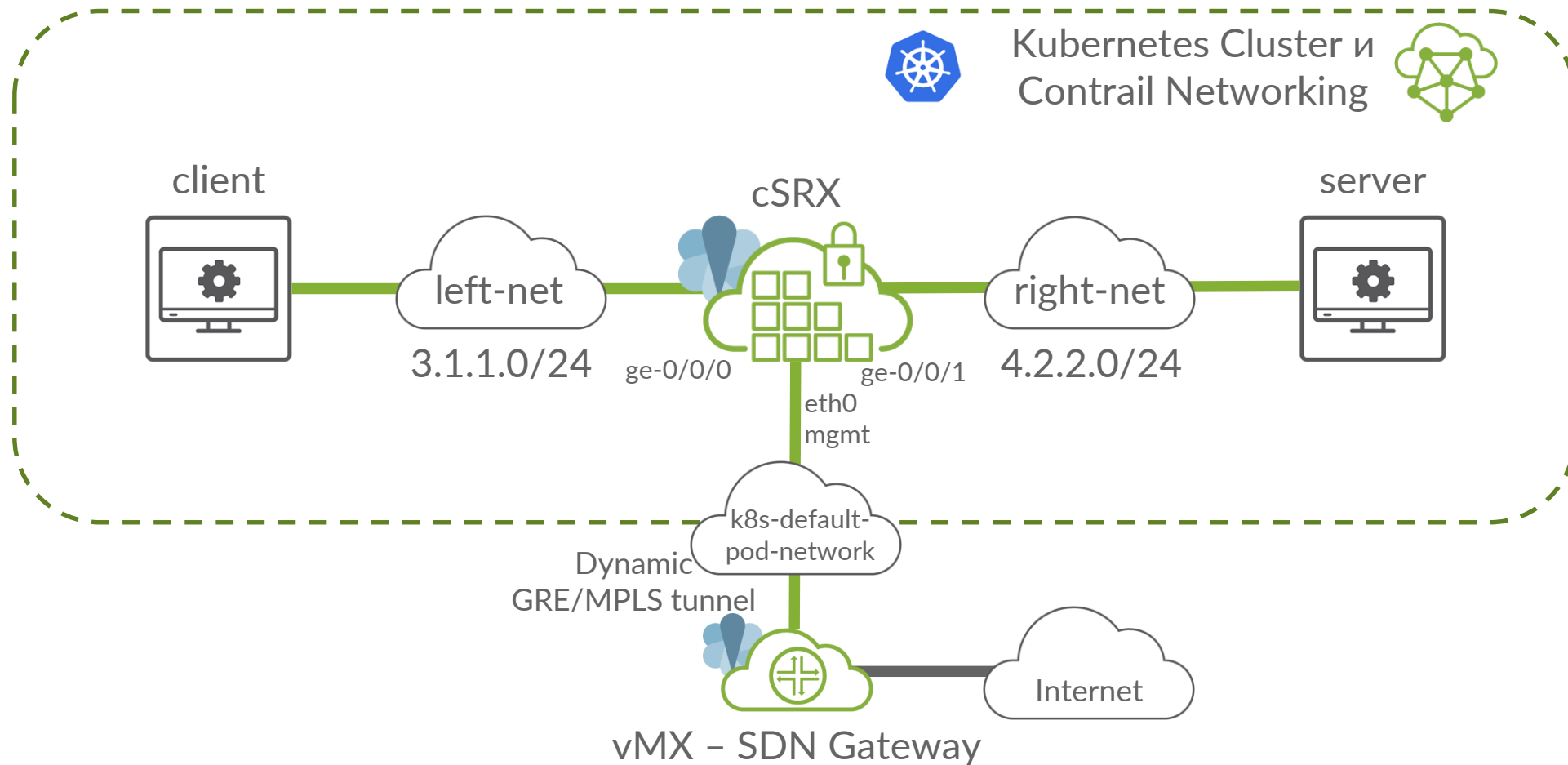


## Демонстрация cSRX в Kubernetes + Contrail

# СХЕМА СТЕНДА



# СХЕМА СТЕНДА



# Шаг 1. Интерфейс: Contrail и Kubernetes Dashboard

### Kubernetes Dashboard

- Kubeconfig**  
Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the [Configure Access to Multiple Clusters](#) section.
- Token**  
Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the [Authentication](#) section.

