Интеграция Cisco ACI с контейнерными платформами

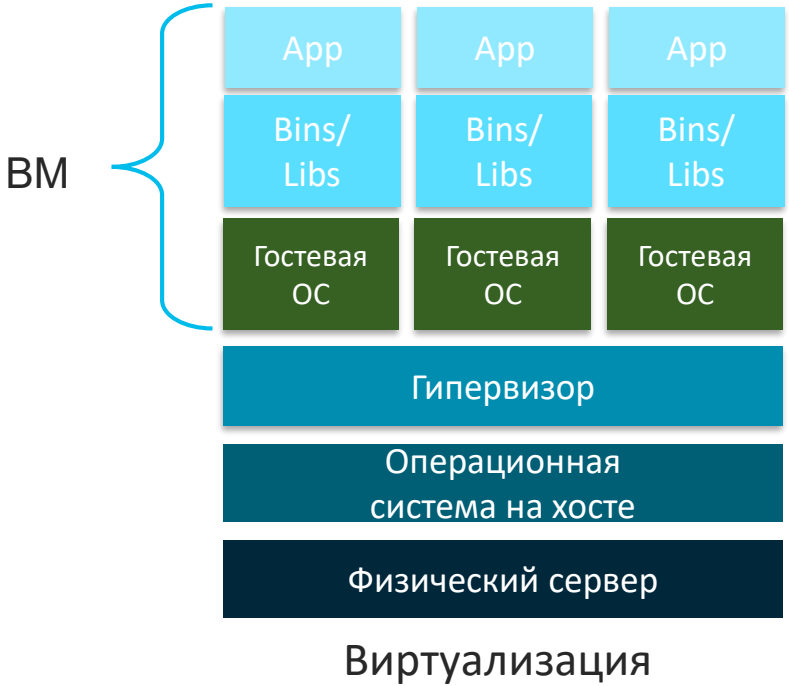
Ключевые возможности, преимущества, демонстрация работы

Александр Скороходов
Архитектор Cisco по технологиям
Октябрь 2020

Что такое контейнер?

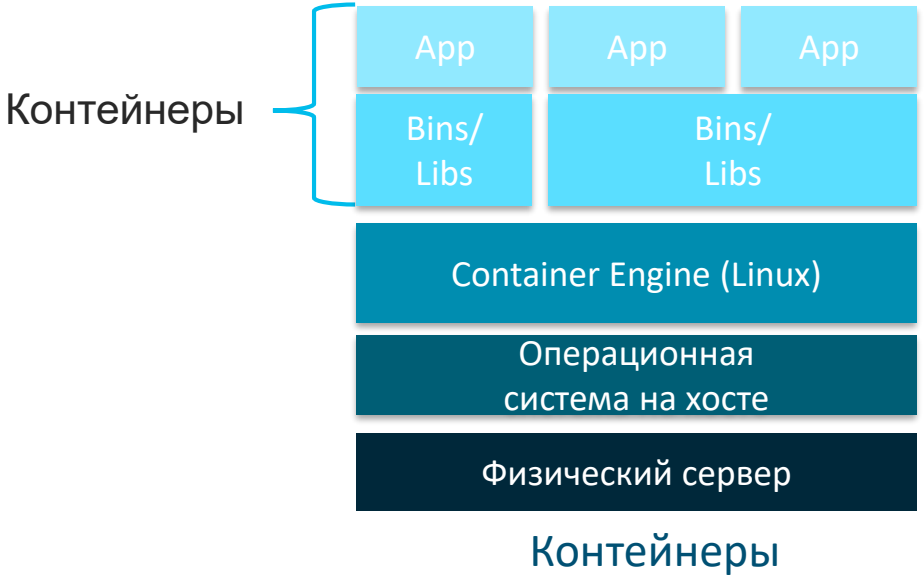
- Бинарный код, упакованный со всем окружением и зависимостями, запускаемый в изолированном пространстве имен (namespace) для которого контролируется доступ к ресурсам операционной системы (cgroups)
- При запуске нескольких контейнеров на одном хосте каждый из них получает (в общем случае) свою операционную среду с файловой системой, сетью, пространством процессов и подсистемой ввода-вывода

Сравнение виртуализации и контейнеров



Виртуализация

Контейнеры изолированы, но разделяют ядро ОС и (где возможно) файлы/библиотеки



Контейнеры

Эффективная платформа для микросервисов

Контейнеры - идеальная среда для запуска микросервисов

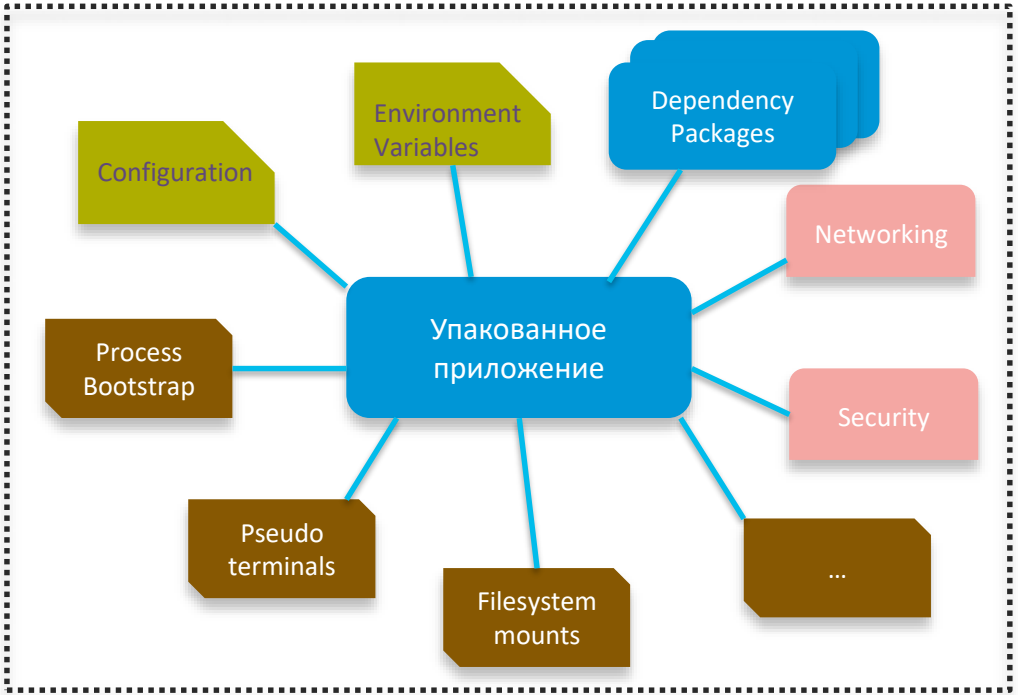
- Микросервисы : слабо-связанные (loosely coupled) компоненты приложения, взаимодействующие между собой при помощи API, которые запускаются в разных окружениях

Контейнеры отвечают требованиям приложений нового поколения

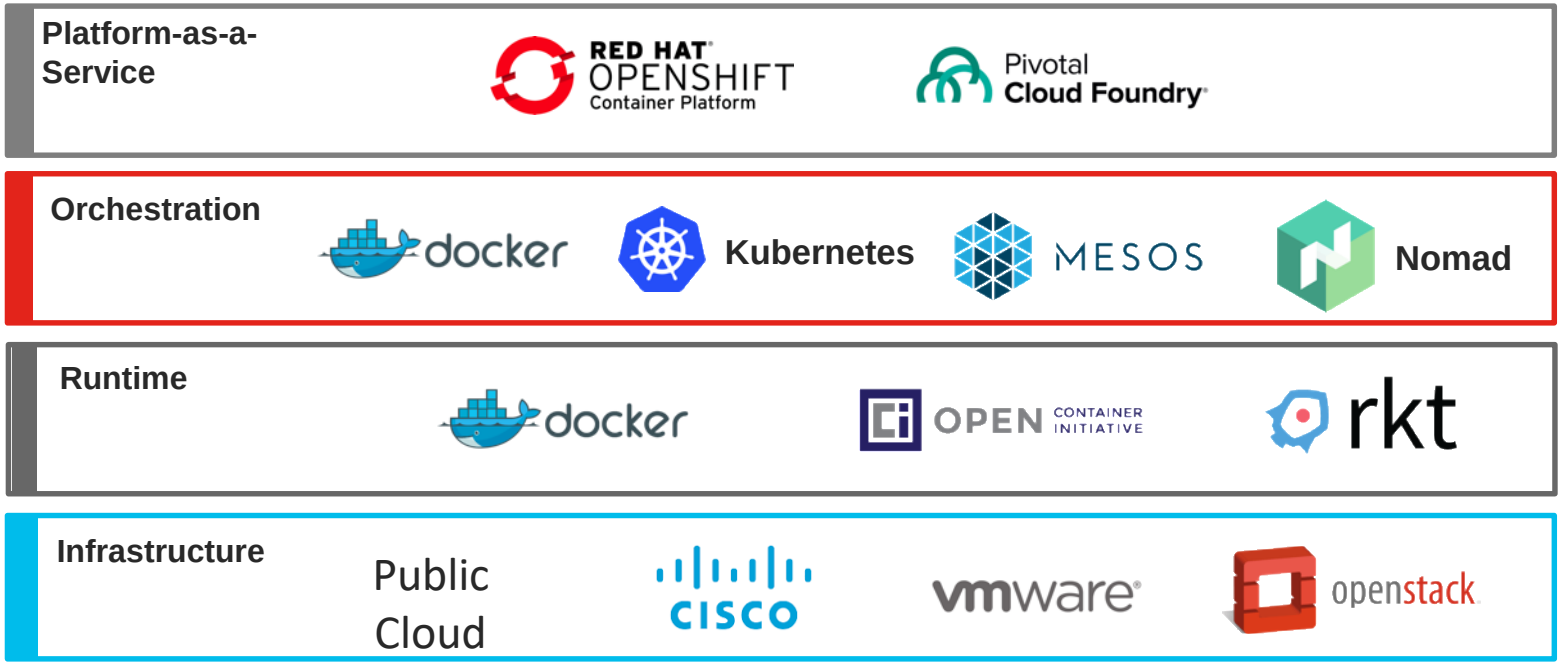
- Высокая плотность
- Скорость
- Низкие затраты на настройку и управление
- Переносимость между средами

