

# Kubernetes ???

????????????????

- [Обобщенно](#)
- [helm](#)
  - [Это база](#)
  - [Пробуем что-то по серьезнее](#)
  - [Пишем свой чарт](#)
- [RKE2, helm](#)
- [K8s теория](#)
  - [в целом про объекты в кубере](#)
  - [Сеть в кубере](#)
  - [Размещение подов](#)
- [Мониторинг](#)
- [Ставим воркер руками](#)
- [Создаем пользователей в кубе](#)
- [cilium-dbg](#)

??????????

Хороший цикл статей: <https://ealebed.github.io/tags/kubernetes/>

Видосики:

[https://www.youtube.com/watch?list=PL8D2P0ruohOA4Y9LQoTtfsGsrWUGWpu6&v=Jp866ltZBSk&embeds\\_referring\\_euri=https%3A%2F%2Fulzette.ru%2F&source\\_ve\\_path=MjM4NTE](https://www.youtube.com/watch?list=PL8D2P0ruohOA4Y9LQoTtfsGsrWUGWpu6&v=Jp866ltZBSk&embeds_referring_euri=https%3A%2F%2Fulzette.ru%2F&source_ve_path=MjM4NTE)

Простой в установке дистрибутив: <https://k3s.io/>

Делаем

```
curl -sfL https://get.k3s.io | sh -
```

Подключаем дополнение команд

```
k3s completion bash >> ~/.bashrc  
source ~/.bashrc
```

```
??????????-?????? kubectl get no
```

??????????

????? ?????????????? ?????? ???????????

еще вариант <https://github.com/paulbouwer/hello-kubernetes>

```
in ~ λ helm repo add hello https://www.kleinloog.ch/hello-helm/  
in ~ λ helm install my-hello hello/hello --version 0.4.0-rc2  
  
in ~ λ cat ingress.yml  
apiVersion: networking.k8s.io/v1  
kind: Ingress  
metadata:  
  name: test-ingress
```

```
namespace: default
spec:
  rules:
  - host: k3s.local
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: my-hello
            port:
              number: 80
```

```
in ~ λ kubectl apply -f ingress.yml
```

????????? ??????????????

```
kubectl run -it --rm --restart=Never busybox --image=busybox:1.28 -- nslookup kubernetes.default
```

# helm

Пакетный менеджер для K8s

helm

???? ?????

Helm это по сути просто пакетник, как в любом дистре, только с возможностью подстановки своих переменных. Под капотом он просто генерит ямки с деплоями и прочими ресурсами. Поэтому что в итоге получится - зависит от фантазии автора пакета

??????????

Для работы необходим конфиг к кубу, например по пути `~/.kube/config`. Хельм не надо как-то дополнительно ставить и хранить его данные, он просто подключается к кубу и создает там все что надо. (FIXME). Чтобы его поставить, надо скачать его из официального репозитория, или, как говорят на официальном сайте,

```
“ you can curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash if you want to live on the edge
```

Чтобы что-нибудь поставить, надо добавить репу. Все как в типичном пакетнике в линуксе. Вот, например, добавим репу VictoriaMetrics и официальную репу хельма:

```
helm repo add stable https://charts.helm.sh/stable
helm repo add vm https://victoriametrics.github.io/helm-charts/
```

После чего надо обновить локальный кеш: `helm repo update`

Кроме репозитория есть еще и [хаб](#), но он выглядит как яма Dockerhub или AUR (Arch User Repository), который засран кучей чартов и никак толком не модерится. Чтобы поставить что-то из чарта, необходимо добавить репозиторий, обновить кеш, а потом поставить пакет. Смотри `helm search hub -o yaml bitnami wordpress`

????? ????-???????? ????-??????

Что мне уже нравится в хельме, так это то что ты можешь получить переменные контейнера (привет, докер) и указать их отдельным файлом при создании. В докере, например, ты должен ручками зайти на докерхаб, посмотреть, какие параметры можно передать в

контейнер и указать их каким-то макаром через ENV файл, докер-композ или еще хрен знает как.

```
# Посмотреть всю инфу о чарте
helm show all stable/mysql

# Посмотреть сам чарт
helm show chart stable/mysql

# Посмотреть переменные чарта
helm show values stable/mysql
```

Вот, например, я поднял мускуль:

```
22:40:32 losted@shit1:~/cube/helm
$ helm install stable/mysql --generate-name
WARNING: This chart is deprecated
NAME: mysql-1680378034
LAST DEPLOYED: Sat Apr 1 22:40:35 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
MySQL can be accessed via port 3306 on the following DNS name from within your cluster:
mysql-1680378034.default.svc.cluster.local

To get your root password run:

    MYSQL_ROOT_PASSWORD=$(kubectl get secret --namespace default mysql-1680378034 -o
    jsonpath="{.data.mysql-root-password}" | base64 --decode; echo)

To connect to your database:

1. Run an Ubuntu pod that you can use as a client:

    kubectl run -i --tty ubuntu --image=ubuntu:16.04 --restart=Never -- bash -il

2. Install the mysql client:

    $ apt-get update && apt-get install mysql-client -y

3. Connect using the mysql cli, then provide your password:
```

```
$ mysql -h mysql-1680378034 -p
```

To connect to your database directly from outside the K8s cluster:

```
MYSQL_HOST=127.0.0.1
```

```
MYSQL_PORT=3306
```

# Execute the following command to route the connection:

```
kubectl port-forward svc/mysql-1680378034 3306
```

```
mysql -h ${MYSQL_HOST} -P${MYSQL_PORT} -u root -p${MYSQL_ROOT_PASSWORD}
```

```
22:40:37 losted@shit1:~/cube/helm
```

```
$ helm ls
```

NAME	NAMESPACE	REVISION	CHART	APP VERSION
mysql-1680378034	default	1	2023-04-01	22:40:35.951681832 +0300
MSK deployed	mysql-1.6.9	5.7.30		

```
22:41:50 losted@shit1:~/cube/helm
```

```
$ kubectl get po
```

NAME	READY	STATUS	RESTARTS	AGE
mysql-1680378034-798b594f7b-kr2dt	0/1	Running	0	76s

Выполняем пункты 1-2-3 и проваливаемся в базу в контейнере убунты внутри куба. Либо можно пробросить порт с контейнера на локальную машину, смотри `Execute the following command to route the connection`

???????

```
23:16:27 losted@shit1:~/cube/helm
```

```
$ helm ls
```

NAME	NAMESPACE	REVISION	CHART	APP VERSION
mysql-1680378034	default	1	2023-04-01	22:40:35.951681832 +0300
MSK deployed	mysql-1.6.9	5.7.30		

```
23:16:31 losted@shit1:~/cube/helm
```

```
$ helm uninstall mysql-1680378034
```

```
release "mysql-1680378034" uninstalled
```

```
23:16:44 losted@shit1:~/cube/helm
```

```
$ kubectl get po
```

No resources found in default namespace.

helm

???????? ????-?? ?? ????????????

helm

????? ???? ???? ???? ?

