
OTUS
ОНЛАЙН-ОБРАЗОВАНИЕ
Онлайн-образование

Меня хорошо видно && слышно?

Ставьте + , если все хорошо

Напишите в чат, если есть проблемы

заодно проверяем, включена ли запись занятия

Включил Юджин запись ли ты

Преподаватель

Непомнящий Евгений

- 15 лет программировал контроллеры на C++ и руководил отделом разработки
- 5 лет пишу на Java
- Последнее время пишу микросервисы на Java/Kotlin

Введение в Kubernetes. Часть 1

Непомнящий Евгений

telegram @EvgeniyN

Правила вебинара

Активно участвуем

Задаем вопрос в чат

Вопросы вижу в чате, могу ответить не сразу

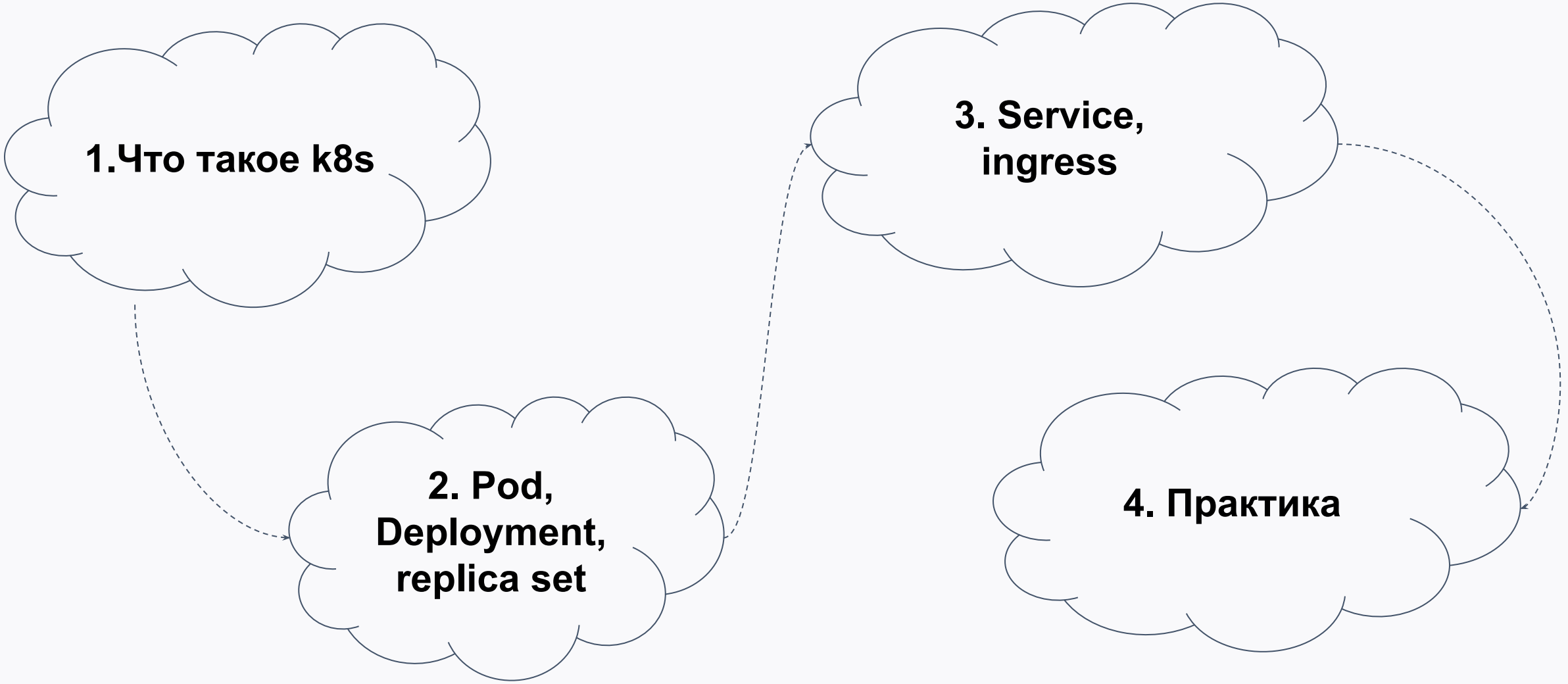
Цели вебинара | После занятия вы сможете