Choose kubeconfig file ⋮

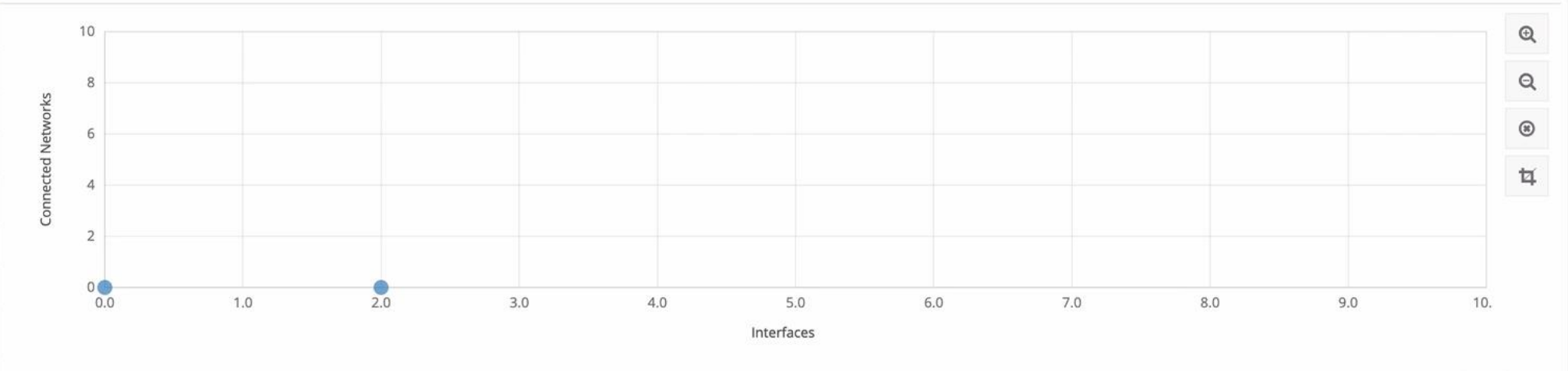
**SIGN IN**      SKIP



## Шаг 2. Создание сетей в Kubernetes и проверка в Contrail

- Monitor ←
- 🖥 Infrastructure
- 🛡 Security
- 👤 Networking
- 📊 Dashboard
- 📁 Projects
- 🌐 Networks
- 📁 Instances
- 🔌 Interfaces
- 🔧 Debug



Networks Summary 📄 🔍 ↻

Network	Instances	Interfaces	Traffic In/Out (Last 1 Hr)	Throughput In/Out
▶ default-domain:k8s-default:k8s-default-pod-network	2	2	280 KB / 2 MB	0 bps / 0 bps
▶ default-domain:k8s-default:k8s-default-service-network	0	0	- / -	0 bps / 0 bps

Total: 2 records | 8 Records ▾ ⏪ ⏩ Page 1 ▾ of 1 ⏪ ⏩

# K8S: КОНФИГУРАЦИЯ СЕТЕЙ (YAML)

```
---
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    opencontrail.org/cidr: "3.1.1.0/24"
  name: left-net
spec:
  config: '{ "cniVersion": "0.3.0", "type": "contrail-k8s-cni" }'
---
apiVersion: k8s.cni.cncf.io/v1
kind: NetworkAttachmentDefinition
metadata:
  annotations:
    opencontrail.org/cidr: "4.2.2.0/24"
  name: right-net
spec:
  config: '{ "cniVersion": "0.3.0", "type": "contrail-k8s-cni" }'
```

## CLI K8s:

```
kubectl apply -f config.yaml
```

```
kubectl get network-attachment-definition
```

```
kubectl describe network-attachment-definition left-net
```

## Шаг 3. Создание linux контейнеров «client» и «server»

- Configure
- Infrastructure
- Security
- Tags
- Physical Devices
- Networking**
  - Load Balancing
  - Networks**
  - Ports
  - Policies
  - Security Groups
  - Routers
  - IP Address Management
  - Floating IP Pools
  - Floating IPs
  - Routing
  - QoS
  - SLO

Networks + 🗑️ 📄 🔍 ↻

Network	Subnets	Tags	Attached Policies	Shared	Admin State
<input type="checkbox"/> k8s-default-pod-networ k	k8s-pod-ipam	application=k8s	-	Disabled	Up
<input type="checkbox"/> k8s-default-service-netw ork	k8s-service-ipam	application=k8s	-	Disabled	Up
<input type="checkbox"/> k8s-left-net-pod-networ k	3.1.1.0/24	-	-	Disabled	Up
<input type="checkbox"/> k8s-right-net-pod-netwo rk	4.2.2.0/24	-	-	Disabled	Up