# RKE2, helm

## RKE2

Хочешь применить какой-то параметр? Укажи его на всех control-plane'х, иначе ничего не применится. Например если на уже засетапленном кластере тебе приспичило отключить nginx-ingress, то `disable: rke2-ingress-nginx` надо указывать на всех rke2-server и всех их по очереди перезапускать

## Helm

```
helm list -n nginx-ingress -o yaml

# с какими параметрами засетаплен чарт
helm get -n cattle-system values rancher

# пересоздать релиз (например если изменились параметры)
helm upgrade -n nginx-ingress --force my-ingress-nginx nginx-stable/nginx-ingress --version
0.17.1 -f values.yml --set controller.disableIPV6=true
```

K8s ???????

?????? ???? ?????????? ? ?????????

Тут я тыкался в чат гпт чтобы разобраться в кубере.

??? ?????? ?????????? ?

????????????????

Сервисы (Services) в Kubernetes - это абстракция, которая позволяет управлять доступом к набору подов (Pods) в приложении или сервисе, предоставляя им стабильное сетевое имя и IP-адрес. Сервисы являются важной частью сетевой инфраструктуры Kubernetes и облегчают взаимодействие между различными компонентами вашего приложения.

Основные типы сервисов в Kubernetes:

1. **ClusterIP:** Этот тип сервиса создает внутренний IP-адрес, который доступен только внутри кластера Kubernetes. Он позволяет подам внутри кластера общаться друг с другом по этому IP-адресу. Этот тип сервиса часто используется для внутренней коммуникации между компонентами приложения.
2. **NodePort:** Сервис типа NodePort открывает порт на каждом узле кластера Kubernetes и перенаправляет трафик на соответствующий сервис. Это позволяет внешним клиентам (извне кластера) обращаться к сервису по IP-адресу узла и порту NodePort.
3. **LoadBalancer:** Тип сервиса LoadBalancer позволяет создать балансировщик нагрузки (Load Balancer) на облачном провайдере (например, AWS, GCP, Azure и т. д.), который затем распределяет трафик на сервисы вашего кластера. Это полезно, если вы хотите обеспечить балансировку нагрузки и высокую доступность для вашего приложения.
4. **ExternalName:** Этот тип сервиса предоставляет внешний DNS-именованный доступ к внешним службам, не связанным с Kubernetes. Это может быть полезно, например, для доступа к внешним базам данных или другим службам.

Сервисы в Kubernetes предоставляют абстракцию для сетевой конфигурации, что делает их более устойчивыми к изменениям внутренней структуры приложения. Они также играют ключевую роль в обеспечении масштабируемости и доступности приложений в Kubernetes-кластерах.

?.?. services ??? ??????  
??  
???????????? ? deployment ??? pod'  
???

Да, вы правильно поняли. Сервисы (Services) в Kubernetes предоставляют сетевой доступ к подам (Pods) или, более точно, к набору подов, объединенных по определенным критериям (например, селекторам). Это позволяет другим компонентам вашего приложения (или внешним клиентам) обращаться к этим подам, используя стабильное сетевое имя и IP-адрес, независимо от того, где и как они развернуты в вашем кластере.

Сервисы обеспечивают абстракцию от конкретных экземпляров подов и позволяют управлять сетевой конфигурацией приложения, что делает его более устойчивым к изменениям и масштабируемым. Они также позволяют реализовать разные типы балансировки нагрузки и организовать внешний доступ к вашему приложению, если это необходимо.

??  
? ??????????????????????????????????????

ChatGPT

Конечно, я могу объяснить разницу между деплоimentами (Deployments) и репликасетами (ReplicaSets) в Kubernetes.

### 1. Репликасет (ReplicaSet):

- **Цель:** Репликасет служит для обеспечения желаемого количества рабочих копий (подов) для приложения в кластере Kubernetes. Он создает и управляет репликами подов.
- **Обновление и масштабирование:** Репликасеты позволяют масштабировать количество реплик (подов) вверх или вниз в соответствии с заданным количеством реплик. Однако они не обеспечивают механизм автоматического обновления приложения при изменении образа контейнера.
- **Селектор:** ReplicaSet использует селекторы для определения, какие поды включать в свою репликацию. Если поды соответствуют селектору ReplicaSet,

он будет следить за их количеством и поддерживать его на заданном уровне.

## 2. Деплоймент (Deployment):

- **Цель:** Деплойменты предоставляют более высокоуровневый уровень абстракции над ReplicaSets. Они используются для управления обновлением приложения и его версий.
- **Обновление и масштабирование:** Деплойменты позволяют легко обновлять версии приложения, контролируя количество старых и новых реплик во время процесса обновления. Они также поддерживают откат на предыдущую версию в случае проблем с новой версией.
- **Репликации:** Деплойменты фактически создают и управляют ReplicaSets внутри себя. Это позволяет им управлять репликацией подов, чтобы обеспечить безопасное обновление.
- **Описание:** Деплойменты могут быть легко описаны с помощью YAML-файлов, что делает их более удобными для развертывания и обновления приложений.

Таким образом, основное отличие между ReplicaSets и Deployments заключается в том, что ReplicaSet управляет репликациями подов, в то время как Deployment обеспечивает управление процессом обновления и развертывания приложений, используя ReplicaSets под капотом. Deployment является более предпочтительным выбором, если вам нужно обновлять или разворачивать новые версии приложения в Kubernetes.

# summary

**под:** минимальная единица, один или более контейнеров, внутри обязательно `/pause`, который обеспечивает сетевые настройки

**репликасет:** сведения о реплике подов, больше там ничего быть не может. Конфигмапы, сервисы, секреты не входят явно в репликасет, они управляются другими объектами в кубернетисе

**деплоймент:** условно, описание приложения, а именно описание репликасетов и прочих объектов, там указываются репликасеты, с помощью деплоймента можно плавно выкатить, при ошибках автоматически откатить репликасеты. В деплойменте указываются:

- Реплики
- Шаблон пода
- Стратегия обновления
- Стратегия отката
- автоскейлинг
- Контроль обновления (по сколько подов обновлять деплоймент)
- метки, аннотации

абстракцией выше является:

- StatefulSet - управление приложениями с хранением состояния, например мускуль или банально сервак майнкрафта, когда мы не можем тупо размножить приложение, когда у нас домашние животные, а не стадо
- DaemonSet - развертывание одной копии пода на каждом узле кластера, например агент мониторинга, сборщик логов

???? ? ???????