1 Иметь представление о Kubernetes

Смысл | Зачем вам это уметь, в результате:

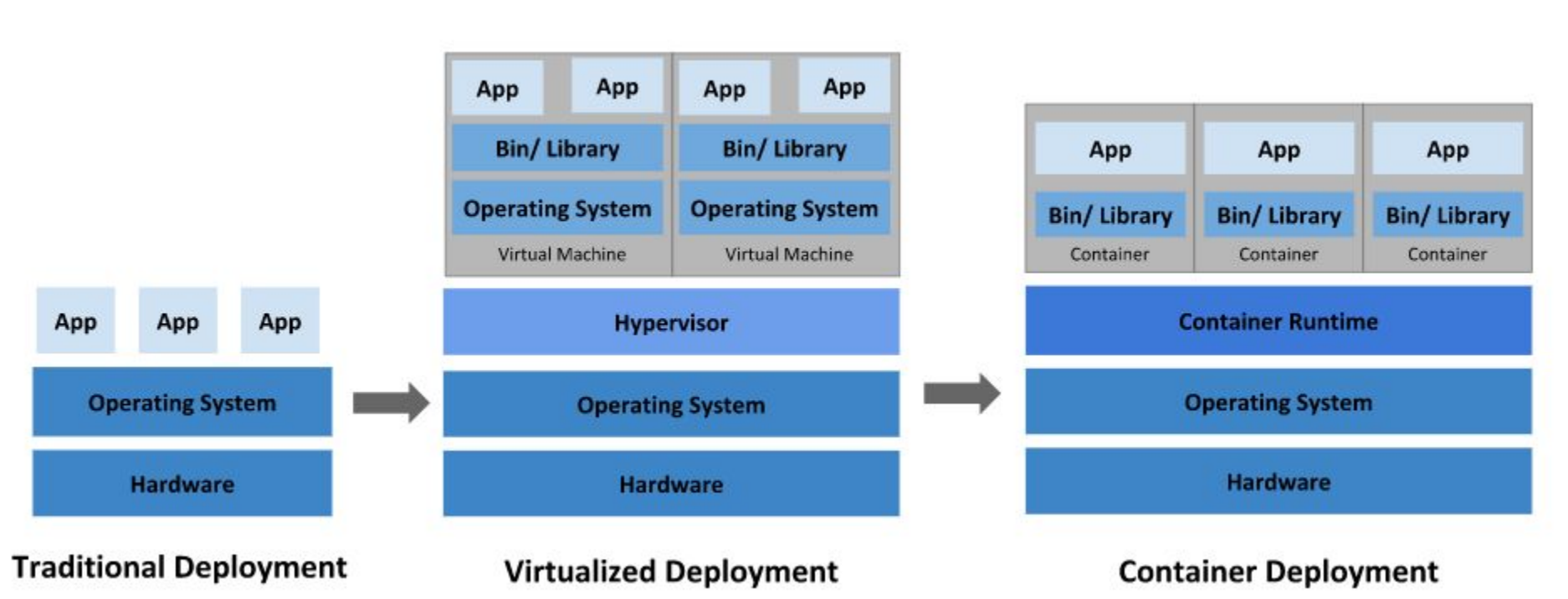
1 Уметь развернуть минимальное приложение и обеспечить доступ к нему извне кластера