Контейнеры как механизм «упаковки» приложений: Подготовить однажды – внедрить где угодно

- Контейнеры сочетают контент (“RPM”) и контекст (среда, в которой RPM должен быть развёрнут)
- Решает проблему управления зависимостями
- Идеальный механизм для переноса прикладных компонент по конвейеру разработки-тестирования-внедрения



Контейнерная экосистема

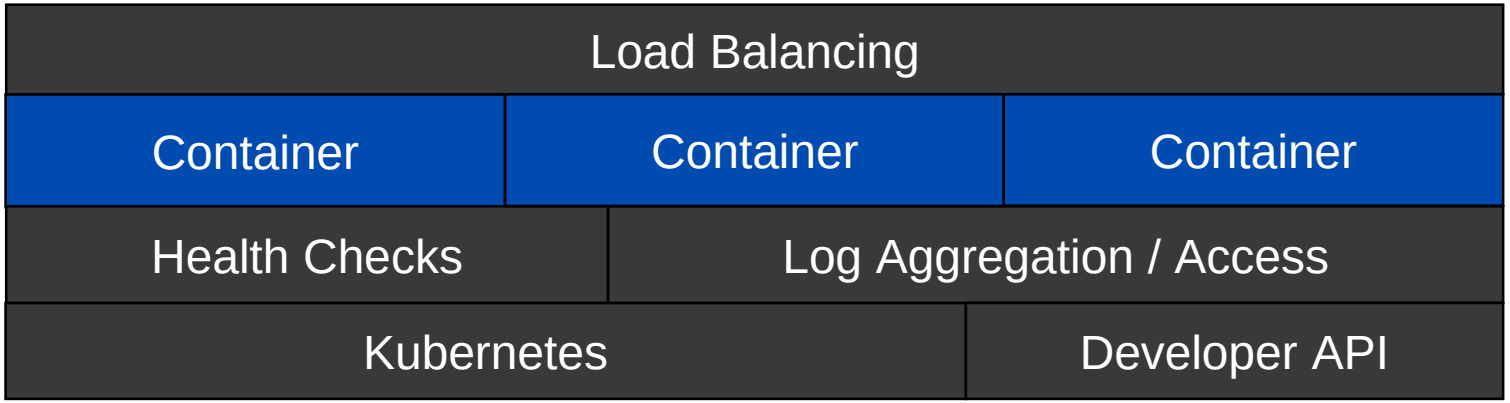


Контейнерная оркестрация: Kubernetes и платформы на его основе

Основные функции контейнерного оркестратора

- Объединение хостов в единый управляемый кластер
- Планирование запуска контейнеров на хостах кластера
- Организация сетевого взаимодействия между контейнерами на разных хостах и с внешним миром
- Связь контейнеров и хранения
- Связь однородных контейнеров в конструкции более высокого уровня
- Контроль и оптимизация использования ресурсов

Как выглядит использование оркестратора



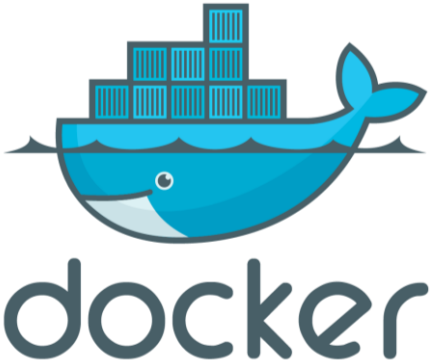
$ kubectl scale deployment <name> --replicas=3

Что такое Kubernetes (K8s)?

- Kubernetes - система оркестрации для контейнеров; автоматизирует их развертывание, масштабирование и управление контейнеризированными приложениями
- Основана на опыте Google по разработке её внутренней системы Google Borg System, в 2015 Google передала как Open Source проект в Cloud Native Computing Foundation (CNCF)
- Сообщество Kubernetes выпускает новый релиз примерно каждые три месяца, текущая версия 1.19 (на октябрь 2020)
- Поддерживается различными облачными средами
- Богатая экосистема плагинов для планирования, хранения, сети

Kubernetes и Docker

- Kubernetes (обычно) использует Docker для запуска контейнеров
- Kubernetes добавляет поверх Docker функциональность оркестрации



Терминология Kubernetes
Pod

- Pod – ресурсная единица планирования в Kubernetes. Логический набор контейнеров, которые всегда запускаются вместе
- Набор контейнеров внутри pod разделяют один IP адрес

```
[root@k8s-01-p1 ~]# kubectl get pod  --namespace=kube-system
NAME                                          READY     STATUS    RESTARTS   AGE
aci-containers-controller-1201600828-qsw5g    1/1       Running   1          69d
aci-containers-host-lt9kl                     3/3       Running   0          72d
aci-containers-host-xnwkr                     3/3       Running   0          58d
aci-containers-openvswitch-0rjbw              1/1       Running   0          58d
aci-containers-openvswitch-7j1h5              1/1       Running   0          72d
```

Терминология Kubernetes Deployment

- Deployments это набор pod-ов, которые предоставляют один и тот же сервис
- Вы описываете желаемое состояние в объекте Deployment, далее Deployment controller приводит актуальное состояние к желаемому с контролируемой скоростью
- Например вы можете создать deployment в котором декларировать что вам необходимо две копии front-end pod

```
[root@k8s-01-p1 ~]# kubectl get deployment --namespace=kube-system
NAME                        DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
aci-containers-controller   1         1         1            1           72d
```

Терминология Kubernetes Service

- Объект service обеспечивает остальным Kubernetes компонентам (включая другие pod и deployment) доступ к вашему приложению
- Pod-ы, которые представляют собой сервис, могут запускаться и терминироваться, IP адрес и порт/ы сервиса остаются неизменными
- Kubernetes автоматически балансирует нагрузку между репликами внутри deployment, которые доступны как Service
- Другие приложения могут обнаружить сервис при помощи Kubernetes service discovery

```
[root@k8s-01-p1 ~]# kubectl get svc --namespace=kube-system
NAME       CLUSTER-IP   EXTERNAL-IP   PORT(S)         AGE
kube-dns   11.96.0.10   <none>        53/UDP,53/TCP   72d
```

Терминология Kubernetes
External Service

- Если к вашему сервису нужно предоставить доступ извне, то для него создаётся External Service, связанный с external IP адресом
- Входящий трафик, который приходит на external IP, будет направлен на один из service endpoint

```
[root@k8s-01-p1 ~]# kubectl get svc front-end --namespace=guest-book
NAME        CLUSTER-IP    EXTERNAL-IP    PORT(S)        AGE
front-end   11.96.0.33    11.3.0.2       80:30002/TCP   3m
```

Терминология Kubernetes
Ingress

- Ingress - набор правил доступа, которые определяют как входящие соединения могут получить доступ к сервисам кластера
- Может использоваться для определения URL по которому сервис будет доступен извне, балансировки трафика, терминации SSL и т.д.
- Для реализации правил доступа используется Ingress Controller, например NGINX

```
[root@k8s-01-p1 ~]# kubectl get ingress
NAME           HOSTS     ADDRESS   PORTS     AGE
test-ingress   *                   80        7s
```

Терминология Kubernetes Label

- Kubernetes использует метки (label) для идентификации сущностей
- Могут использоваться для определения ролей или других важных атрибутов
- Все в Kubernetes может найти по метке
  - Например можно сделать запрос на отображение всех Pod-ов с меткой “PreProduction”

```
[root@k8s-01-p1 ~]# kubectl get pod --namespace=kube-system -l component=kube-apiserver
NAME                        READY     STATUS    RESTARTS   AGE
kube-apiserver-k8s-01-p1    1/1       Running   0          72d
```