Total: 4 records | 50 Records Page 1 of 1

# K8S: КОНФИГУРАЦИЯ LINUX КОНТЕЙНЕРА CLIENT (YAML)

---

```
---
apiVersion: v1
kind: Pod
metadata:
  name: client
  annotations: {
    "opencontrail.org/network" : '{"domain":"default-domain", "project": "k8s-default",
"name":"k8s-left-net-pod-network"}'
  }

spec:
  containers:
  - image: registry.jnpr.lab/ubuntu-dm
    name: c1
    securityContext:
      privileged: true
```

# K8S: КОНФИГУРАЦИЯ LINUX КОНТЕЙНЕРА SERVER (YAML)

---

```
---
apiVersion: v1
kind: Pod
metadata:
  name: server
  annotations: {
    "opencontrail.org/network" : '{"domain":"default-domain", "project": "k8s-default",
"name":"k8s-right-net-pod-network"}'
  }

spec:
  containers:
  - image: registry.jnpr.lab/ubuntu-dm
    name: s1
    securityContext:
      privileged: true
```

## Шаг 4. Создание cSRX контейнера




☰ Overview

- Cluster
  - Namespaces
  - Nodes
  - Persistent Volumes
  - Roles
  - Storage Classes
- Namespace
  - default ▾
- Overview
- Workloads
  - Cron Jobs
  - Daemon Sets
  - Deployments
  - Jobs
  - Pods
  - Replica Sets
  - Replication Controllers
  - Stateful Sets

### Workloads

#### Workloads Statuses



100.00%

Pods

#### Pods

Name	Node	Status	Restarts	Age
✓ client	k8-node2.jnpr.lab	Running	0	a minute
✓ server	k8-node2.jnpr.lab	Running	0	a minute

#### Discovery and Load Balancing

#### Services

# K8S: КОНФИГУРАЦИЯ CSRX КОНТЕЙНЕРА (YAML)

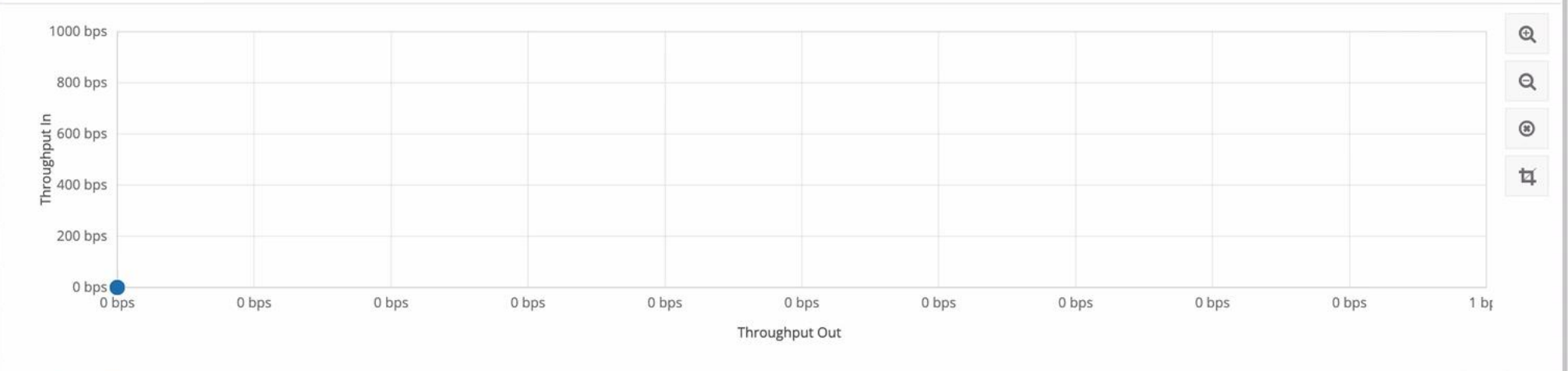
---

```
---
apiVersion: v1
kind: Pod
metadata:
  name: csrx-pod
  annotations:
    k8s.v1.cni.cncf.io/networks: '[
      { "name": "left-net" },
      { "name": "right-net" }
    ]'
spec:
  containers:
  - image: hub.juniper.net/security/csrx:19.2R1.8
    imagePullPolicy: IfNotPresent
    name: csrx
    stdin: true
    tty: false
    env:
    - name: CSRX_FORWARD_MODE
      value: "routing"
    securityContext:
      privileged: true
    restartPolicy: Always
    imagePullSecrets:
    - name: regcred
```