Сеть в Kubernetes играет ключевую роль, так как она обеспечивает связь между всеми компонентами приложения, работающими в кластере. Организация сети в Kubernetes включает в себя несколько аспектов:

1. **Поды (Pods):** Поды - это минимальные рабочие единицы в Kubernetes, и они могут иметь свой собственный IP-адрес. Поды в одной ноде могут общаться друг с другом через локальную сеть на этой ноде без какой-либо дополнительной конфигурации. Однако для обмена данными между подами, размещенными на разных нодах, требуется сетевой роутинг.
2. **Сетевые плагины (CNI):** Kubernetes использует сетевые плагины, такие как Calico, Flannel, Cilium, и другие, для управления сетью в кластере. Эти плагины обеспечивают создание виртуальной сети, настройку маршрутизации, а также возможность применения политик безопасности сети.
3. **Сервисы (Services):** Сервисы в Kubernetes предоставляют стабильное именование и IP-адреса для доступа к набору подов. Когда вы создаете сервис, Kubernetes автоматически настраивает прокси и балансировщики нагрузки для перенаправления трафика к подам, соответствующим этому сервису. Это обеспечивает сетевую доступность и управление трафиком между подами.
4. **Ингрес-контроллеры (Ingress Controllers):** Ингрес-контроллеры позволяют управлять входящим сетевым трафиком в кластере. Они предоставляют функциональность маршрутизации трафика на основе правил и настройки SSL-терминации.
  - nginx
  - traefik
  - haproxy ingress
5. **Политики сетевой безопасности (Network Policies):** Политики сетевой безопасности позволяют ограничивать сетевой доступ между подами на основе правил. Это помогает обеспечивать безопасность приложений и сетевые политики в кластере.
6. **Сетевая политика (Network Policy):** Сетевая политика Kubernetes определяет правила сетевого доступа для селекторов подов, контролируя, какие поды могут общаться между собой и какие порты и протоколы разрешены.
7. **Интеграция с внешними сетями:** Kubernetes также позволяет интегрировать внешние сети и сервисы, что важно для обеспечения связности приложений с



#### 4. Weave:

- **Особенности:** Weave предоставляет сетевую сегментацию, маршрутизацию и возможность настройки сетевых политик.
- **Модель маршрутизации:** Weave использует маршрутизацию на основе оверлея (overlay routing) и умеет автоматически обнаруживать узлы в сети.
- **Применение:** Подходит для кластеров, требующих простой настройки сети и сетевой сегментации.

#### 5. Kube-router:

- **Особенности:** Kube-router предоставляет сетевую политику, маршрутизацию на основе хоста и дополнительные функции безопасности.
- **Модель маршрутизации:** Использует маршрутизацию на основе хоста и BGP для маршрутизации пакетов между узлами.
- **Применение:** Подходит для кластеров, где важна простота и безопасность сети.

Выбор сетевого плагина зависит от ваших требований по производительности, безопасности, масштабируемости и функциональности. При проектировании и настройке сети в Kubernetes важно учитывать конкретные потребности вашего приложения и кластера.

## ??????? (Services)

в Kubernetes - это абстракция, которая позволяет управлять доступом к набору подов (Pods) в приложении или сервисе, предоставляя им стабильное сетевое имя и IP-адрес. Сервисы являются важной частью сетевой инфраструктуры Kubernetes и облегчают взаимодействие между различными компонентами вашего приложения.

Основные типы сервисов в Kubernetes:

1. **ClusterIP:** Этот тип сервиса создает внутренний IP-адрес, который доступен только внутри кластера Kubernetes. Он позволяет подам внутри кластера общаться друг с другом по этому IP-адресу. Этот тип сервиса часто используется для внутренней коммуникации между компонентами приложения.
2. **NodePort:** Сервис типа NodePort открывает порт на каждом узле кластера Kubernetes и перенаправляет трафик на соответствующий сервис. Это позволяет внешним клиентам (извне кластера) обращаться к сервису по IP-адресу узла и порту NodePort.
3. **LoadBalancer:** Тип сервиса LoadBalancer позволяет создать балансировщик нагрузки (Load Balancer) на облачном провайдере (например, AWS, GCP, Azure и т. д.), который затем распределяет трафик на сервисы вашего кластера. Это полезно, если вы хотите обеспечить балансировку нагрузки и высокую доступность для вашего приложения.
4. **ExternalName:** Этот тип сервиса предоставляет внешний DNS-именованный доступ к внешним службам, не связанным с Kubernetes. Это может быть полезно, например,



12. **!! Volume:** Volume используется для монтирования директорий и файлов в поды. Различные типы Volume поддерживают разные методы хранения и доступа к данным. **Локальные данные! КЕШ**

Это лишь некоторые из ключевых объектов и ресурсов в Kubernetes. Kubernetes предоставляет богатую экосистему инструментов и API для управления контейнеризированными приложениями и их инфраструктурой в кластере.

????????????????

- **горизонтальное** - увеличение/уменьшение кол-ва подов в зависимости от нагрузки
- **вертикальное** - управление ресурсами подов

???? ? ????????

# Service

указание, в какой порт пода стучаться.

1. **ClusterIP**: Этот тип сервиса создает внутренний IP-адрес, который доступен только внутри кластера Kubernetes. Он позволяет подам внутри кластера общаться друг с другом по этому IP-адресу. Этот тип сервиса часто используется для внутренней коммуникации между компонентами приложения.
2. **NodePort**: Сервис типа NodePort открывает порт на каждом узле кластера Kubernetes и перенаправляет трафик на соответствующий сервис. Это позволяет внешним клиентам (извне кластера) обращаться к сервису по IP-адресу узла и порту NodePort.
3. **LoadBalancer**: Тип сервиса LoadBalancer позволяет создать балансировщик нагрузки (Load Balancer) на облачном провайдере (например, AWS, GCP, Azure и т. д.), который затем распределяет трафик на сервисы вашего кластера. Это полезно, если вы хотите обеспечить балансировку нагрузки и высокую доступность для вашего приложения. **НА SELFHOSTED КЛАСТЕРЕ ЕГО НЕТ**
4. **ExternalName**: Этот тип сервиса предоставляет внешний DNS-именованный доступ к внешним службам, не связанным с Kubernetes. Это может быть полезно, например, для доступа к внешним базам данных или другим службам.

Скорее всего тебе придется иметь дело либо с NodePort, либо с ClusterIP.

**TLDR:** ClusterIP - айпишник внутри кластера. Это здравый подход, на этот айпишник надо натравлять ингресс. NodePort - проброс порта с пода напрямую на адрес ноды. Не делай так, это некруто.