Терминология Kubernetes Annotation

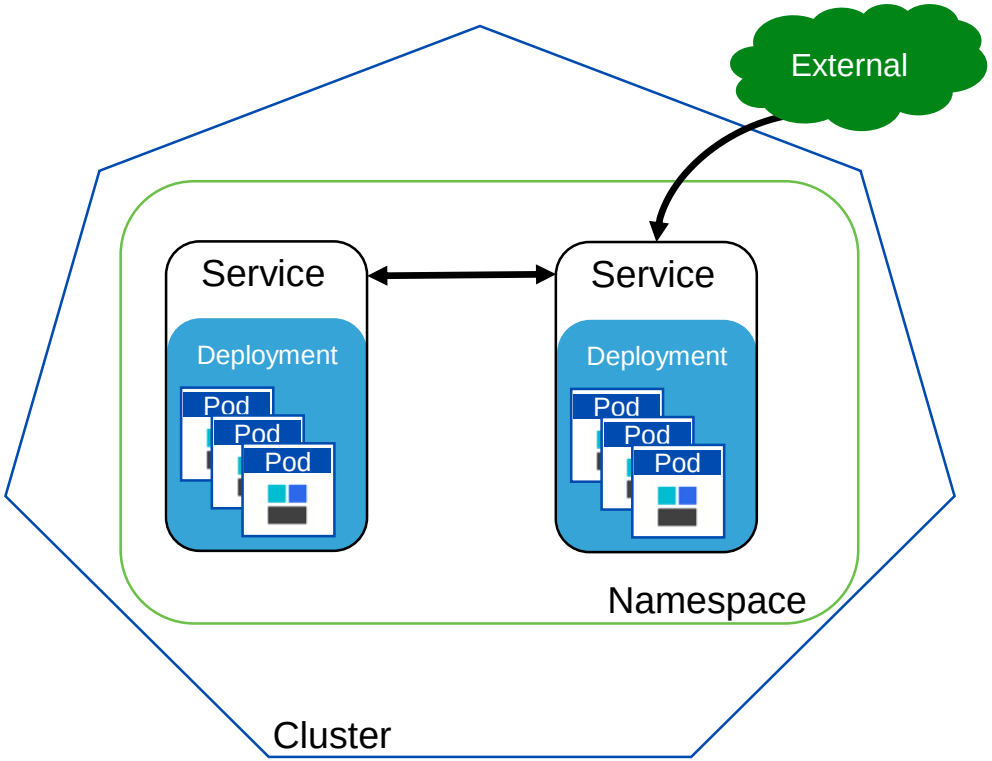
- Используются для определения произвольных метаданных
- Например, при интеграции с ACI можно будет определить EPG для контейнеров при помощи annotation

```
[root@k8s-01-p1 ~]# kubectl describe node k8s-01-p1 | more
Name:               k8s-01-p1
Role:
Labels:                   beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/hostname=k8s-01-p1
                    node-role.kubernetes.io/master=
Annotations:              node.alpha.kubernetes.io/ttl=0
                    opflex.cisco.com/pod-network-ranges={"V4":[{"start":"11.2.0.130","end":"11.2.1.1"}]}
                    opflex.cisco.com/service-endpoint={"mac":"66:85:9a:e9:ef:2f","ipv4":"11.5.0.3"}
                    volumes.kubernetes.io/controller-managed-attach-detach=true
```

Терминология Kubernetes Namespace

- Группирует все объекты вместе в единый «контур»:
  - Pod
  - Deployment
  - Volume
  - Service
  - и т.д.

Концепции Kubernetes



- **Кластер**: вся инсталяция Kubernetes
- **Namespace**: пространство имён (не означает безопасности или изоляции)
- **Pod**: ”контейнер”
- **Deployment**: реплицированный набор Pod-ов
- **Service**: набор Pod-ов и способ доступа к ним

Red Hat OpenShift Container Platform (OCP)

- OCP – контейнерная платформа с функциональностью platform as a service (PaaS).
- Коммерческое решение Red Hat, основанное на open source проекте OKD (ранее OpenShift Origin).
- OCP построена на основе Kubernetes и оптимизирована для непрерывной разработки приложений и multi-tenant deployment.
- OCP включает набор дополнительных функций безопасности и инструментов для разработчиков и операторов

Cisco Container Platform

Готовое решение для запуска контейнерной инфраструктуры на предприятий

Основано на актуальной версии Kubernetes
Доступ к обновлениям и лучшим практикам

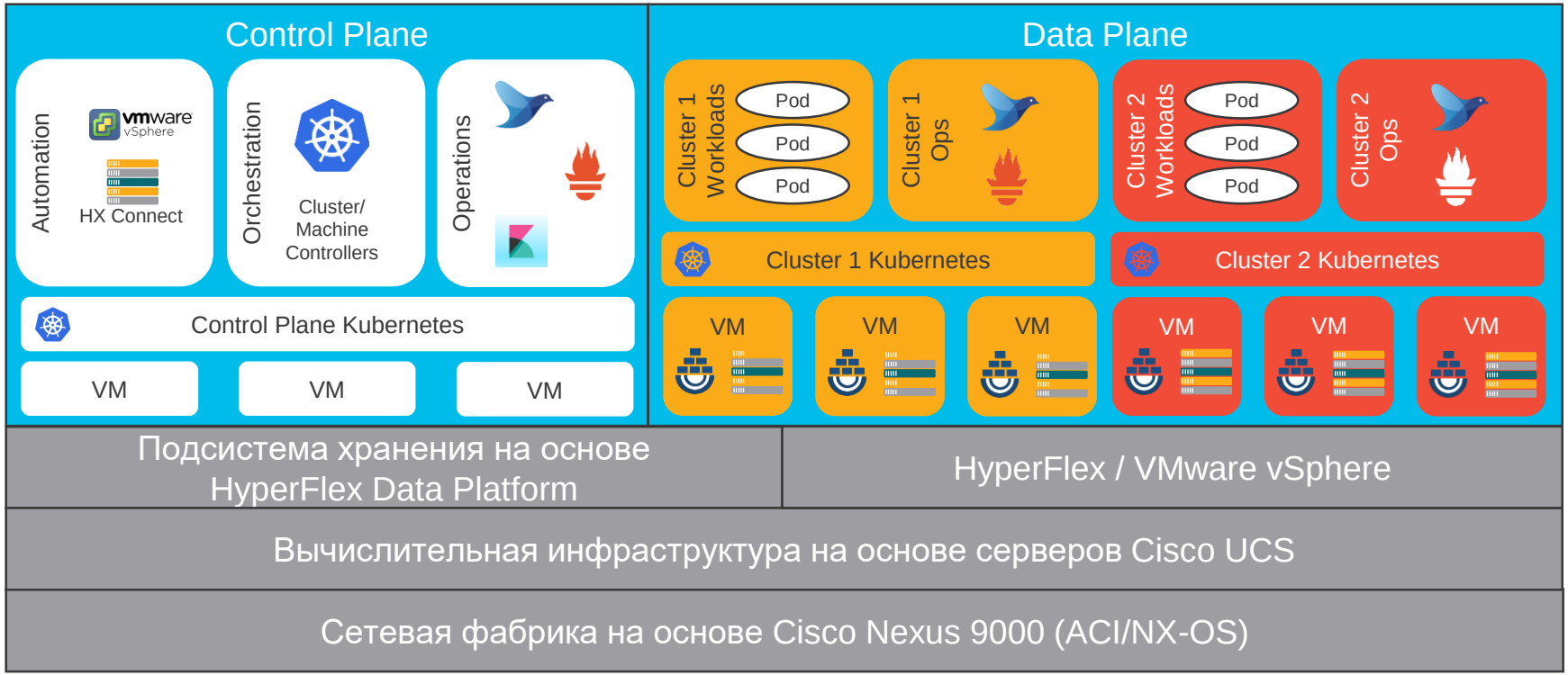
Оптимизировано для облаков
Поддержка Amazon EKS, Azure KS*, Google GKE

Интегрированное решение
Сеть | Управление | Безопасность | Мониторинг

Различные варианты запуска
Cisco HyperFlex | VMware vSphere | Cisco cVIM

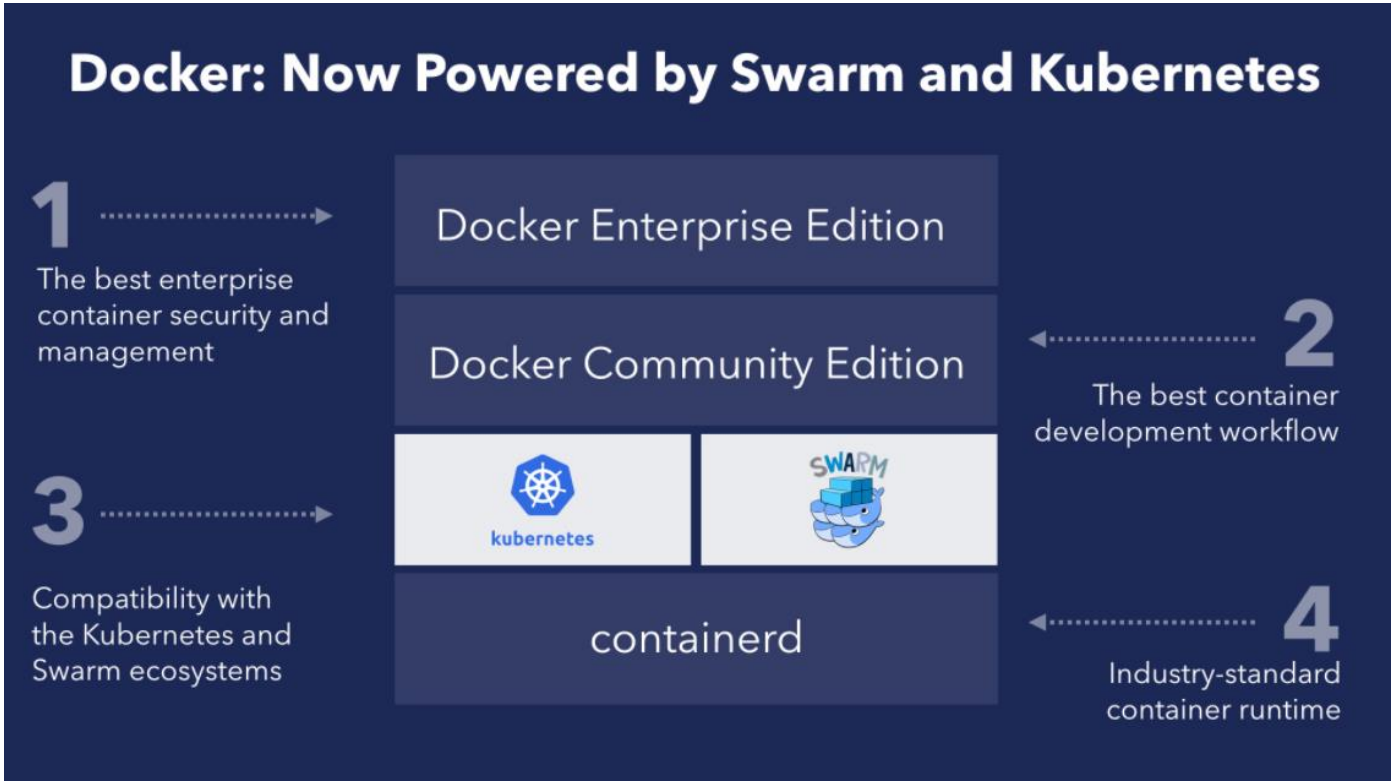
Простой запуск и управление | Основано на Open Source | Поддержка облаков | Cisco TAC