## Шаг 5. Настройка/конфигурирование cSRX

📊 ⚙️ ⚙️ 🔍

- Monitor
- Infrastructure
- Security
- Networking
- Dashboard
- Projects
- Networks
- Instances
- Interfaces
- Debug



Interfaces Summary

Name	UUID	IP Address	Instance Name	Floating IPs ( Agg Stats In/Out)	Traffic In/Out (Last 1 H...	Throughput In/Out	Status
▶ default-domai...	423a1a44-f02...	3.1.1.252	client_422a62e8-f026-11e9-b8f7-5254...	-	- / -	- / -	● Active
▶ default-domai...	4f6e18e6-f026...	10.47.255.252	csrx-pod_4f5d7b38-f026-11e9-b8f7-52...	-	- / -	- / -	● Active
▶ default-domai...	4f8eeb3e-f026...	3.1.1.251	csrx-pod_4f5d7b38-f026-11e9-b8f7-52...	-	- / -	- / -	● Active
▶ default-domai...	4fadf7a4-f026...	4.2.2.251	csrx-pod_4f5d7b38-f026-11e9-b8f7-52...	-	- / -	- / -	● Active
▶ default-domai...	4284c1ac-f02...	4.2.2.252	server_422b9d43-f026-11e9-b8f7-5254...	-	- / -	- / -	● Active
▶ default-domai...	a2d49226-effe...	10.47.255.244	coredns-576cbf47c7-p647g_a27a9855-...	-	- / -	- / -	● Active
▶ default-domai...	a31cee22-effe...	10.47.255.243	coredns-576cbf47c7-s9tt6_a27d7527-e...	-	- / -	- / -	● Active
▶ default-domai...				-	- / -	- / -	● Inactive

## CSRX: БАЗОВАЯ КОНФИГУРАЦИЯ (JUNOS)

---

```
set interfaces ge-0/0/0 unit 0 family inet address 3.1.1.251/24
set interfaces ge-0/0/1 unit 0 family inet address 4.2.2.251/24

set security zones security-zone trust host-inbound-traffic system-services all
set security zones security-zone trust host-inbound-traffic protocols all
set security zones security-zone trust interfaces ge-0/0/0.0

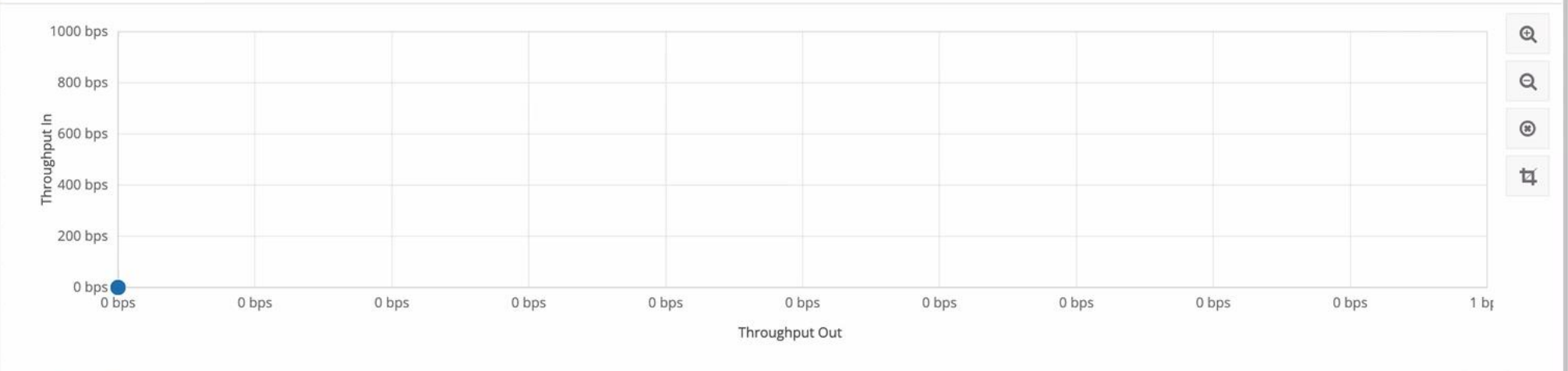
set security zones security-zone untrust host-inbound-traffic system-services all
set security zones security-zone untrust host-inbound-traffic protocols all
set security zones security-zone untrust interfaces ge-0/0/1.0

set security policies default-policy permit-all
```