Маршрут вебинара




Kubernetes

Kubernetes. Back in Time



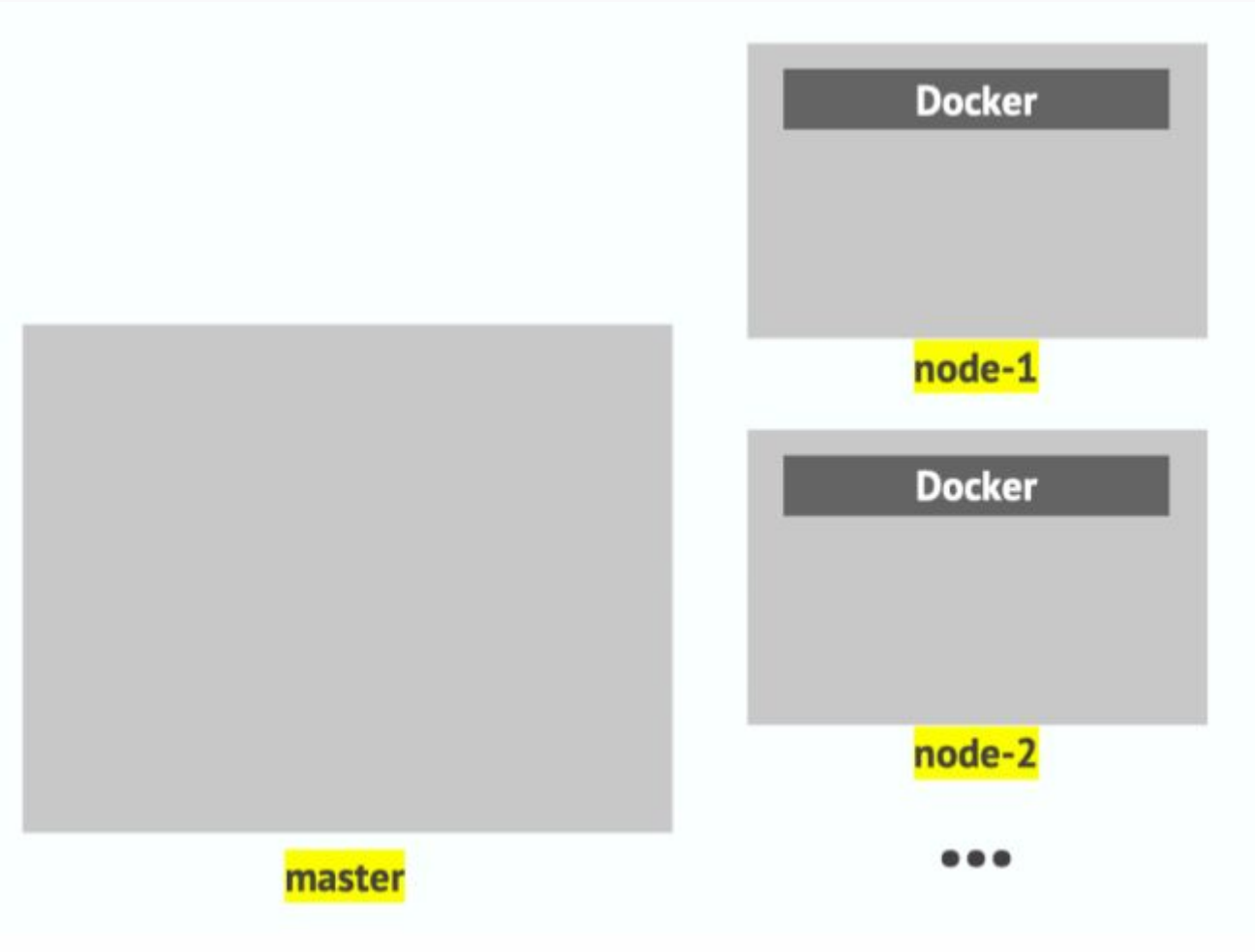
Kubernetes

Kubernetes – это оркестратор+, который решает следующие задачи:
• Service discovery и балансинг
• Управление хранилищами
• Деплой новых версий и откат
• Распределение контейнеров по нодам
• Управление конфигурациями
• Self-healing

Kubernetes **не решает** (из коробки) такие задачи:
• Из коробки нет логирования и мониторинга приложений
• Из коробки нет предоставляет инфраструктурных сервисов (DB, MQ, Storage и т.д)
• Не предоставляет CI/CD