Компоненты Cisco Container Platform



Kubernetes Fluentd Prometheus & Grafana Kibana Hyperflex CNI

Docker Enterprise Edition: поддерживает Kubernetes как оркестратор



Сетевое взаимодействие контейнеров: CNI

Container Networking Interface (CNI)

- Модульное сетевое решение для контейнеров на основе плагинов
- Все CNI плагины работают по-разному, но в целом отвечают за:
  - Передачу пакетов внутри и между хостами (обычно с использованием оверлеев)
  - Знание того, где находится каждый Pod
  - Функции IPAM
  - Реализацию политик безопасности (опционально)
- Архитектура CNI используется Kubernetes: http://blog.kubernetes.io/2016/01/why-Kubernetes-doesnt-use-libnetwork.html

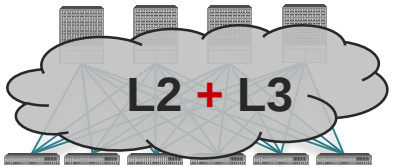
Сетевая реализация в OpenShift

- 3 режима встроенной сетевой поддержки:
  - **ovs-subnet**: «плоская» сеть, где любой pod может взаимодействовать с любым pod и service.
  - **ovs-multitenant**: изоляция pod и service между проектами
  - **ovs-networkpolicy**: администраторы проектов настраивают свои политики изоляции с использованием объектов NetworkPolicy
- Или использование стороннего SDN решения с использованием CNI Plugin - например, ACI CNI Plugin

Cisco ACI для контейнерных сред

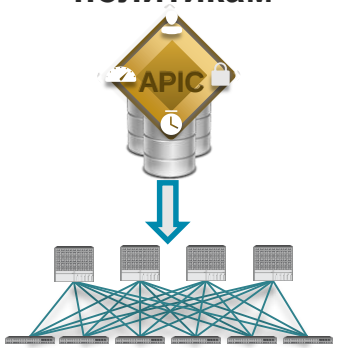
Ключевые аспекты Cisco ACI

Современная сетевая фабрика



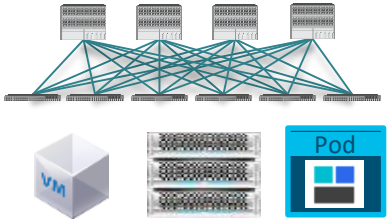
- Коммутация L2+L3 через опорную IP сеть
- Распределенный «шлюз по умолчанию»
- Хостовые маршруты (/32): оптимизация и уход от фладинга
- Внедрение в одном и во многих ЦОД

Центральное управление по политикам



- Единое управление через GUI, CLI, API
- Развёртывание, настройка и мониторинг: FCAPS
- Декларативный подход

Физические серверы + VM + контейнеры



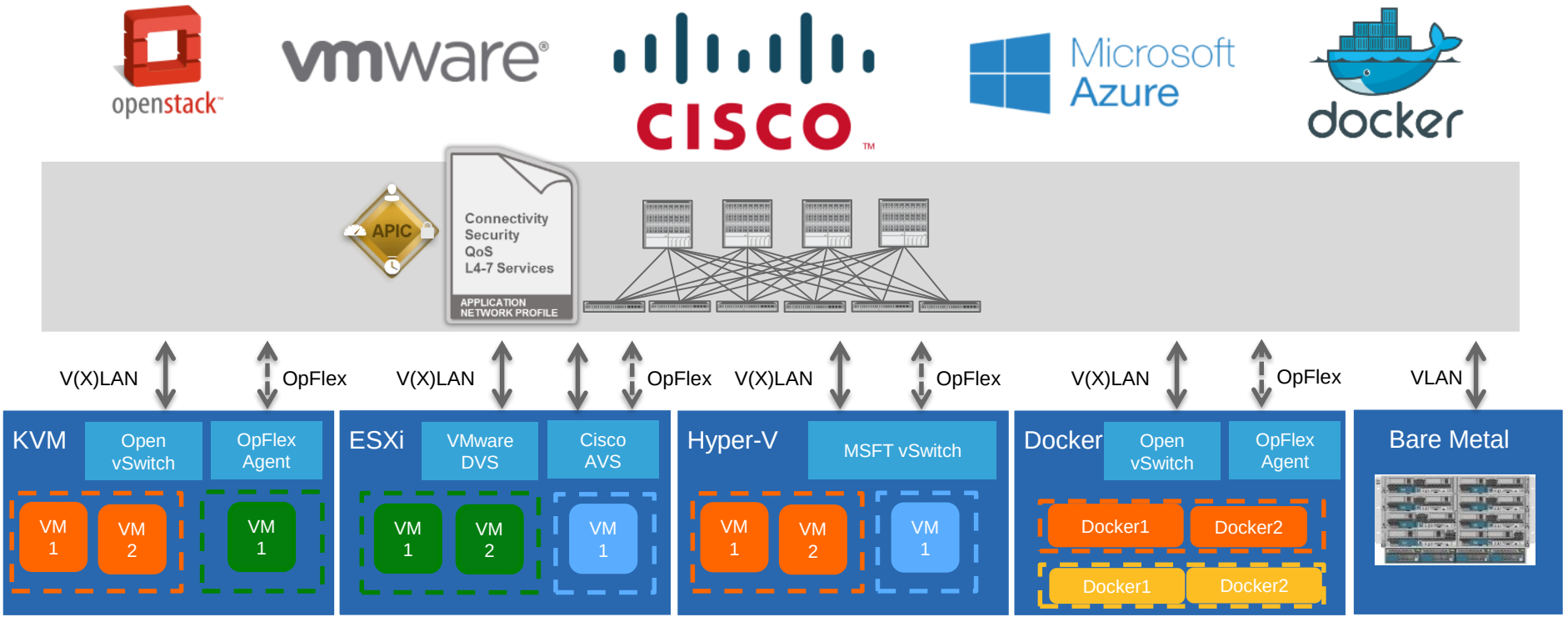
- Интеграция со средами виртуализации
- Поддержка контейнерных платформ
- Сетевой транспорт, безопасность, мониторинг

Интегрированные политики безопасности

- Принцип «белого списка» по умолчанию
- Политики безопасности, независимые от адресов
- Микросегментация
- Сегментация транспорта и управления

ACI – мультивендорное решение

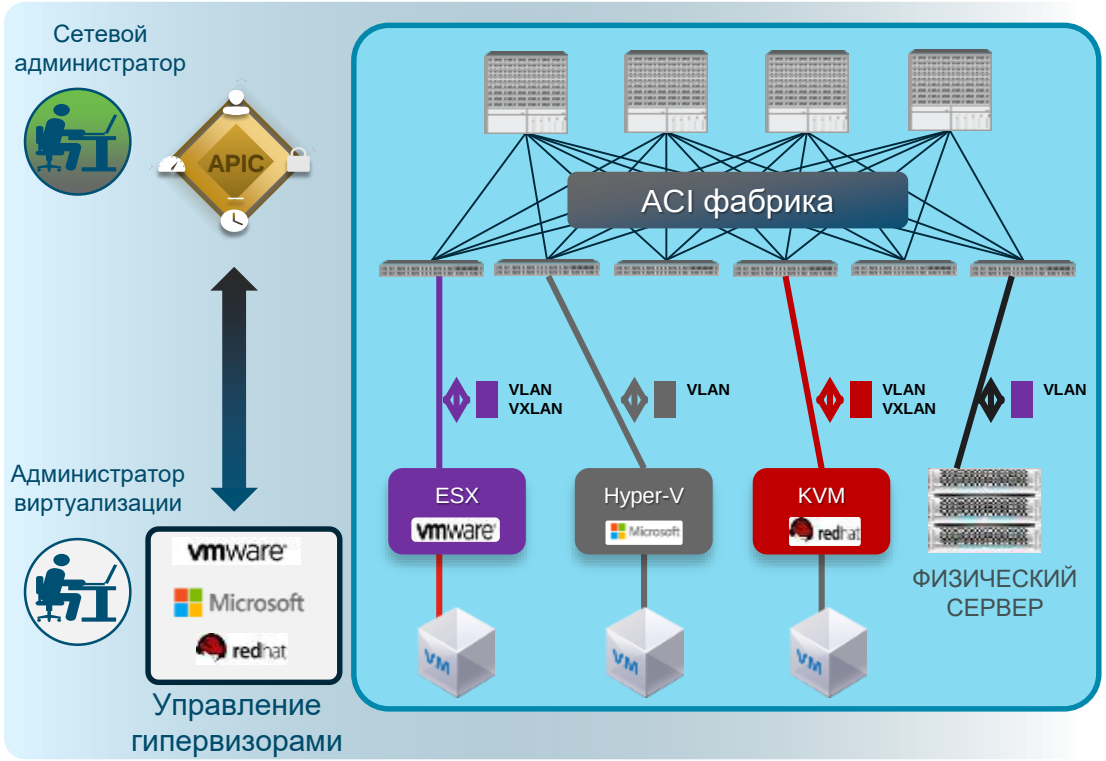
Заказчик не ограничен в выборе платформы виртуализации/облака