# Ingress

Описание того, что куда и как надо проксировать. Не путать с описанием ингресс контроллера! Ингресс ничего никуда не проксирует!

# Ingress Controller

Само приложение, которое проксирует условные http запросы на под. Например nginx.

????????????? ?????????????? ??  
????????????? ? ??????????

```
helm repo add hello https://www.kleinloog.ch/hello-helm/  
helm install my-hello hello/hello --version 0.4.0-rc2
```

K8s теория

???????????? ???? ?

Допустим мы хотим разместить ингрессы на машинах, которые мы (в своей голове) выделили под ингрессы. Одного желания тут недостаточно, один из способов - лейблы на нодах, nodeSelector в манифесте, taint'ы, Toleration'ы.

???????????? ???? ?

Мы можем пометить ноду, чтобы на ней размещались только определенные поды. Делается это так:

```
kubectl label nodes k8s-ingress-1 nodeType=ingress
```

Теперь если мы посмотрим конфиг ноды, увидим там этот лейбл:

```
kubectl get no -o yaml k8s-ingress-1
apiVersion: v1
kind: Node
metadata:
  annotations:
    ...
  creationTimestamp: "2023-10-04T14:44:13Z"
  finalizers:
  - wrangler.cattle.io/node
  - wrangler.cattle.io/managed-etcd-controller
  labels:
    beta.kubernetes.io/arch: amd64
    beta.kubernetes.io/instance-type: rke2
    beta.kubernetes.io/os: linux
    kubernetes.io/arch: amd64
    kubernetes.io/hostname: k8s-ingress-1
    kubernetes.io/os: linux
    node.kubernetes.io/instance-type: rke2
    nodeType: ingress # <--- Вот наш лейбл
```

Мы сделали так, чтобы наши поды попадали на эту ноду и только на нее, но на нее будут попадать также **другие поды**.

Чтобы этого избежать нам нужны **taint'ы**. Теинты указывают, где размещать поды нельзя. Чтобы сказать шедулеру, чтоб он убрал все поды кроме уже запущенных, повесим теинт NoSchedule

```
kubectl taint nodes k8s-ingress-1 ingress-taint=true:NoSchedule
```

```
kubectl get no -o yaml k8s-ingress-1
```

```
apiVersion: v1
```

```
kind: Node
```

```
metadata:
```

```
  annotations:
```

```
    ...
```

```
  creationTimestamp: "2023-10-04T14:44:13Z"
```

```
  finalizers:
```

- wrangler.cattle.io/node
- wrangler.cattle.io/managed-etcd-controller

```
  labels:
```

```
    beta.kubernetes.io/arch: amd64
    beta.kubernetes.io/instance-type: rke2
    beta.kubernetes.io/os: linux
    kubernetes.io/arch: amd64
    kubernetes.io/hostname: k8s-ingress-1
    kubernetes.io/os: linux
    node.kubernetes.io/instance-type: rke2
    nodeType: ingress
```

```
  name: k8s-ingress-1
```

```
  resourceVersion: "1765082"
```

```
  uid: XXX
```

```
spec:
```

```
  podCIDR: X.X.X.X/24
```

```
  podCIDRs:
```

- X.X.X.X/24

```
  providerID: rke2://k8s-ingress-1
```

```
  taints:
```

- effect: NoSchedule # <----

```
    key: ingress-taint
```

```
    value: "true"
```

?????? ?????????? ???????

Указываем в манифесте пода/хельм чарта следующее:

```
...
nodeSelector:
  nodeType: ingress
...
tolerations:
  - key: "ingress-taint"
    operator: "Exists"
    effect: "NoSchedule"
...
```

Src:

<https://prudnitskiy.pro/post/2021-01-15-k8s-pod-distribution/>

??????????

Дашборды: <https://github.com/dotdc/grafana-dashboards-kubernetes>

Prometheus stack: <https://github.com/prometheus-community/helm-charts>

<https://artifacthub.io/packages/helm/prometheus-community/kube-prometheus-stack>



```
req_extensions      = kube-work-4_req_extensions
```

```
[kube-work-4_req_extensions]
```

```
basicConstraints    = CA:FALSE
```

```
extendedKeyUsage    = clientAuth, serverAuth
```

```
keyUsage            = critical, digitalSignature, keyEncipherment
```

```
nsCertType          = client
```

```
nsComment           = "kube-work-4 Certificate"
```

```
subjectAltName      = DNS:kube-work-4, IP:10.138.117.7 # IP воркера
```

```
subjectKeyIdentifier = hash
```

```
[kube-work-4_distinguished_name]
```

```
CN = system:node:kube-work-4
```

```
O  = system:nodes
```

```
C  = US
```

```
ST = Washington
```

```
L  = Seattle
```

```
[default_req_extensions]
```

```
basicConstraints    = CA:FALSE
```

```
extendedKeyUsage    = clientAuth
```

```
keyUsage            = critical, digitalSignature, keyEncipherment
```

```
nsCertType          = client
```

```
nsComment           = "Admin Client Certificate"
```

```
subjectKeyIdentifier = hash
```

```
###
```

```
for host in kube-work-4; do
```

```
    openssl genrsa -out "${host}.key" 4096
```

```
    openssl req -new -key "${host}.key" -sha256 \
```

```
        -config "ca.conf" -section ${host} \
```

```
        -out "${host}.csr"
```

```
    openssl x509 -req -days 3653 -in "${host}.csr" \
```

```
        -copy_extensions copyall \
```

```
        -sha256 -CA "ca.crt" \
```

```
        -CAkey "ca.key" \
```

```
-CAcreateserial \  
-out "${host}.crt"
```

```
kubectl config set-cluster kubernetes-the-hard-way \  
--certificate-authority=ca.crt \  
--embed-certs=true \  
--server=https://10.138.117.204:6443 \  
--kubeconfig=${host}.kubeconfig
```

done

###

```
root@kube-ctrl:~/add-manually-worker# cat kube-work-4.kubeconfig
```

```
apiVersion: v1
```

```
clusters:
```

```
- cluster:
```

```
  certificate-authority-data: `cat ca.crt | base64 -w0`
```

```
  server: https://10.138.117.204:6443
```

```
  name: kubernetes-the-hard-way
```

```
contexts:
```

```
- context:
```

```
  cluster: kubernetes-the-hard-way
```

```
  namespace: default
```

```
  user: system:node:kube-work-4
```

```
  name: default-context
```

```
current-context: default-context
```

```
kind: Config
```

```
preferences: {}
```

```
users:
```

```
- name: system:node:kube-work-4
```

```
  user:
```

```
    client-certificate-data: `cat crt | base64 -w0`
```

```
    client-key-data: `cat key | base64 -w0`
```