## Шаг 6. Настройка сервисной цепочки в Contrail

📊 ⚙️ ⚙️ 🔍

- Monitor
- Infrastructure
- Security
- Networking
- Dashboard
- Projects
- Networks
- Instances
- Interfaces
- Debug



### Interfaces Summary

Name	UUID	IP Address	Instance Name	Floating IPs ( Agg Stats In/Out)	Traffic In/Out (Last 1 H...	Throughput In/Out	Status
▶ default-domai...	423a1a44-f02...	3.1.1.252	client_422a62e8-f026-11e9-b8f7-5254...	-	- / -	- / -	● Active
▶ default-domai...	4f6e18e6-f026...	10.47.255.252	csrx-pod_4f5d7b38-f026-11e9-b8f7-52...	-	- / -	- / -	● Active
▶ default-domai...	4f8eeb3e-f026...	3.1.1.251	csrx-pod_4f5d7b38-f026-11e9-b8f7-52...	-	- / -	- / -	● Active
▶ default-domai...	4fadf7a4-f026...	4.2.2.251	csrx-pod_4f5d7b38-f026-11e9-b8f7-52...	-	- / -	- / -	● Active
▶ default-domai...	4284c1ac-f02...	4.2.2.252	server_422b9d43-f026-11e9-b8f7-5254...	-	- / -	- / -	● Active
▶ default-domai...	a2d49226-effe...	10.47.255.244	coredns-576cbf47c7-p647g_a27a9855-...	-	- / -	- / -	● Active
▶ default-domai...	a31cee22-effe...	10.47.255.243	coredns-576cbf47c7-s9tt6_a27d7527-e...	-	- / -	- / -	● Active
▶ default-domai...				-	- / -	- / -	● Inactive

## Шаг 7. Проверка функционала Screen (защита от syn-flood, scan, и других атак)



☰ **Shell**

- Cluster
  - Namespaces
  - Nodes
  - Persistent Volumes
  - Roles
  - Storage Classes

---

- Namespace
  - default ▾

---

- Overview
- Workloads
  - Cron Jobs
  - Daemon Sets
  - Deployments
  - Jobs
  - Pods
  - Replica Sets
  - Replication Controllers
  - Stateful Sets

```

Shell in csrx in csrx-pod
root@csrx-pod>
root@csrx-pod>
root@csrx-pod> show security flow session | refresh 2
---(refreshed at 2019-10-16 15:18:20 UTC)---
Total sessions: 0
---(refreshed at 2019-10-16 15:18:22 UTC)---
Total sessions: 0
---(refreshed at 2019-10-16 15:18:24 UTC)---
Total sessions: 0
---(refreshed at 2019-10-16 15:18:26 UTC)---
Total sessions: 0
---(refreshed at 2019-10-16 15:18:28 UTC)---
Total sessions: 0
---(refreshed at 2019-10-16 15:18:30 UTC)---
Total sessions: 0
---(refreshed at 2019-10-16 15:18:32 UTC)---
Session ID: 17, Policy name: default-policy-logical-system-00/2, Timeout: 2, Valid
  In: 3.1.1.252/33258 --> 4.2.2.252/80;tcp, Conn Tag: 0x0, If: ge-0/0/0.0, Pkts: 6, Bytes: 393,
  Out: 4.2.2.252/80 --> 3.1.1.252/33258;tcp, Conn Tag: 0x0, If: ge-0/0/1.0, Pkts: 4, Bytes: 1075,
Total sessions: 1
---(refreshed at 2019-10-16 15:18:34 UTC)---
Total sessions: 0
---(refreshed at 2019-10-16 15:18:36 UTC)---
Total sessions: 0
---(*more 100%)---[abort]

root@csrx-pod>