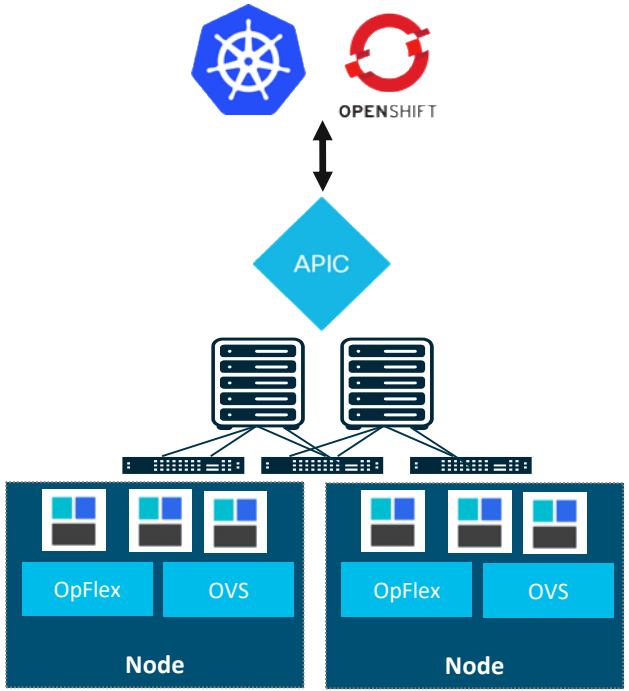
Фабрика с поддержкой нескольких гипервизоров

Интеграция с физическим и виртуальным миром

- Заказчик не ограничен в выборе платформы виртуализации: VMWare, Microsoft, KVM/OpenStack, RedHat RHV или невиртуализированных серверов
- Возможность использования нескольких VMM в одной группе EPG
- Интегрированный шлюз для VLAN и VxLAN сетей
- Не требуется дополнительное лицензирование
- **Поддержка контейнеров: Kubernetes, OpenShift, Docker EE**



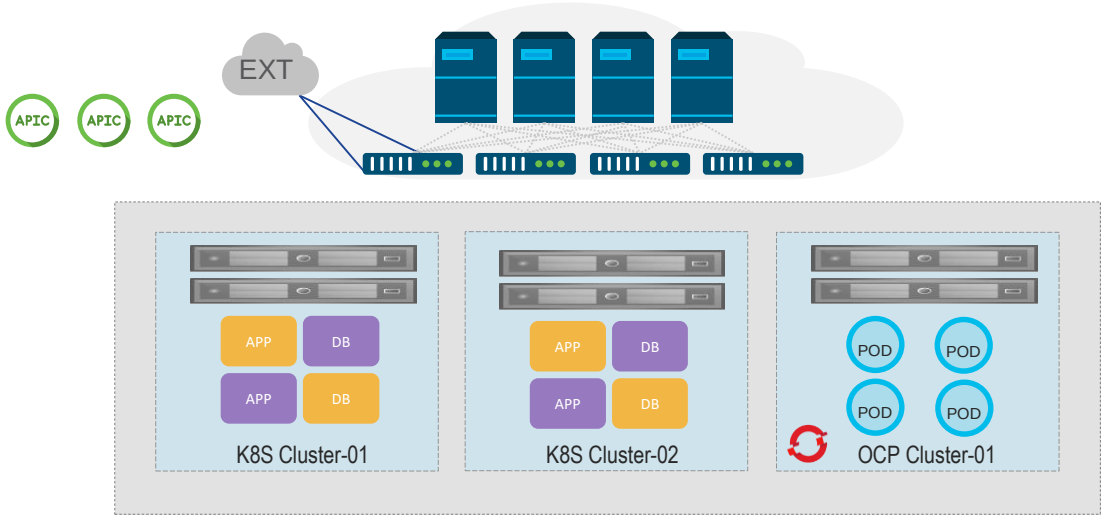
Cisco ACI и интеграция с контейнерами



ACI и Контейнеры

- Единая сеть : контейнеры, BM и физические серверы с распределённой коммутацией L2/L3
- Балансировка нагрузки на внешние сервисы средствами фабрики для обеспечения производительности и HA, опционально SNAT
- Изоляция при помощи multi-tenancy, а так же интеграция правил доступа Kubernetes Network Policy с политиками ACI
- Видимость и статистика в APIC контроллере на уровне namespace, deployment, service, pod

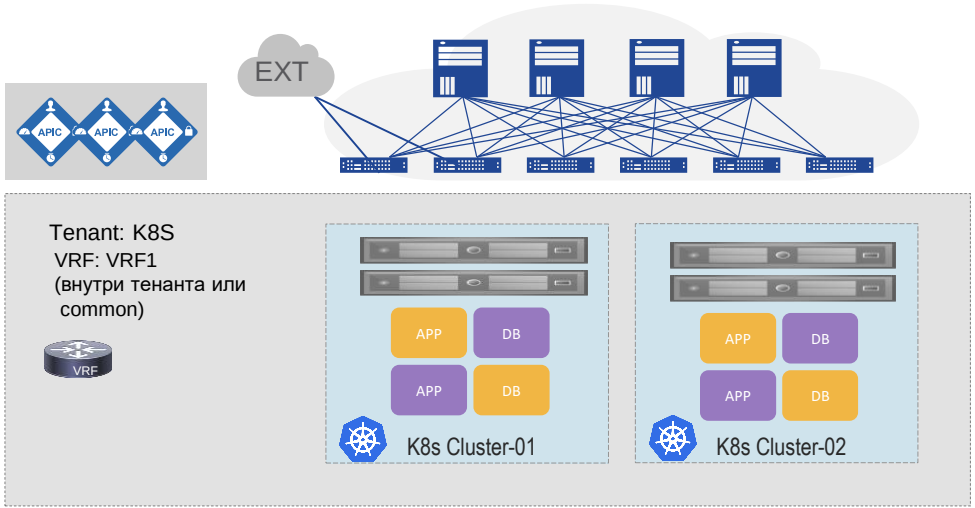
Много кластеров Kubernetes/OpenShift на одной ACI фабрике



Вы можете развернуть много кластеров Kubernetes/OpenShift на одной ACI фабрике, например для разных подразделений, разделения Production/Testing и т.д.

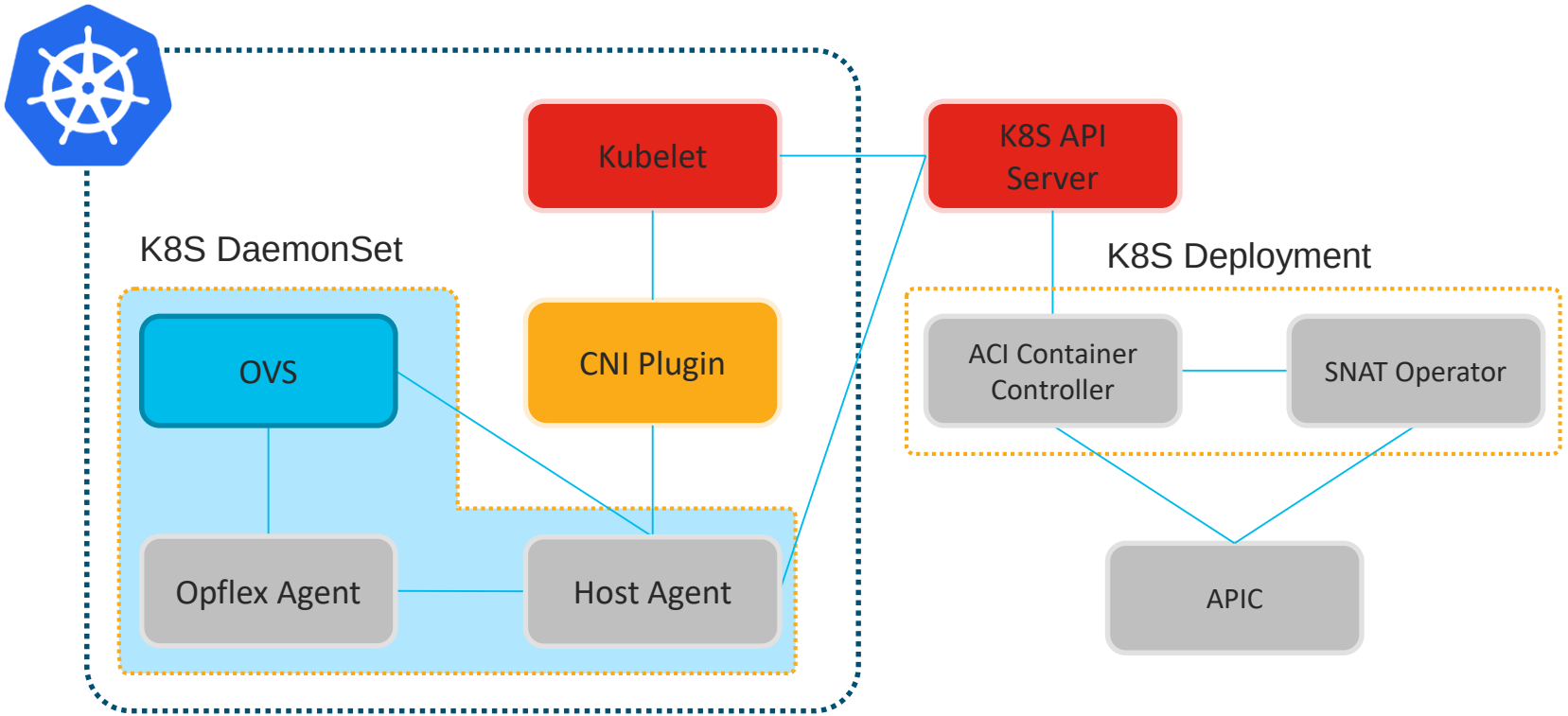
Интеграция Kubernetes кластеров в ACI

- Kubernetes кластер разворачиваются внутри ACI Tenant-a (существующего или нового)
- ACI CNI installer создаёт Bridge Domais и EPG для Kubernetes узлов и настраивает подсети внутри VRF
- The ACI CNI также сам создаёт VMM Domain и все необходимые для него объекты
- ACI поддерживает работу множества K8S кластеров на одной фабрике
- ACI CNI plugin совместим с Multi-POD