###

```
root@kube-ctrl:~/add-manually-worker# kubectl --kubeconfig kube-work-4.kubeconfig get no
```

NAME	STATUS	ROLES	AGE	VERSION
kube-work-1	Ready	<none>	85d	v1.30.3+k0s
kube-work-2	Ready	shits	15h	v1.30.6+k0s

```
kube-work-3    Ready    <none>    14h    v1.30.5+k0s
```

```
###
```

```
root@kube-work-4:~# cat down
```

```
https://github.com/containernetworking/plugins/releases/download/v1.3.0/cni-plugins-linux-amd64-v1.3.0.tgz
```

```
https://storage.googleapis.com/kubernetes-release/release/v1.30.1/bin/linux/amd64/kube-proxy
```

```
https://storage.googleapis.com/kubernetes-release/release/v1.30.1/bin/linux/amd64/kubelet
```

```
root@kube-work-4:~# wget -q --show-progress --https-only --timestamping -P downloads -i down
```

```
cni-plugins-linux-amd64-v1.3.0.tgz
```

```
100%[=====]  
=====>] 43.24M 21.5MB/s in 2.0s
```

```
kube-proxy
```

```
100%[=====]  
=====>] 54.91M 17.3MB/s in 3.2s
```

```
kubelet
```

```
100%[=====]  
=====>] 95.46M 22.5MB/s in 5.0s
```

```
apt install containerd -y
```

```
###
```

```
root@kube-work-4:~# systemctl cat kube.service
```

```
# /etc/systemd/system/kube.service
```

```
[Unit]
```

```
Description=Kubernetes Kubelet
```

```
Documentation=https://github.com/kubernetes/kubernetes
```

```
After=containerd.service
```

```
Requires=containerd.service
```

```
[Service]
```

```
ExecStart=/usr/local/bin/kubelet \
```

```
--config=/var/lib/kubelet/kubelet-config.yaml \
```

```
--kubeconfig=/var/lib/kubelet/kubeconfig \
```

```
--register-node=true \
```

```
--v=2
```

```
Restart=on-failure
```

RestartSec=5

[Install]

WantedBy=multi-user.target

###

```
root@kube-work-4:~# cat /var/lib/kubelet/kubelet-config.yaml
kind: KubeletConfiguration
apiVersion: kubelet.config.k8s.io/v1beta1
authentication:
  anonymous:
    enabled: false
  webhook:
    enabled: true
  x509:
    clientCAFile: "/var/lib/kubelet/ca.crt"
authorization:
  mode: Webhook
clusterDomain: "cluster.local"
clusterDNS:
  - "10.32.0.10"
cgroupDriver: systemd
containerRuntimeEndpoint: "unix:///var/run/containerd/containerd.sock"
podCIDR: "100.64.32.0/24"
resolvConf: "/etc/resolv.conf"
runtimeRequestTimeout: "15m"
tlsCertFile: "/var/lib/kubelet/kubelet.crt"
tlsPrivateKeyFile: "/var/lib/kubelet/kubelet.key"
```

###

```
kube-worker-4.{crt,key}->/var/lib/kubelet/kubelet.{crt,key}
```

###

```
root@kube-work-4:~# cat /var/lib/kubelet/kubeconfig
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: # ca base64
```

```
server: https://10.138.117.204:6443
name: kubernetes-the-hard-way
contexts:
- context:
  cluster: kubernetes-the-hard-way
  namespace: default
  user: system:node:kube-work-4
  name: default-context
current-context: default-context
kind: Config
preferences: {}
users:
- name: system:node:kube-work-4
  user:
    client-certificate-data: # base64
    client-key-data: # base64
```

```
###
```

```
systemctl start kube.service
```

```
root@kube-ctrl:~# kubectl get no -o wide
```

NAME	STATUS	ROLES	AGE	VERSION	INTERNAL-IP	EXTERNAL-IP	OS-
IMAGE				KERNEL-VERSION		CONTAINER-RUNTIME	
kube-work-1	Ready	<none>	85d	v1.30.3+k0s	10.138.117.150	<none>	Debian
GNU/Linux 12 (bookworm)		6.1.0-23-amd64				containerd://1.7.20	
kube-work-2	Ready	shits	16h	v1.30.6+k0s	10.13.37.3	<none>	Rocky Linux
9.4 (Blue Onyx)		5.14.0-427.42.1.el9_4.x86_64				containerd://1.7.22	
kube-work-3	Ready	<none>	14h	v1.30.5+k0s	10.138.117.79	<none>	AlmaLinux
9.4 (Seafoam Ocelot)		5.14.0-427.42.1.el9_4.x86_64				containerd://1.7.22	
kube-work-4	Ready	<none>	35s	v1.30.1	10.138.117.7	<none>	Ubuntu
20.04.6 LTS		5.4.0-200-generic				containerd://1.7.12	

???????? ???? ???? ???? ???? ?  
????