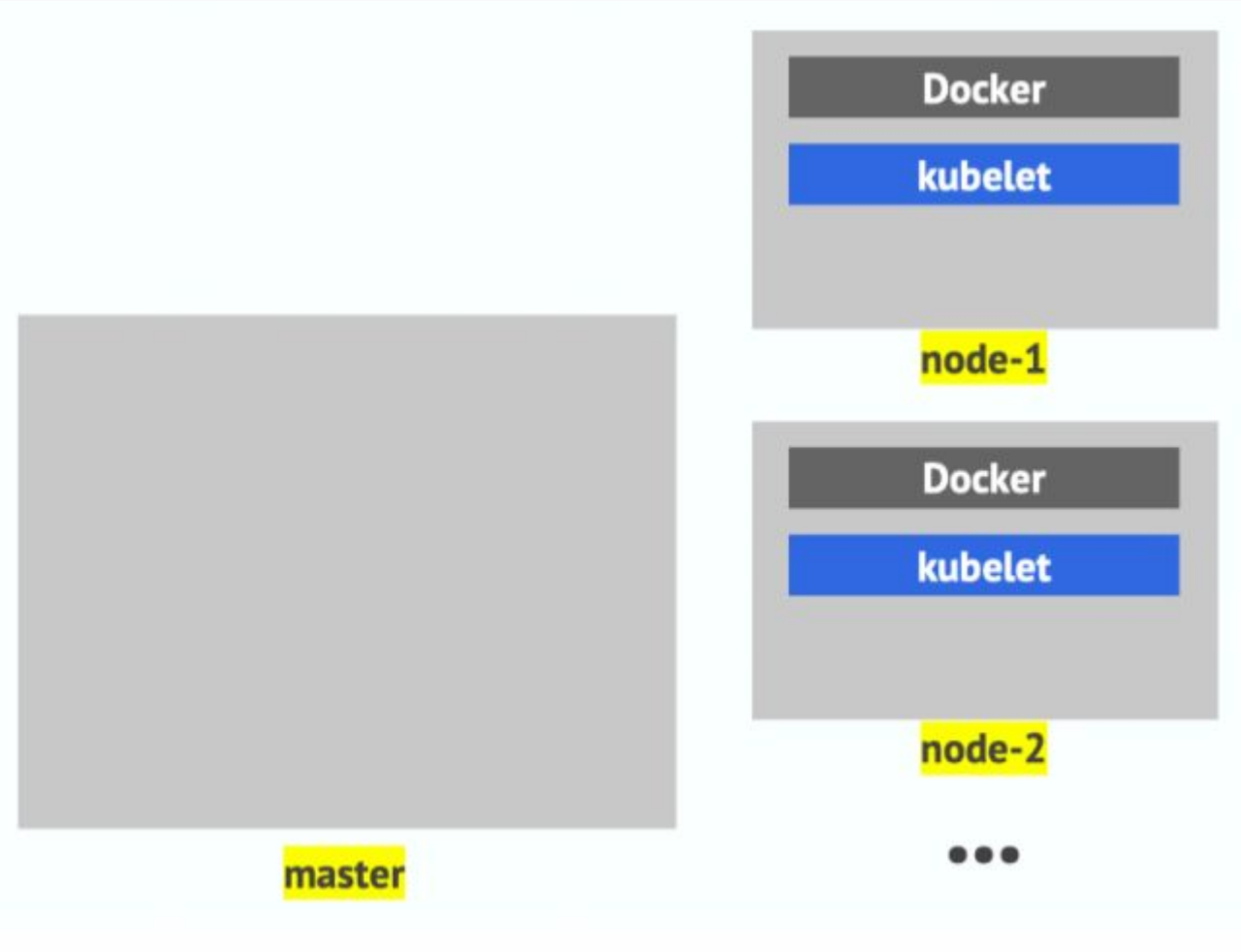
https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Узлы - контейнеризация