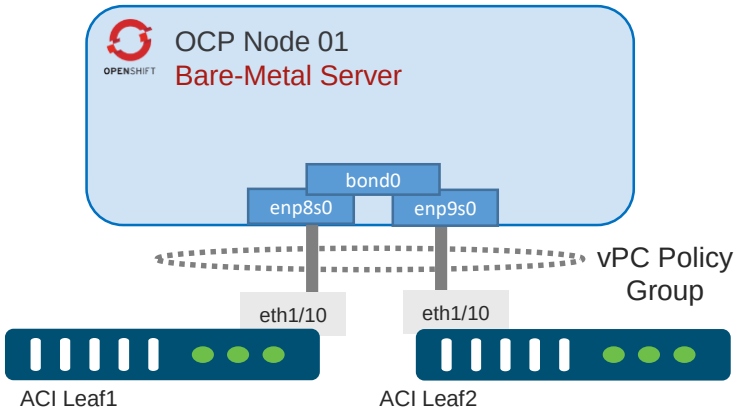
Интеграция Kubernetes кластеров в ACI

Архитектура ACI CNI



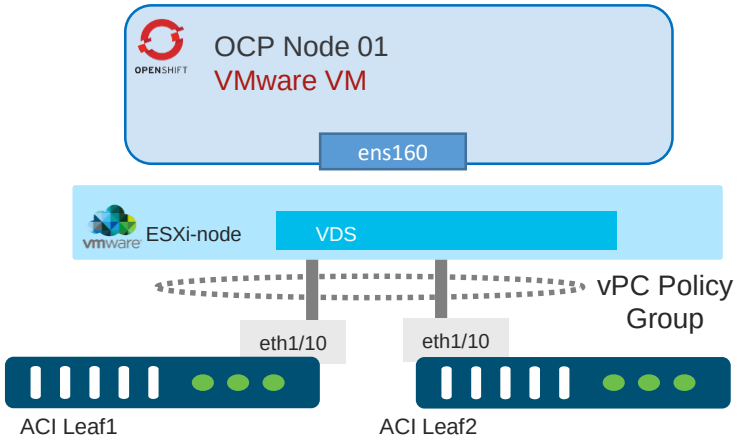
Варианты внедрения: Bare-Metal сервера

- Может использоваться подключение по VPC и bonding на основе LACP между нодами K8s и Leaf коммутаторами ACI для оптимальной балансировки и отказоустойчивости
- Поддерживается для K8S, OpenShift, Docker EE



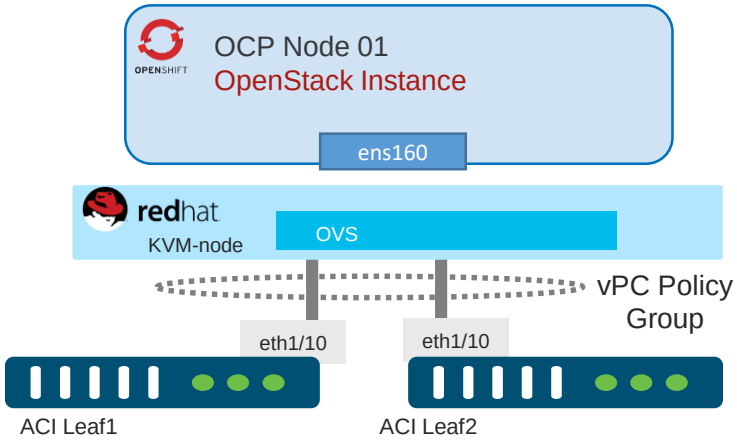
Варианты внедрения: виртуальные машины VMware vSphere

- ВМ может быть подключена через VMware VMM домен
- ВМ подключается к транковой порт-группе, которую создаёт инсталлятор ACI CNI плагина (acc_provision)
- Поддерживается для K8S, OpenShift, Docker EE, CCP



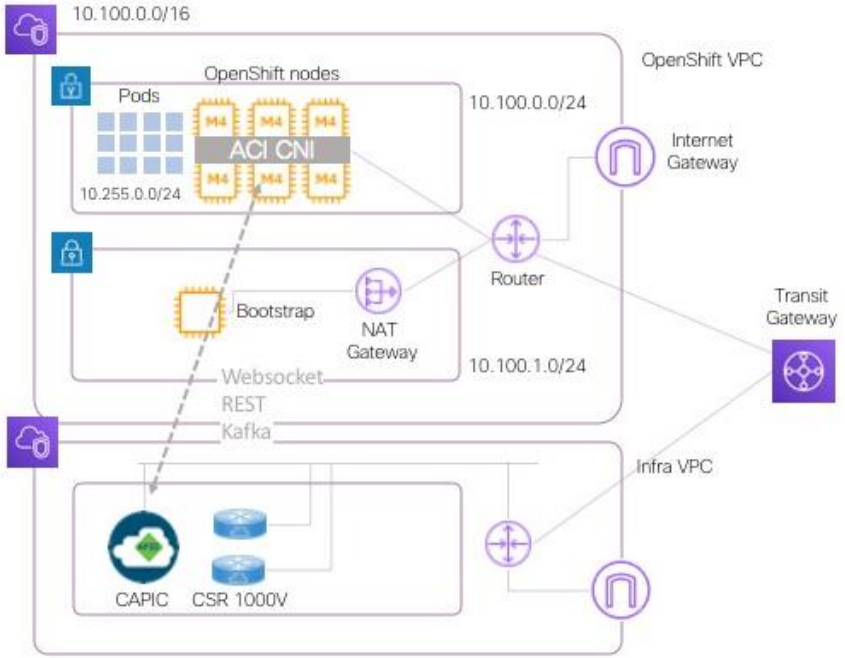
Варианты внедрения: виртуальные машины RedHat OpenStack Platform

- Может использоваться BM на кластере RedHat OpenStack (должен использоваться OpenStack VMM домен)
- Нет необходимости в создании сетевых политик (Security Groups) в OpenStack
- Не происходит «двойной инкапусляции»
- Поддерживается для OpenShift



Варианты внедрения: облако

- Продолжение функций ACI CNI для облачной модели внедрения Cloud ACI
- Поддержка общей модели политик и мониторинга
- В данный момент (ACI 5.0) поддерживается для OpenShift 4.3 IPI в облаке Amazon EC2



Официальная матрица поддержки ACI для виртуализации и контейнеров

https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/aci/virtualization/matrix/virtmatrix.html

	3.0(1)	3.0(2)	3.1(1)	3.1(2)	3.2(1)	3.2(2)	3.2(3)	3.2(4) to 3.2(6)	3.2(7) to 3.2(9)	4.0(1)	4.0(2) 4.0(3)	4.1(1) 4.1(2)	4.2(1) to 4.2(3)	4.2(4)	4.2(5)	5.0(1)	5.0(2)
OpenShift 3.6	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x
OpenShift 3.9	x	x	x	x	x	Note	Note	Note	Note	Note	Note	Note	Note	Note	Note	Note	Note
OpenShift 3.10	x	x	x	x	x	Note	Note	Note	Note	x	x	Note	Note	Note	Note	Note	Note
OpenShift 3.11	x	x	x	x	x	Note	Note	Note	Note	x	x	Note	Note	Note	Note	Note	Note
OpenShift 4.3	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	Note	Note
Kubernetes Upstream 1.6	Note			x	x	x	x	x	x	x	x	x	x	x	x	x	x
Kubernetes Upstream 1.7	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x
Kubernetes Upstream 1.9	x	x	x	x	Note	Note	Note	x	x	x	x	x	x	x	x	x	x
Kubernetes Upstream 1.10	x	x	x	x	Note	Note	Note	Note	Note	Note	x	x	x	x	x	x	x
Kubernetes Upstream 1.11	x	x	x	x	x	Note	Note	Note	Note	Note	Note	x	x	x	x	x	x
Kubernetes Upstream 1.12	x	x	x	x	x	Note	Note	Note	Note	Note	Note	Note	x	x	x	x	x
Kubernetes Upstream 1.13	x	x	x	x	x	Note	Note	Note	Note	x	x	Note	Note	x	x	x	x
Kubernetes Upstream 1.14	x	x	x	x	x	x	x	x	x	x	x	x	Note	x	x	x	x
Kubernetes Upstream 1.15	x	x	x	x	x	x	x	x	x	x	x	x	Note	Note	Note	Note	Note
Kubernetes Upstream 1.16	x	x	x	x	x	x	x	x	x	x	x	x	Note	Note	Note	Note	Note
Kubernetes Upstream 1.17	x	x	x	x	x	x	x	x	x	x	x	x	Note	Note	Note	Note	Note
Docker Enterprise 2.1 (UCP 3.1)	x	x	x	x	x	x	x	x	x	x	x	x	Note	Note	Note	x	x
Docker Enterprise 3.0 (UCP 3.2)	x	x	x	x	x	x	x	x	x	x	x	x	Note	Note	Note	Note	Note