Предположим мы создали куб (on-premise или в облаке) и нам выдали админский кубконфиг, где авторизация происходит по сертификатам, например:

```
apiVersion: v1
clusters:
- cluster:
  certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSU0tLS0tCk1JSURBRENDQWVpZ0F3SUJBZ0lVTVhZMjhWMEJaQld2Ry9hQnNZdDJ5OV
prRmVRd0RRWUplb1pJaHZjTkFRRUwKQlFBd0dERVdNQLFHQTFVRUF4TU5hM1ZpWlhKdVpYUmxjeTFqWVRBZU3MHl0akEw
TURrd056TTBNREJhRncwegp0akEwTURZd056TTBNREJhTUJneEZqQVVCZ05WQkFNVERXdDFZbVZ5Ym1WMFpYTXRZMkV3Z2
dFaU1BMEdDU3FHC1NJYjNEUUVQVFVQUE0SUJEd0F3Z2dFS0FvSUJBURU0E9CZW9wN1VyektEbnpjQzRUNFNjZWVKZndG
VFkvZWsKNXZQZjIhIw1FWUWhzRzg5d1MrZEdJdkRZK2ZmMjd0YjZCVUs3ajdGd1Jta3R0SFpNSnNuUEVvUeZvdWp6YlKxSQ
o2SuhQVEY2TXE2VVJnU0NGSwTEVWRrd3JTZ0xQSF1KK3Bic0hNSFF2bW81ZFRQV1dCVz1VeVpxSnFzeW80WjBvCjVaS0VH
c01reUFkVFE1b1J4d2Z3MkMwb1YyZ2N2dWxSdWl1aUtCVXBYRTM4ZGkrEz6WkNCaE3NUQ2UU90cXgKUjlydHZte1RmRm
JETlo3WkdEZlM4cFN4RTBwUTHQZEF5M2pjQVpacGpueGZvYldueTdYRGVHOFJkQktzMTJMWApFK3RuSE9HRVB3ck5hb1Q1
MTg4R3R3a095dGxvTEgyanlwUjJtT05FMTJxZlMrVzRnR2hsQWdNQkFBR2pRakJBCK1BNEdBmVvKRHdFQI93UUVBd0lCQm
pBUEJnTlZlUk1CQWY4RUJUURBUUgvTUIwR0ExVWREZ1FXQkJRvG1NbkYKWDhkUWhQYWdzZ2VXRWJNYXpnRDdrVEF0Qmdr
cWhraUc5dzBCQVFzRkFBT0NBUEVBYTVjOWl1YUdGNdiUGcWdApBZlRFRXURbEZXMxlVnRU4zWlU4T0lHdjIzMWh3ekRDVV
dmYXlN0TkrcS9hbn1UbU5YMWZ4eExtMGNDMk9lRk1JXCNm2bWc2STF1SFJr0TdJRu1FNGZkNF1hNVVo0GR0NTh0Vld3T3JS
S1hoUFFkZWNTVGVISnM2MzBtcktsa1k2aTcKZTZFMHk0UU80cjBRVzN1N1hTMDNZSEpZc0duVCtmYVdSSTBmcEoyWWQ40T
A4SS9FbzBmMDR6bjlmYjZ0VE5wMApw0HJUeDNpa1lhek1GNk1seHI10FJoRmcxa1k5dWZEM1NsdKN5Q1F4Q2hDMUZKe1BQ
aVNMctZFNkt2VFloTHVzCkNTbw54WnRtMkh6Z25BWGE5aVRsN3krNDNQZDM3bEZ2VTJlQnRMczY0YjIrlZn1bj1KL2wwWi
twVUJkVzhxenMKNXdVb0xnPT0KLS0tLS1FTkQgQ0VSVElGSUNBVEUtLS0tLQo=
  server: https://10.13.37.199:6443
  name: local
contexts:
- context:
  cluster: local
  user: user
  name: Default
current-context: Default
kind: Config
users:
```

- name: user

user:

client-certificate-data:

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSU0tLS0tCk1JSURXVENDQWtHZ0F3SUJBZ0lVnNkV283dkZQ0Hd2N1kwWXJmVGwrY2  
tBeEdrd0RRWUpLb1pJaHJzTkFRRUwKQlFBd0dERVdNQLFHQTFVRUF4TU5hM1ZpWlhKdVpYUmxjeTFqWVRBZU3MHL0akEw  
TURrd056TTBNREJhRncweQp0ekEwTURrd056TTBNREJhTURReEZ6QVZCZ05WQkFvVERuTjVjM1JsYlRwdFlYtjBaWEp6TV  
Jrd0Z3WURWUWFECkV4QnJkV0psY201bGRHVnpMV0ZrYltdU1JSUJJakF0QmdrcWhraUc5dzBCQVFFRkFBT0NBUThtBTU1J  
QkNnS0MKQVFFQXg1MmtpTHJtd1RHcG5YTXd5NStldnR0MzFtMzRBEhtVGky0E1xcjN5YTVRWUpEK1hYWewrSzLZMForNQ  
pXSjhESlhUSzJaendvenJIZFZjUXZpNm9iSDNFUudhdGtSQXdiVkfVek900G1lbzYzdUNaaU9GRHhRdWpBZ3l6Cm1u0W1t  
cTNobi9STHVEWEV1LzlyWfHyZGduYm14SmF4ZXNoQ2U2ZUpstSS8rYkNYaDVyWgtxRXlpUFVtck5zcmgKZ2xiL1Jp0G5FME  
RRaDRsZG9j0HpjNEtJVnFHT1JJQmhtWXMzMmRuWnduSstHYTLWbTRMQXRxdmLPd0RhMk9zZQppqSXBMT1FhSStUNkLHUGh5  
blpTU1pUVjFYamZlZlIRWxQSUVENFlMeWIrRnY2bjQ5WGN4NDA2UU9veFB6ek9zCkNGc0px0ENvM0s5eUlINnJkCvJtYm  
pSYVh3SURBUUFcbzM4d2ZUQU9CZ05WSFE4QkFm0EVCQU1DQmFbD0hRWUQKVLlWbEJCWXdGQVlJS3dZQkJRVUhd0VHQ0Nz  
R0FRVUZCd01DTUF3R0ExVWRfd0VCL3dRQ01BQXdIUUVLEVlIwTwpCQlFRk12NnRlejMzaLFdTS85UkM3YkMrRGtNNm5JeU  
1COEdBMVVkSXdRWU1CYUFGQk9JewNwZngxQ0U5cUN5CkI1WVJzeHJPQVB1Uk1BMEduD3FHU0liM0RRRUJDd1VBQTRJQkFR  
Q1UwRnNueVpPcG50ZFY1TEtsRTRF0C8rTXMKd1R6S1FJUkk1RUZMZ0JicXBhUzJmYTLNL3pDUElua1U2Yk53RTZTWE80Ym  
t0RmkyWjRzSSt2aDjHS1VKWwNlNwpR0Fk4YmNxTVoxSWI5Smo1TXNkekJCOFV5YmZ1d2hCNFB6Wwc5c3L3MHNbWcTl3Nr  
UEt6YlBUNFY1WUdnRFhiCm9ra2or0Wo3U05IMHniTkdwTjJGV2dCRisvNlNZaFJEClpZdlZVWmsybzJNUK9zRXFhcUlhej  
NWQ25WUWYzZmkKRvVtV1dDdFg2VER6ejFEVm44M0LYQ2ErcwLvb3dEYVpGNTMwbFRPSHVnb0NHWGdtcnQ3ZnFpR3FzNFF0  
T2RVbgpW0UhaSURFQ0xsNXJvRVoxdDA1Smw2awdrVkt3dUdzZkRtaTN0V0xRL05od1lSSkNSVDUrbEN6UDNweUQKLS0tLS  
1FTkQgQ0VSVELGSUNBVEUtLS0tLQo=

client-key-data:

LS0tLS1CRUdJTiBSU0EgUUFJJVkJFURSBURVktLS0tLQpNSU1Fb3dJQkFBS0NBuUUVBeDUya2lMcm13VEdwblhNd3k1K2V2dH  
QzMW0zNEFMSG1UaTI4TXFyM3lhNVFZSkQrClhYWewrSzLZMForNVdK0ERKWFRLMlp6d296ckhkVmNRdmk2b2JIM0VRR2F0  
a1JBd2JWQW96T3Q4aXVvNjN1Q1oKaU9GRHhRdWpBZ3l6bW45bW1xM2huL1JMdURZUTUvOXJYWFhkZ25iaXhKYXhlc2hdZT  
ZlSm1JLytiQ1hoNXJYawpxRXlpUFVtck5zcmhnbGIvUmk4bkUwRFFoNGxkb2M4emM0S0lWcUdPUklCaG1ZczMyZG5ad25J  
K0dh0VZtNExBcNRxdmLPd0RhMk9zZwpJcExPUWFJK1Q2SudQaHluWlNTWlRWMVhqZnVkeUhfBfBJRUQ0WUx5YitGdjZuND  
lYY3gKNDA2UU9veFB6ek9zQ0ZzSnE4Q28zSzl5SUg2cmRxUm1ialJhWhdJREFRQUJBB0lCQUFXQmgvWC9pUGxNb2VKZwpC  
WVlKN2NxR2c4K1pDKzE3WVNBdXVdc2JuU2NZNGVRV1pWd295dkUzWjEyczIzMElaUUZTbGVNQldiTVNIa200Ci9TeTg5dT  
I3S2ZLN20ra0FoMm82NFlCc3RHdzdUU1NhZDVHeEZjYkNVNE1HM0hhVC90YzZBQUcxKzZSTU8rRlkKdTFxQ0dlQ05FeVFD  
Q1VsbjBiR2pxN0tqc2LYMjN3Zld2RkJRZzBLSlR1TnEycVhqYU5YQ2RyU2NqNStLYVB0RwpzVGhCU2JVYjVvMDVrcXRsd3  
A5U1NoenZzeHBVWFBoTmNCYjVQRlZmSG12L3F3NzB5L0cwQmhoUGVxv050RmJVCKyZL1k5aEJrMVhtZ1NNVmNIVFUrT3ZW  
QXNjVm10VnBvVvhHq2VHNEE0MWRCK1lWb24zZnE5Vkp1RzFuR0MwU3gKbDhUdXZmRUNnWUvBMGRzMLZwWmtFUHVrR05RdU  
NVSjBocGVQV0R0S0ThSN3lNY1hvNVJorS9NWghvR3lGUFFPMQphUkxpeS90MFVJNFFkUjVYSWQ4RTgyQ0lYQ0x5azN0RUN3  
bHVBUzhQc2lJVHg4bHRjBERER25ZTlBVWEZzK2kwClQ5SmFOMFNpL2o0Q2lkTjUrYm1VNna0Q3h1bXNvZzEzTG5zTUhVU  
i8rM1pPaXBoazEzaFQxaEVDZ1lFQtg0SCsKVmpIM2VcV1E2Nj1WbmrKsfU4dWdvbUweFNxekRv0TkxaE1ydFBjNUlVWnd0  
ODZNWUhYcWtWRS9KcUI0cXp0YwpNaXh4MG9QU2hYm2taYlNpcENjblz4d09HYU5uTWprNFY0Z0M0eDYwRzJIcFY1THNFUm  
EzaU50UjhaUDdhMTBQCnVMQ3JaUTFmNit3bkowd0hzRHR6UFlYaWJvcmxJd1Z2dHBoaytX0ENnWUvBbWfXrkdfcE9BmZBx  
ZlI5RwdYVU4KZWNtWg93R2M2eUE4TlNMd3pHTkRoMHFlVW1XQis5VXVTanNRb0VaL3Q4YjcxN1FhR1d0KzVXNGxDRWh5RU  
hicwpyNlA4elpNV1M5YlZZcTVnbXBUMDgVZkE0NzZrN0g4UkhXd21yMVpxZS9rTXhMcDRFTlhHYVN50VhjT1Nxs2R4ClBp

```
L0xB0WEyV1hjYU5ERTFpK2pHZ1BFQ2dZQk8vRm10aUFPbmxnYytD0CtQdFdiVGpYZDdkUEpxbmLFYWxmeLIKVmNLV5RUM
ZBTVFodGdQZXZpRHFqZWFZRnZGTlNhSHNQSEpuUUK4bThlRUdCSVBGRDFiQVLQ0UozYkQ1bjRuaApDcU0xSEo1N1MxVXZM
TjhaNCs2QW0vT0drdu80dmFU0TJZQ2U5S21xa3gxWUo5ZE9tTm9XbUxrTGpvr294MGdIClNJTM9UUUtCZ0Q3SWV3MnBJeT
FzRGg4bUxUbGJQS294ejhCWnZnK2RhMmp0VjN5enp1ZHRsbkNXL0o3MWL4MVIKZU1TbFpXcGRZR09NYWltUU10Mnp2QjdG
UCtPNnpxaE16SldIL0Y2dm92c2lpWExsQmJpcmNTUnBYyVZy0ERZdAo2di9XRHFMRlh1b0tmKyttM1FkZzhlATY0MmZsWG
J0aVBoWVRwdlhLSTl6dU5hdXFLV0lnCi0tLS0tRU5EIFJTQSBQUklWQVRFIEtFWS0tLS0tCg==
```

Мы радостно начинаем раздавать этот кубконфиг всем сотрудникам нашей дружной организации и всем хорошо. Но всем хорошо до того момента как кто-то уволится и заберет с собой этот кубконфиг.

Возникает вопрос - как отозвать у уволившегося сотрудника данный сертификат. Сейчас уже никак, потому что клиентский сертификат выпускается от корневого сертификата кластера и действует 1 год (обычно), а kubernetes не имеет встроенных механизмов отзыва сертификатов.

Поэтому если с кубом работает больше одного человека - лучше заранее научиться управлять пользователями в кубе.

???????? ???? ???????????

В кубе есть несколько механизмов ([тык](#)) для авторизации. То что мы имеем на руках - "X.509 client certificates", о его минусах я писал выше. Из того что (как мне кажется) больше всего подойдет для живых пользователей: "[Service account tokens](#)" и "[OpenID Connect Tokens](#)". По второму можно найти инструкции в интернете, а по SA Tokens инфы довольно мало, о нем и пойдет речь.

Авторизация по SA токенам удобна тем что не требуется внешний OpenID сервер и аккаунт вместе с токен(ами) можно отозвать в одну команду. Однако лучше всего все таки настроить OpenID сервер, потому что он выдает короткоживущие токены, что безопаснее.

За основу возьмем конфиг выше. Нам надо будет создать SA в кубе, повесить на него ClusterRoleBinding (можно более точно выдать роли, но ради упрощения выдадим "cluster-admin"), сгенерировать токен и сформировать кубконфиг.

