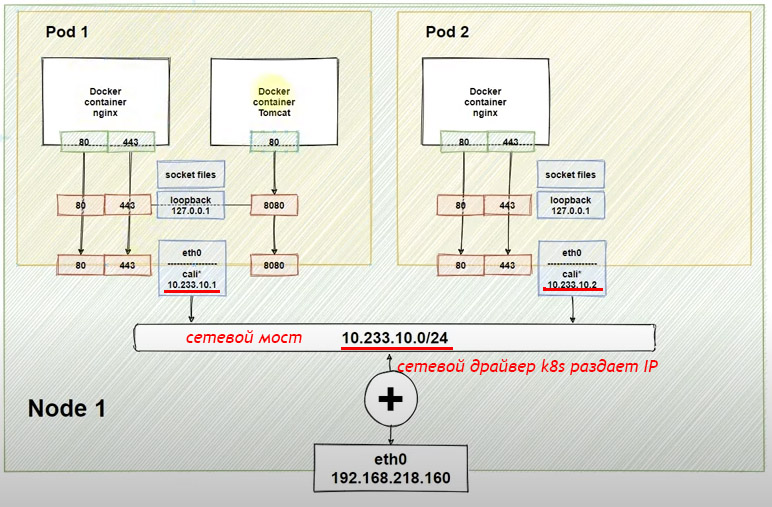
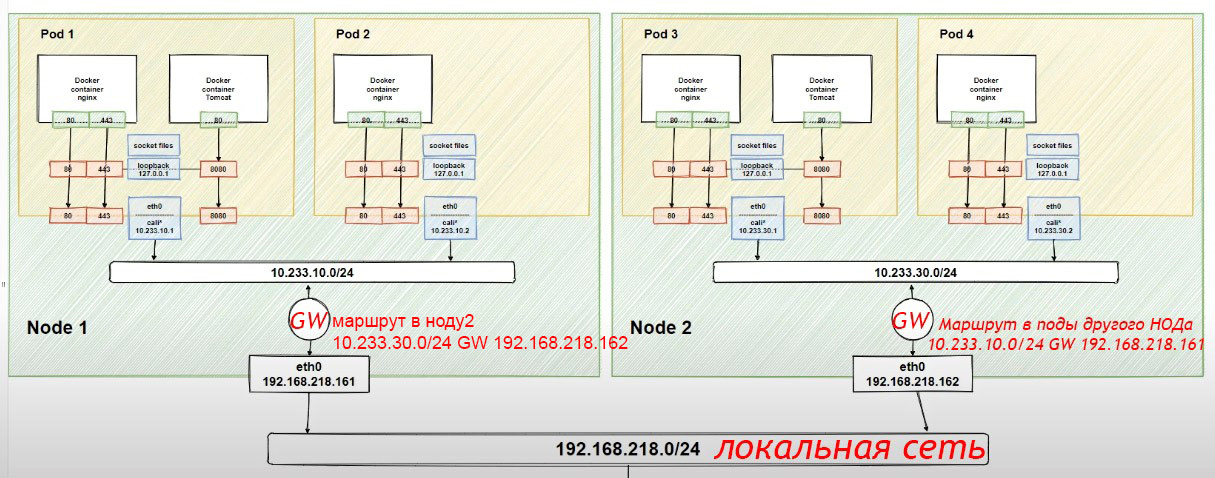
**Сети K8S**

****

**СЕТЬ POD к POD**

## **Выделение подсетей узлам для назначения IP-адресов pod'ам**

Поскольку каждый pod кластера должен иметь IP-адрес, важно убедиться в том, чтобы этот адрес был уникальным. Это достигается путем выделения каждому узлу уникальной подсети, из которой затем pod'ам на этом узле назначаются IP-адреса.

Kube-controller-manager каждому ноду присваивает podCIDR. Pod'ы каждого нода получают IP-адреса из пространства адресов в выделенном диапазоне podCIDR. Поскольку podCIDR'ы нодов не пересекаются, все pod'ы получают уникальные IP-адреса.  
  
Администратор кластера Kubernetes настраивает и устанавливает kubelet, среду запуска контейнеров, агента сетевого провайдера и копирует плагины CNI на каждый узел. Во время старта агент сетевого провайдера генерирует конфиг CNI. Когда pod планируется на узел, kubelet вызывает CRI-плагин для его создания. Далее, если используется containerd, плагин Containerd CRI вызывает CNI-плагин, указанный в конфиге CNI, для настройки сети pod'а. В результате pod получает IP-адрес.