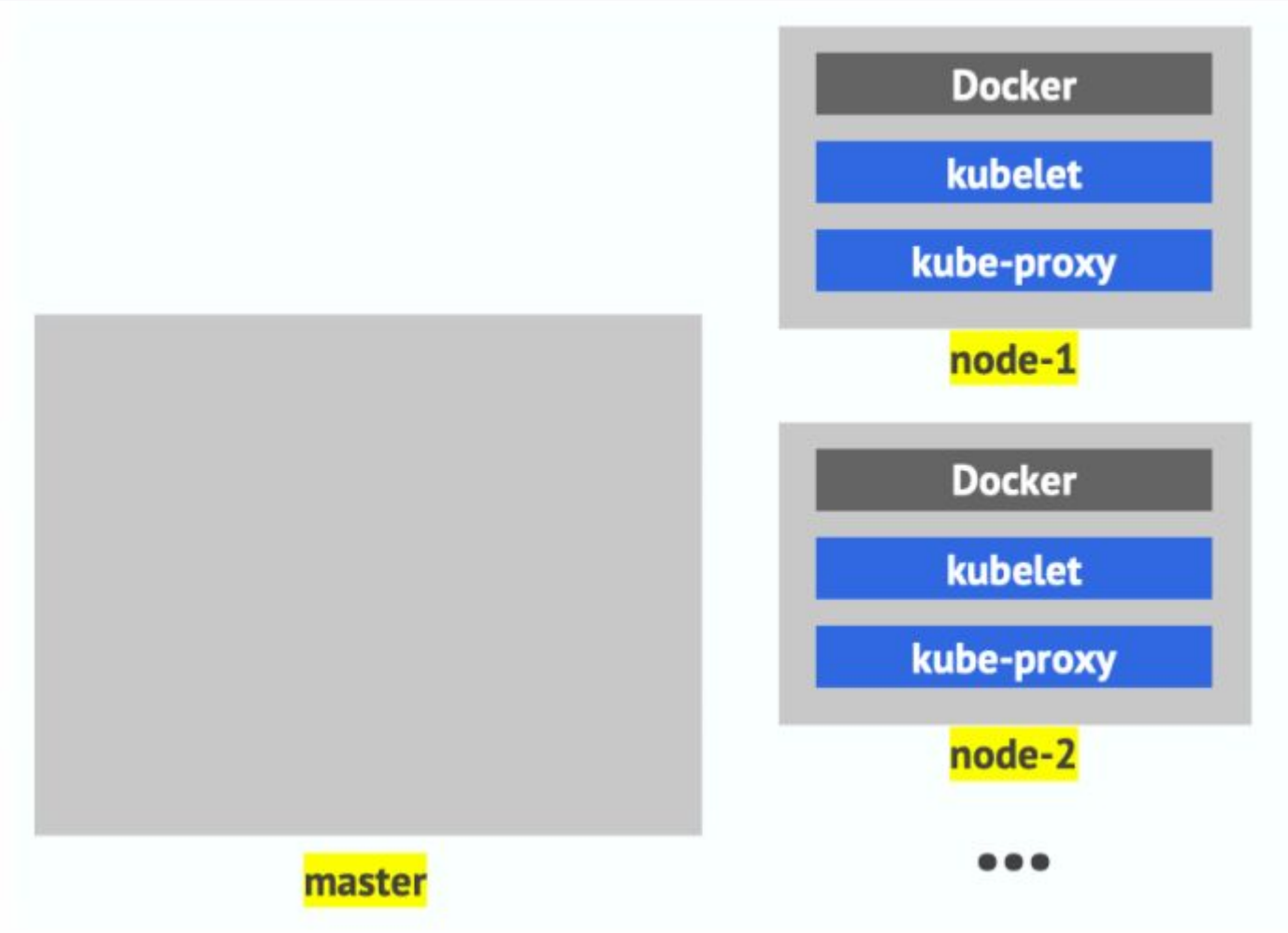
Чтобы запускать на рабочих нодах приложения, необходимо наличие какой-либо системы контейнеризации на каждой из этих нод.
Например, это может быть Docker. Но может быть и не docker, главное, чтобы удовлетворял CRI
(Container Runtime Interface)