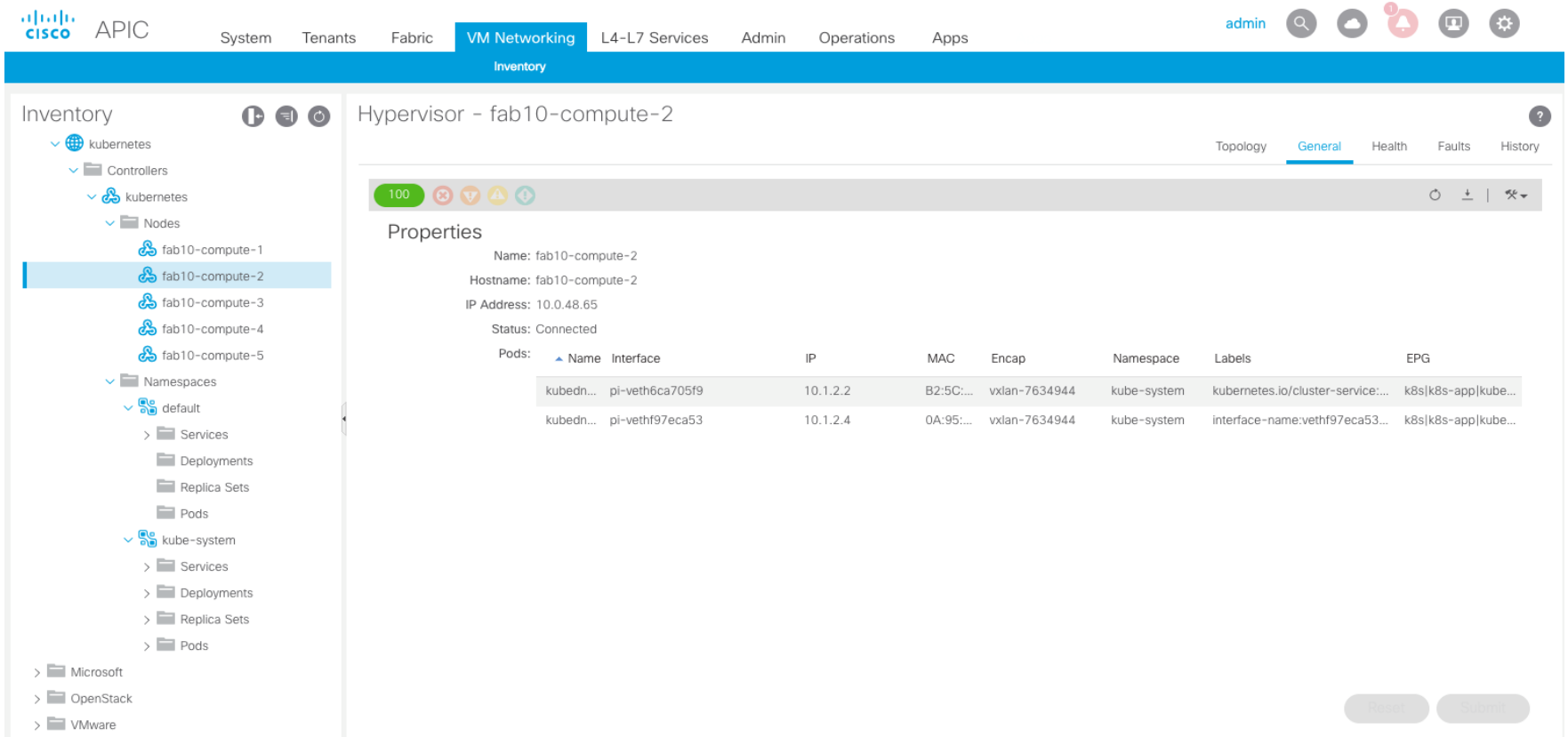
Проблемы, решаемые ACI CNI

Демонстрация функций

Обеспечение видимости

- Проблема в традиционной реализации:
  - Сетевые администраторы не видят контейнерные нагрузки и детали их трафика
  - Диагностика затруднена
- Решение благодаря интеграции ACI с контейнерными средами:
  - Видны в деталях подключения контейнеров (имена, адреса, расположение)
  - Видна статистика для контейнерного трафика
  - Видны проблемы (faults), в том числе и ретроспективно

Детальная видимость контейнерных подключений

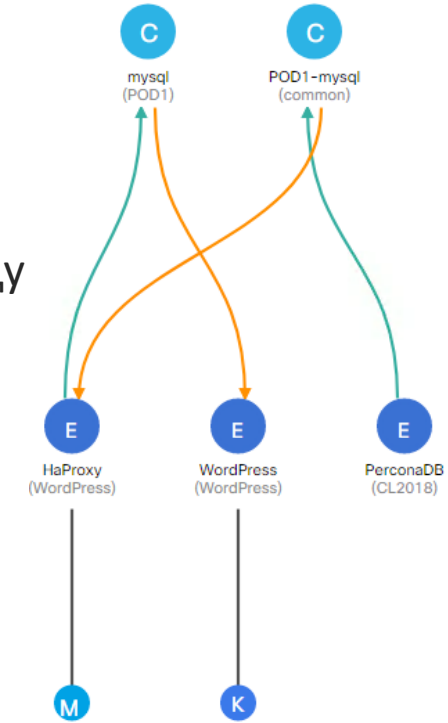


Единая среда взаимодействия с разделением зон администрирования (multi-tenancy)

- Проблема в традиционной реализации:
  - Взаимодействие между контейнерами, VM и невиртуализированными серверам требует программных «шлюзов» между наложенными и физическими сетями
  - Нет эффективных механизмов разделения полномочий для администрирования и мониторинга
- Решение благодаря интеграции ACI с контейнерными средами:
  - Поддержка контейнеров, VM и невиртуализированных серверов как равноправных подключений к фабрике
  - Взаимодействие между разными типами подключений возможно напрямую через фабрику
  - Поддержка многих доменов виртуализации и контейнеров в рамках единой фабрики
  - Механизм ролевого управления позволяет разграничить зоны управления и видимости

Единая среда взаимодействия между контейнерами и не-контейнерами

- В продуктивных средах высокопроизводительные сервисы (например БД) запускаются на физических серверах или виртуальных машинах
- Возникает необходимость организации взаимодействия между контейнерами и ВМ/физическими серверами
- Просто настройте контракт между EPG, а ACI сделает все остальное
- Поддерживается между любыми VMM доменами и физическими доменами



Применение политик безопасности/сегментации

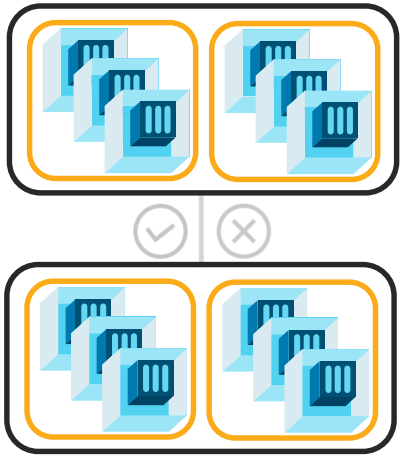
- Проблема в традиционной реализации:
  - Политики безопасности в контейнерной среде отсутствуют или применяются на «своём языке» (Kubernetes NetworkPolicy)
  - Администраторы безопасности предприятия не могут применить свои политики доступа/изоляции в контейнерных средах
- Решение благодаря интеграции ACI с контейнерными средами:
  - Совместное применение политик безопасности ACI (контракты) и политик сетевого взаимодействия K8S (Network Policy)
  - Набор вариантов отображения объектов K8S на EPG в ACI через аннотации
  - Механизм описания не «привязывает» внедрение к ACI – объекты NetworkPolicy стандарты, а аннотации не мешают работе на других контейнерных средах

Различные подходы к определению EPG для контейнерных сред

Изоляция кластера

- Одна EPG для всего кластера Kubernetes (вариант по умолчанию)
- Для внутреннего взаимодействия не нужны контракты

Изоляция Namespace



- Каждое namespace – своя EPG
- Для взаимодействия нужны контракты

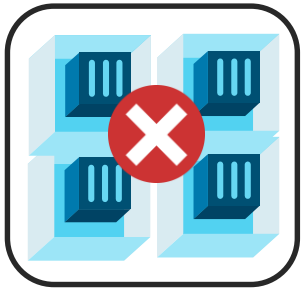
Изоляция Deployment

- Каждый deployment – своя EPG
- Жёсткий контроль с помощью контрактов

EPG NetworkPolicy

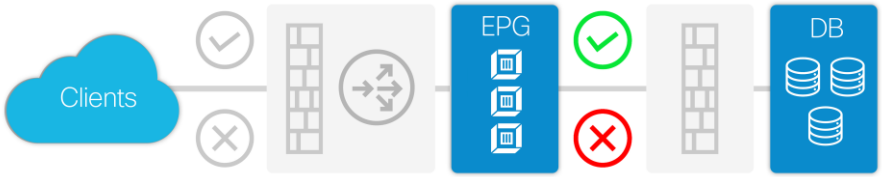
Двухуровневая реализация политик фильтрации в ACI

И Kubernetes Network Policy и ACI контракты реализуются в Linux kernel на том сервере, где запущен контейнер.