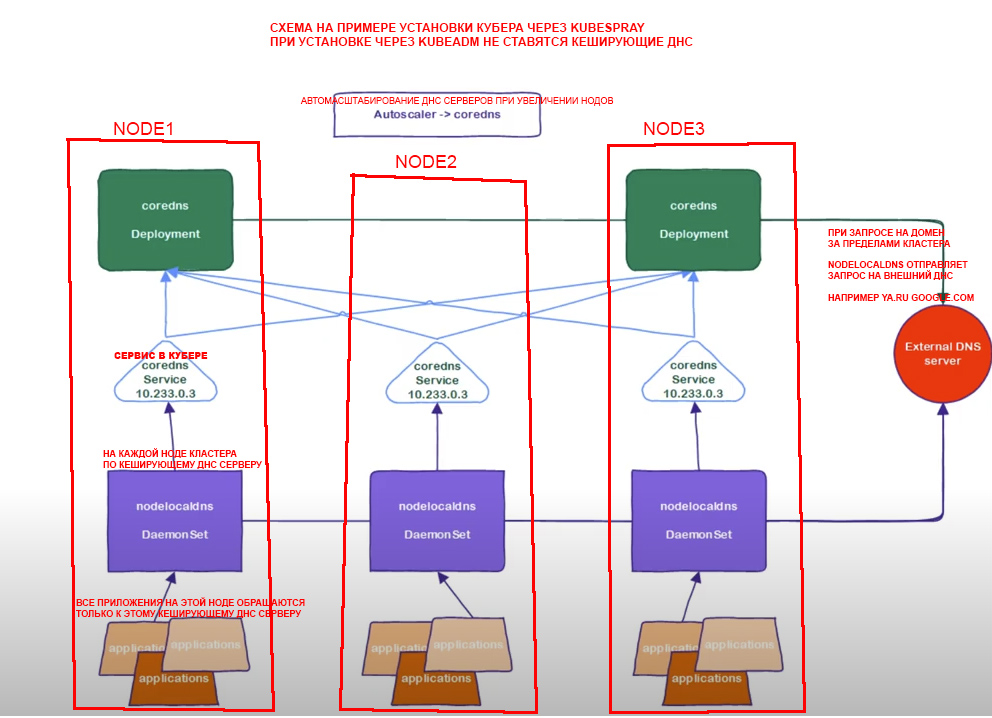
### Что такое CRI?

[CRI (Container Runtime Interface)](https://github.com/kubernetes/cri-api) — это плагин, позволяющий kubelet'у использовать разные исполняемые среды контейнеров. API CRI встроен в различные исполняемые среды, поэтому пользователи могут выбирать runtime по своему усмотрению.

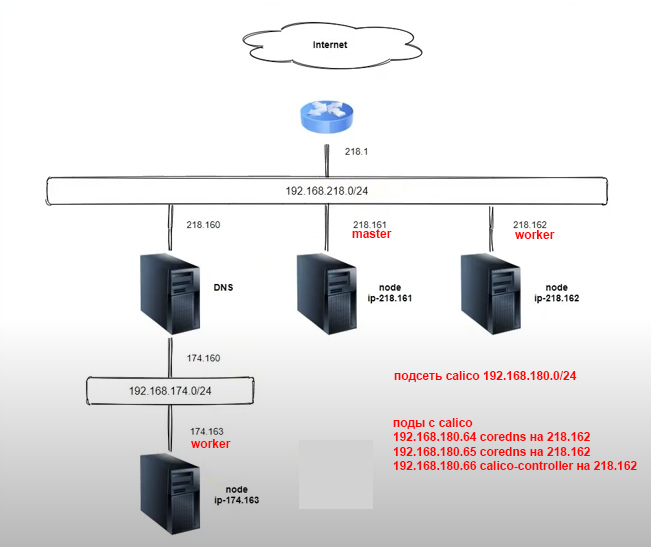
Что такое CNI?

[Проект CNI](https://github.com/containernetworking/cni) представляет собой [спецификацию](https://github.com/containernetworking/cni/blob/master/SPEC.md) для организации универсального сетевого решения для Linux-контейнеров. Кроме того, он включает в себя [плагины](https://github.com/containernetworking/plugins), отвечающие за различные функции при настройке сети pod'а. Плагин CNI — это исполняемый файл, соответствующи спецификации

**DNS**

****

**CALICO NETWORK**

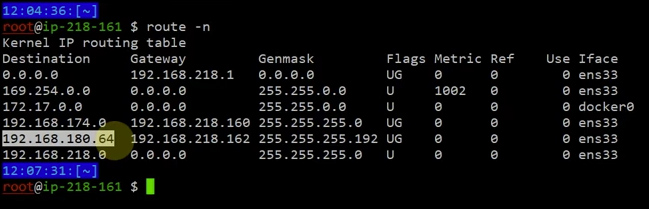
****

**Маршруты на нодах:**

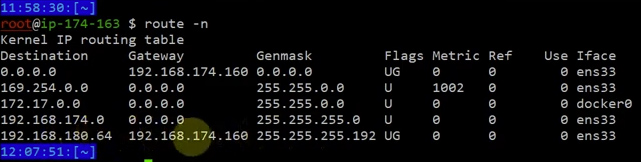
**192.168.218.162**

****

**192.168.210.161**

****

**192.168.218.163**

****

**Создадим под nginx на ноде 192.168.174.163**

<https://www.kryukov.biz/kubernetes/set-kubernetes-teoriya/calico/>

***он недостижим из 218.0/24 т.к. нет тунеля, в конфигурации calicoctl укажем ipipMode Always***

calicoctl get ippool default-ipv4-ippool -o yaml > pool.yaml

apiVersion: projectcalico.org/v3

kind: IPPool

metadata:

creationTimestamp: "2020-11-08T17:41:07Z"

name: default-ipv4-ippool

resourceVersion: "2278"

uid: 3da935c0-63ba-4c24-b63d-9f49b7549855

spec:

blockSize: 26

cidr: 192.168.180.0/24

ipipMode: Never

natOutgoing: **true**

nodeSelector: all()

vxlanMode: Never

применить конфигурацию calicoctl apply -f pool.yaml

***заработает, но будет слишком много тунелей включая внутри сети 218.0/24***

***надо установить режим:***

ipipMode: Always -> CrossSubnet

***тогда тунели будут только между подсетями***

## ***Calico IPAM***

<https://www.kryukov.biz/kubernetes/set-kubernetes-teoriya/calico/>

Kubernetes использует плагины IPAM (IP Adress Management) для выделения IP адресов подам. Проект calico предоставляет модуль: calico-ipam.

Модуль calico-ipam использует Calico IP pool для определения каким образом выделять IP адреса для подов в кластере.

calicoctl get ippool

По умолчанию используется один IP pool для всего кластера. Но его можно разделить на несколько пулов. В дальнейшем эти пулы можно назначать на под используя различные условия выбора:

* node selectos,
* аннотаций к namespaces,
* аннотаций к подам.

Calico разделяет пулы на меньшие по размеру блоки, которые прикрепляются к node. Мы уже видели эти блоки, когда смотрели таблицу маршрутизации ноды. К каждой ноде кластера может быть подключен один или несколько таких блоков. Calico будет самостоятельно добавлять и удалять их.

По умолчанию размер блока соответствует подсети /26 (64 адреса). Это параметр можно изменить как в процессе установки calico, так и во время обычной работы кластера.

calicoctl get ippool default-ipv4-ippool -o yaml

calicoctl ipam show

### Назначение пула IP адресов

Существует несколько вариантов назначения пула IP адресов. Мы посмотрим наиболее часто используемый при создании территориально распределенных кластеров.

Предположим, что первые две ноды нашего кластера расположены в одном датацентре, а третья в другом. Сеть подов кластера: 192.168.180.0/24

Необходимо, что бы первые две ноды были в подсети 192.168.200.0/24, а третья в 192.168.201.0/24

Нам потребуется выполнить следующие действия:

* Поставить метки на ноды кластера.
* Создать два IP пула, с определением нод кластера, на какие они будут применяться.
* Перевод пула default-ipv4-ippool в состяние disabled.
* Удалить (перезапустить) поды, что бы они при создании получили IP адреса из новых пулов.
* Удалить пул default-ipv4-ippool.

Ставим метки на ноды кластера:

kubectl label nodes ip-218-161 location=datacenter1

kubectl label nodes ip-218-162 location=datacenter1

kubectl label nodes ip-174-163.kryukov.local location=datacenter2

kubectl get nodes --show-labels

Создаём два пула:

---

apiVersion: projectcalico.org/v3

kind: IPPool

metadata:

name: datacenter1

spec:

cidr: 192.168.200.0/24

ipipMode: CrossSubnet

natOutgoing: **true**

nodeSelector: location == "datacenter1"

---

apiVersion: projectcalico.org/v3

kind: IPPool

metadata:

name: datacenter2

spec:

cidr: 192.168.201.0/24

ipipMode: CrossSubnet

natOutgoing: **true**

nodeSelector: location == "datacenter2"

calicoctl apply -f pool-locations.yaml

calicoctl get ippool

Переводим пул default-ipv4-ippool в состяние disabled.

calicoctl get ippool default-ipv4-ippool -o yaml > pool.yaml

vim pool.yaml

Удаляем строки:

creationTimestamp:

resourceVersion:

uid:

Добавляем:

disabled: true

Получается файл следующего содержимого:

apiVersion: projectcalico.org/v3

kind: IPPool

metadata:

name: default-ipv4-ippool

spec:

blockSize: 26

cidr: 192.168.180.0/24

ipipMode: CrossSubnet

natOutgoing: **true**

nodeSelector: all()

vxlanMode: Never

disabled: **true**

Применяем конфиг.

calicoctl apply -f pool.yaml

calicoctl get ippool -o wide

Смотрим какие поды работают на старых ip в сети:

kubectl get pods --all-namespaces -o wide | grep 192.168.180

Удаляем их.

kubectl -n kube-system rollout restart deployment/calico-kube-controllers

kubectl -n kube-system rollout restart deployment.apps/coredns

kubectl delete pod/nginx

Запускаем nginx на третей ноде:

kubectl run --image=nginx:latest nginx \

--overrides='{"apiVersion": "v1", "spec": {"nodeSelector": { "kubernetes.io/hostname": "ip-174-163.kryukov.local" }}}'

Смотрим что получилось.

kubectl get pods -o wide --all-namespaces | grep -E '192.168.200|192.168.201'

route -n

Удаляем пул.

calicoctl delete pool default-ipv4-ippool

calicoctl get ippool

route -n