Узлы - kubelet



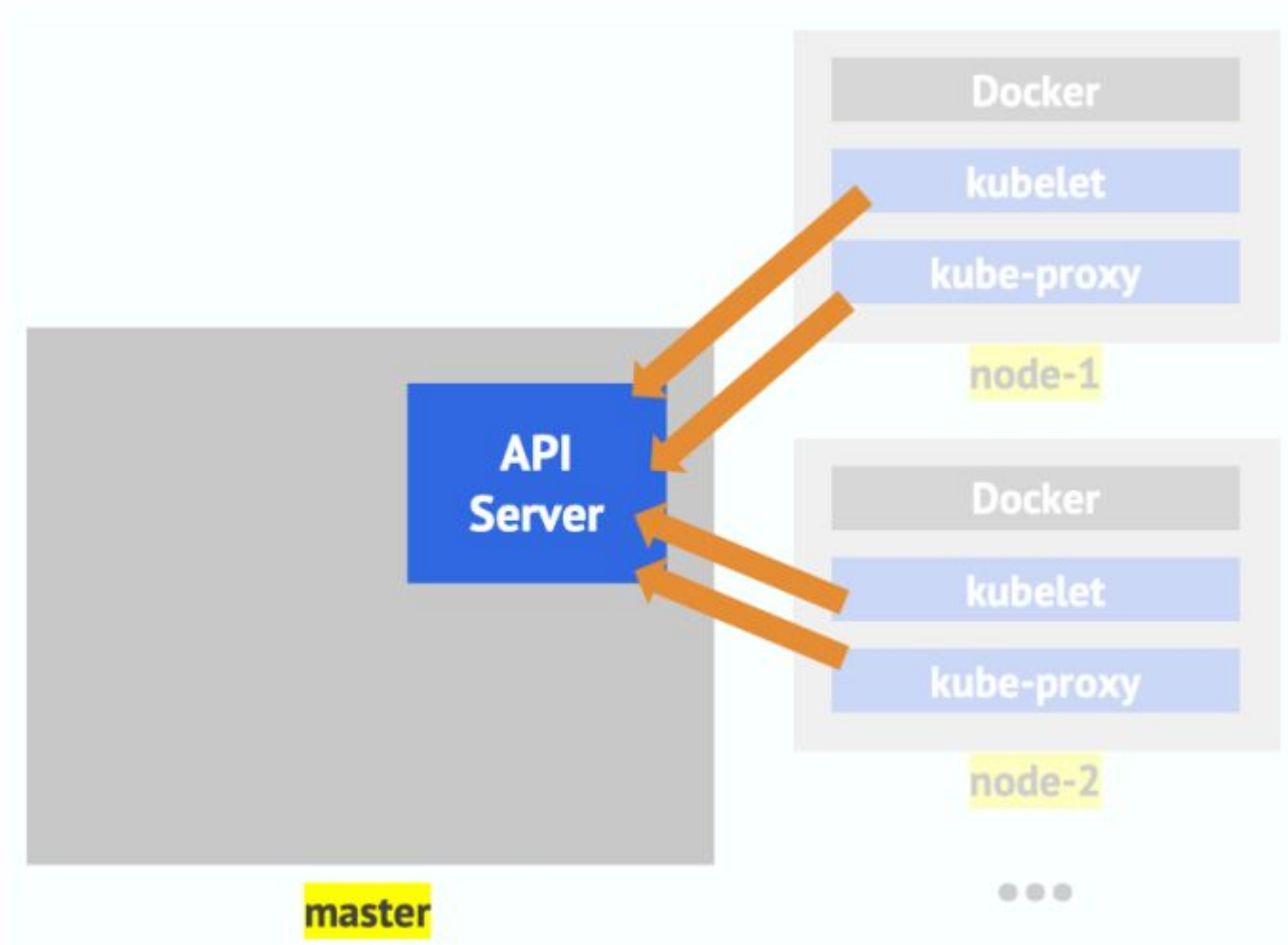
Для того, чтобы работать с контейнерами в реальном времени на каждой ноде должен быть установлен некоторый агент, который бы следил за живостью контейнера, запускал новые контейнеры, ограничивал контейнеры по ресурсам и т.д.
В рамках kubernetes-а, таким агентом является kubelet