```

## Шаг 8. Проверка функционала IDP/IPS

☰ Shell

- Cluster
  - Namespaces
  - Nodes
  - Persistent Volumes
  - Roles
  - Storage Classes
- Namespace
  - default ▾
- Overview
- Workloads
  - Cron Jobs
  - Daemon Sets
  - Deployments
  - Jobs
  - Pods
  - Replica Sets
  - Replication Controllers
  - Stateful Sets

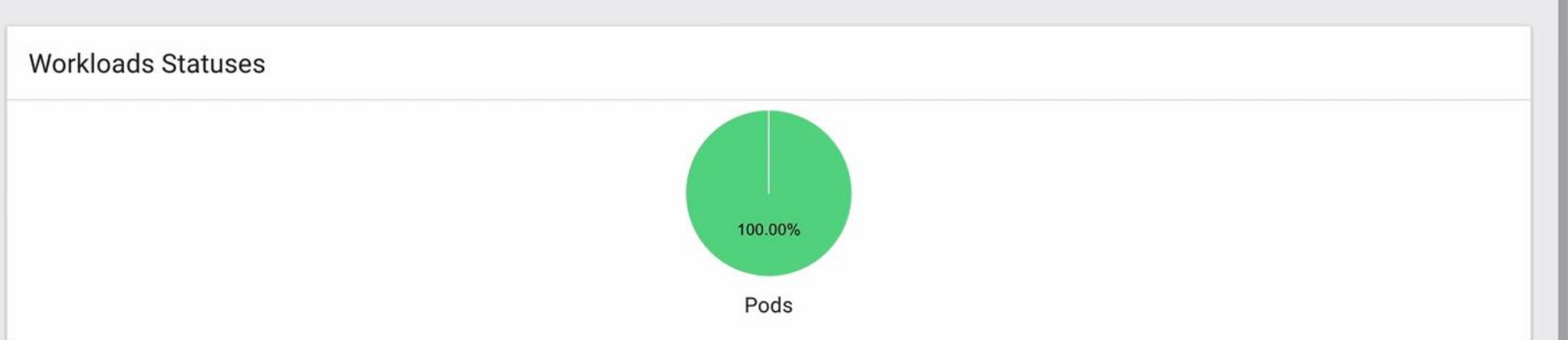
```
Shell in c1 in client
round-trip min/avg/max = 0.4/20.6/1035.8 ms
root@client:/#
root@client:/# curl http://4.2.2.252
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
<p><em>Thank you for using nginx.</em></p>
</body>
</html>
root@client:/#
```

# Шаг 9. Проверка функционала Application Security

☰ Overview

- Cluster
  - Namespaces
  - Nodes
  - Persistent Volumes
  - Roles
  - Storage Classes
- Namespace
  - default ▾
- Overview
- Workloads
  - Cron Jobs
  - Daemon Sets
  - Deployments
  - Jobs
  - Pods
  - Replica Sets
  - Replication Controllers
  - Stateful Sets

### Workloads



#### Pods

Name	Node	Status	Restarts	Age	
✓ csrx-pod	k8-node2.jnpr.lab	Running	0	7 minutes	☰ ⋮
✓ client	k8-node2.jnpr.lab	Running	0	an hour	☰ ⋮
✓ server	k8-node2.jnpr.lab	Running	0	an hour	☰ ⋮

### Discovery and Load Balancing



Где взять cSRX?

# ЛИЦЕНЗИРОВАНИЕ И ДОСТУПНОСТЬ

- Полностью доступен для заказа и использования
- Возможность проведения демо и PoC --> через Juniper SE
- Доступен в репозитории [hub.juniper.net](http://hub.juniper.net) (в [juniper.net/support/downloads/](http://juniper.net/support/downloads/) cSRX'а нет)
- Лицензия – подписка на 1 и 3 года, по полосе пропускания

SKU В ПРАЙС-ЛИСТЕ	ОПИСАНИЕ
CSRX-1G-ADV01-1	Upto 1G Throughput, 1 year Subscription License for cSRX including stateful firewall, IPS, Appsecure, Anti-virus, Web-filtering, & Content Filtering. Support included
CSRX-1G-ADV01-3	Upto 1G Throughput, 3 year Subscription License for cSRX including stateful firewall, IPS, Appsecure, Anti-virus, Web-filtering, & Content Filtering. Support included



ВОПРОСЫ?

---

JUNIPER  
NETWORKS | Engineering  
Simplicity

[dkaryakin@juniper.net](mailto:dkaryakin@juniper.net)

 [dkaryakin](https://t.me/dkaryakin)