Политика через NetworkPolicy:

```
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  name: allow-red-to-blue-same-ns
spec:
  podSelector:
    matchLabels:
      type: blue
  ingress:
  - from:
    - podSelector:
        matchLabels:
          type: red
```



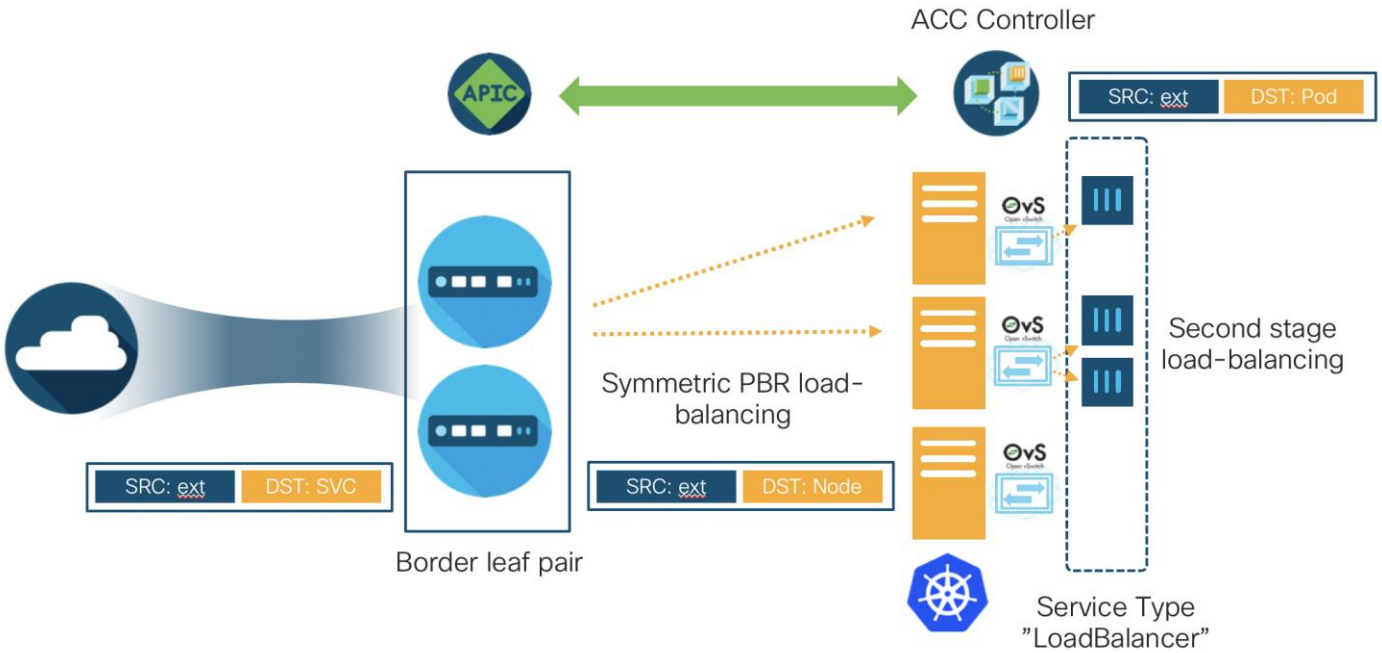
Оба механизма политик могут использоваться одновременно.

Если контейнеры отображаются на EPG, то контракты между ними реализуются на всех коммутаторах фабрики, где это необходимо. Например для трафика который из внешнего, по отношению к K8S кластеру, EPG передается на EPG который является часть K8S кластера.

Балансировка входящего трафика

- Проблема в традиционной реализации:
  - Входящий трафик требует использования внешнего балансировщика (сервис Loadbalancer), «заводится» на конкретные узлы (MetalLB) или предполагает перебрасывание трафика между узлами (сервис NodePort)
  - Проблемы производительности и отказоустойчивости
- Решение благодаря интеграции ACI с контейнерными средами:
  - Аппаратная балансировка средствами коммутаторов фабрики
  - Масштабируемость публичных сервисов до сотен Гбит/с
  - Возможность отслеживать состояние сервисов на нодах в том числе и средствами самой фабрики
  - Возможность комбинировать с Ingress для получения обработки на L7

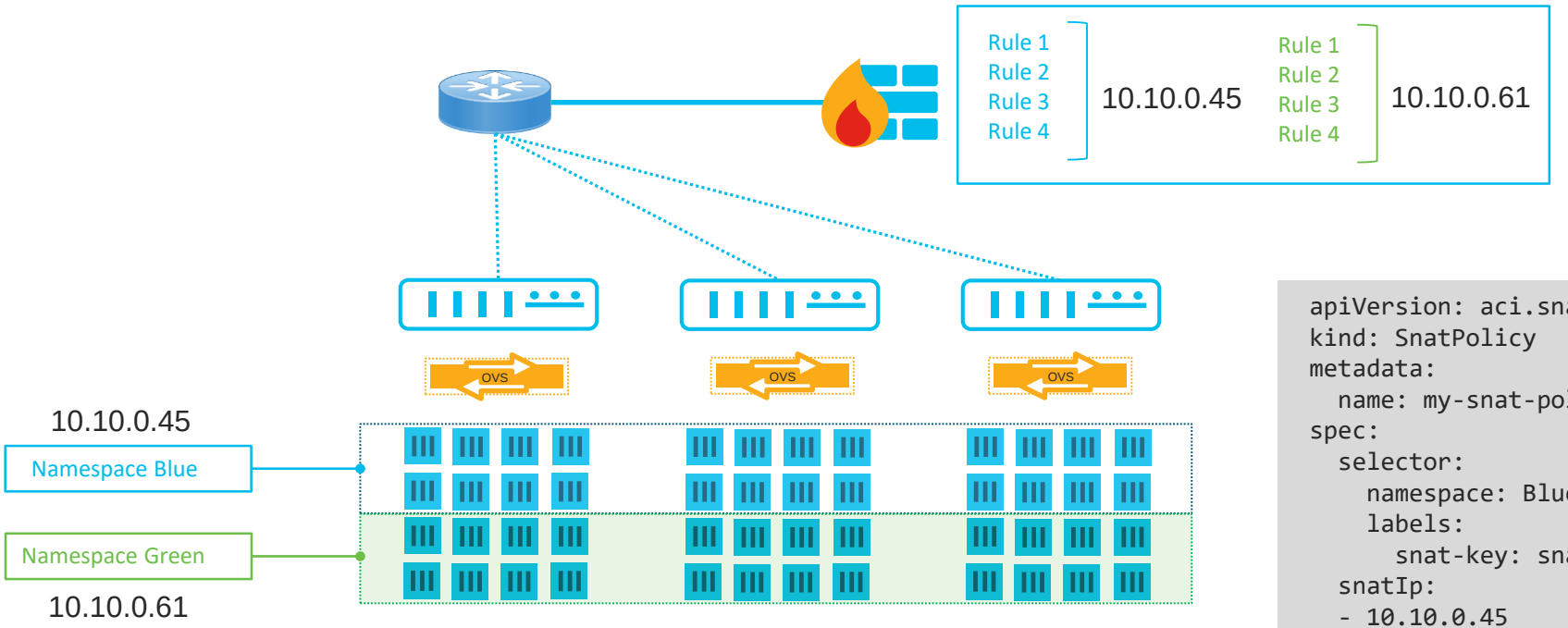
Балансировка входящего трафика



SNAT для исходящего трафика

- Проблема в традиционной реализации:
  - Исходящий трафик Pod-ов в традиционную инфраструктуру обычно отправляется с адреса worker node
  - Нельзя определить источник (pod/deployment/namespace)
  - Проблема безопасности и диагностики
- Решение благодаря интеграции ACI с контейнерными средами:
  - Применение SNAT политик
  - Source адреса или диапазоны могут быть «привязаны» к конкретным элементам контейнерного ландшафта
  - Легко идентифицировать контейнерный трафик при выходе в традиционную инфраструктуру
  - Можно при необходимости также разрешить выход Pod IP без трансляции

Применение ACI SNAT для классификации на МСЭ




```
apiVersion: aci.snat/v1
kind: SnatPolicy
metadata:
  name: my-snat-policy
spec:
  selector:
    namespace: Blue
    labels:
      snat-key: snat-value
  snatIp:
  - 10.10.0.45
```

В заключение…

Дополнительная информация

- Cisco ACI and Kubernetes Integration
  https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/kb/b_Kubernetes_Integration_with_ACI.html
- Cisco ACI and OpenShift Integration
  https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/kb/b_Cisco_ACI_and_OpenShift_Integration.html
- Cisco ACI CNI Plugin for Red Hat OpenShift Container Platform Architecture and Design Guide
  https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/white_papers/Cisco-ACI-CNI-Plugin-for-OpenShift-Architecture-and-Design-Guide.html
- Cisco DevNet: ACI CNI plug-in for Kubernetes learning track https://developer.cisco.com/learning/tracks/acik8s
- Сессии CiscoLive 2020:
  - https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2020/pdf/BRKACI-2505.pdf
  - https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2020/pdf/BRKACI-3330.pdf
  - https://www.ciscolive.com/c/dam/r/ciscolive/emea/docs/2020/pdf/BRKCLD-3181.pdf


CISCO