Создаем пользователя, выдаем ему доступы, генерируем временный токен:

```
# Создаем аккаунт (можно указать неймспейс)
[root@k8s-istio ~]# kubectl create serviceaccount ivan
serviceaccount/ivan created
# Вешаем clusterrole на Ивана
[root@k8s-istio ~]# kubectl create clusterrolebinding ivan-access-full-cluster --
clusterrole=cluster-admin --serviceaccount=default:ivan
```



```
2zvzk1bGUF52kwohavmG0Hw3SCfG7Leh52mbfB8HmeJkCMcZK028DREvh9GiYrdoa0C4gtzHF14vsBmdsL7jVmh67at0YJ
y0zBeiS9ioc85_faaQkwth7LoxVAhBJqdsXc6Zj593-
oezG89wBujW6Ty4YT0i0yKiSu1jUrvkCPGnc012Rzd5pjUCna1HNy7cZcrD-
5dT28xBJttELFsZ9xGKPY9xW5FaI2r2K9S_FpdNCXoqfzCoVruED9If8yKkoerKYNrAsaE25gtTjnRJBhKaX4a0XdGKm3y
X6fC32d7nKDEjyiXsmCQ8L4U21iYSqYnxMBZnhNg
```

Теперь необходимо сформировать с этим токеном кубконфиг:

```
# Удаляем стандартного юзера
[root@k8s-istio ~]# kubectl config delete-user user
deleted user user from ivan.kubeconfig

# Удаляем стандартный контекст
[root@k8s-istio ~]# kubectl config delete-context Default
warning: this removed your active context, use "kubectl config use-context" to select a
different one
deleted context Default from ivan.kubeconfig

# Прописываем в конфиге пользователя и его токен
[root@k8s-istio ~]# kubectl config set-credentials ivan --token=$TOKEN
User "ivan" set.

# Создаем контекст и переключаемся на него
[root@k8s-istio ~]# kubectl config set-context Default --cluster=local --user=ivan
Context "Default" created.

[root@k8s-istio ~]# kubectl config use-context Default
Switched to context "Default".

# Проверяем
[root@k8s-istio ~]# kubectl get no
NAME          STATUS    ROLES          AGE    VERSION
k8s-istio    Ready    control-plane  55m    v1.34.6+k0s
```

Все работает!

Если Иван уволится, мы можем в одну команду отозвать его кубконфиг:

```
# Все работает
[root@k8s-istio ~]# kubectl get no
```

```
NAME          STATUS    ROLES          AGE    VERSION
k8s-istio    Ready    control-plane  59m    v1.34.6+k0s
```

```
# Удаляем Ивана
```

```
[root@k8s-istio ~]# kubectl delete serviceaccount ivan
serviceaccount "ivan" deleted from default namespace
```

```
# Токены кешируются - отзыв происходит в течении минуты, но не сразу
```

```
[root@k8s-istio ~]# kubectl get no
```

```
NAME          STATUS    ROLES          AGE    VERSION
k8s-istio    Ready    control-plane  59m    v1.34.6+k0s
```

```
[root@k8s-istio ~]# kubectl get no
```

```
NAME          STATUS    ROLES          AGE    VERSION
k8s-istio    Ready    control-plane  59m    v1.34.6+k0s
```

```
[root@k8s-istio ~]# kubectl get no
```

```
error: You must be logged in to the server (Unauthorized)
```

Для понимания, вот так вот будет выглядеть кубконфиг с авторизацией по токену:

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURSB0tLS0tCk1JSURBRENDQWVpZ0F3SUJBZ0lVTVhZMjhWMEJaQld2Ry9hQnNZdDJ5OV
prRmVRd0RRWUplb1pJaHJjTkFRRUwKQlFBd0dERVdNQlFHQTFRUF4TU5hM1ZpWlhKdVpYUmxjeTFqWVRBZUZ3MHl0akEw
TURrd056TTBnREJhRncwegp0akEwTURZd056TTBnREJhTUJneEZqQVVCZ05WQkFNVERXdDFZbVZ5Ym1WMFpYTXRZMkV3Z2
dFaU1BMEdDU3FHC1NJYjNEUUVQVQVQUE0SUJEd0F3Z2dFS0FvSUJBUURY0E9CZW9wN1VyeKtEbnpjQzRUNFNjZWVKZndG
VFkvZWskNXZQZjhhIwlfWUWhzRzg5d1MrZEdJdkRZK2ZmMjd0YjZCVUs3ajdGd1Jta3R0SFpNSnNuUEVvSEZvdWp6YlksSQ
o2SuhQVEY2TXE2VVJnU0NGSwTtEVWRrd3JTZ0xQSF1KK3Bic0hNSFF2bW81ZFRQV1dCVz1VeVpxSnFzew80WjBvCjVaS0VH
c01reUFkVFE1b1J4d2Z3MkMwb1YyZ2N2dWxSdWl1aUtCVXBYRTM4ZGkrceEZ6WkNCaEp3NUQ2UU90cXgKUjlydHZte1RmRm
JETlo3WkdEZlM4cFN4RTBwUTHQZEF5M2pjQVpacGpueGZvYldueTdYRGVHOFJkQktzMTJmWApFK3RuSE9HRVB3ck5hb1Q1
MTg4R3R3a095dGxvTEgyanlwUjJtT05FMTJxZlMrVzRnR2hsQWdnNQkFBR2pRakJBCK1BNEdBmVvKRHdFQi93UUUVBd0lCQm
pBUEJnTlZiUk1CQWY4RUJUURBUUgvtUIwR0ExVWREZ1FXQkJRvGlnbkyKWdhkUWhQYwdzZ2VXRWJNYXpnRDdrVEF0Qmdr
cWhraUc5dzBCQVfzRkFBT0NBuUVBYTVj0Wl1YUdGNdiUGcWdApBZlRFRXUrbEZXMxlVRU4zWlU4T0lHdjIzMWh3ekRDVV
dmYXln0TkrCS9hbnlUbU5YMWZ4eExtMGNDMk9lRk1JXCNm2bWc2STF1SFJR0TdJRUIFNGZkNF1hNVVo0GR0NTh0Vld3T3JS
S1hoUFFkZWNTVGVISnM2MzBtcktsa1k2aTcKZTZFMHk0UU80cjBRVzN1N1hTMDNZSEpZc0duVCtmYVdSSTBmcEoyWWQ40T
```



# cilium-dbg

Мне некуда это записать

```
kubectl -n kube-system exec ds/cilium -- cilium-dbg service list
```

```
helm upgrade --reuse-values --set devices=eth1 --set hostFirewall.enabled=false cilium  
cilium/cilium -n kube-system --set envoy.enabled=true --set kubeProxyReplacement=true --set  
k8sServicePort=6443
```

```
helm repo add cilium https://helm.cilium.io/ ; helm upgrade --reuse-values --set devices=eth1  
cilium cilium/cilium -n kube-system --set envoy.enabled=true
```

```
kubectl -n kube-system exec ds/cilium -- cilium-dbg status
```

```
kubectl -n kube-system exec ds/cilium -- cilium-dbg service list
```