Узлы - kube-proxy



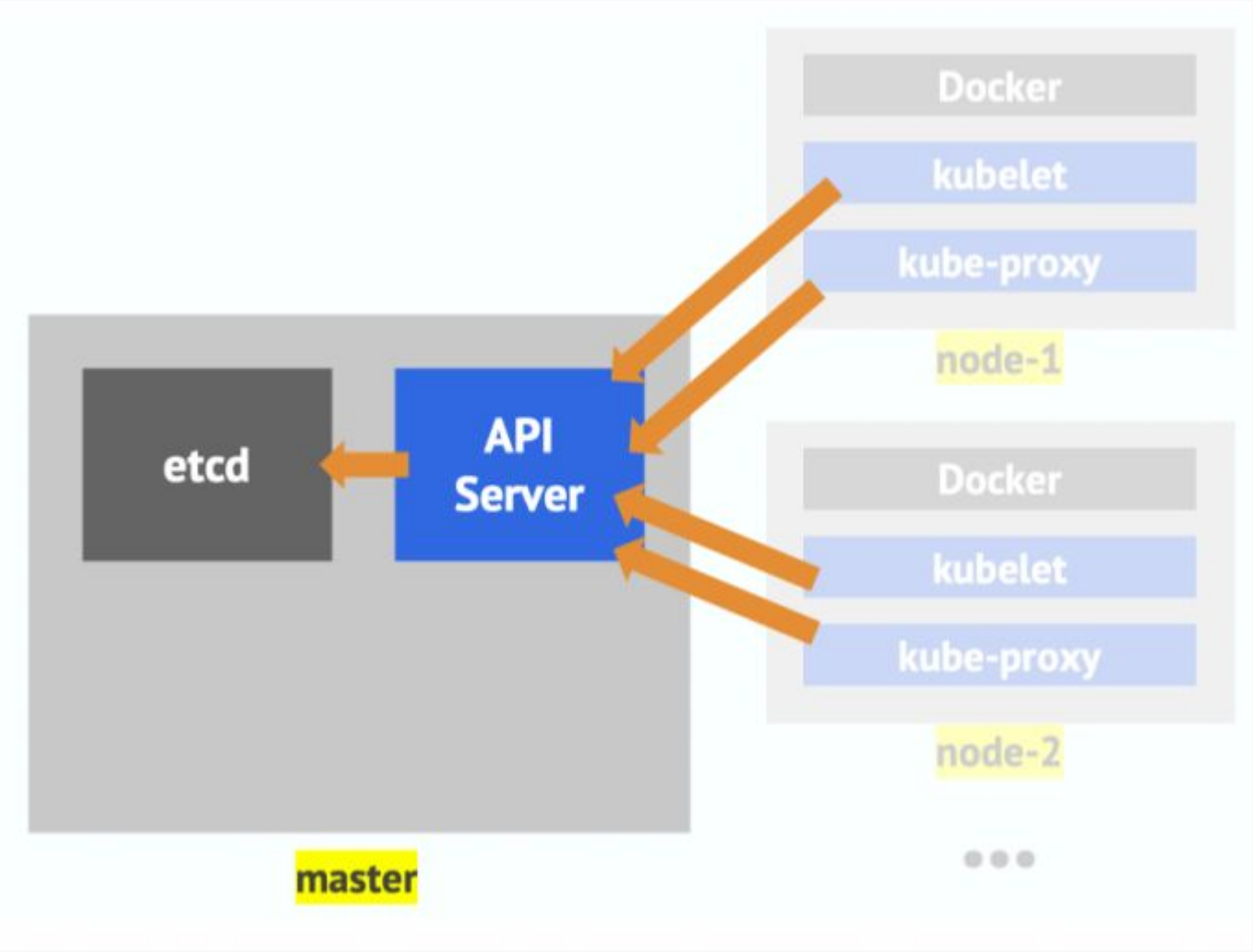
Для того, чтобы обеспечивать балансировку и service discovery, на каждой ноде необходимо прописывать соответствующие правила в сетевом файерволе (iptables, ipfw).
Агентом, который занимается этим на всех нодах, является kube-proxy

Мастер - API server



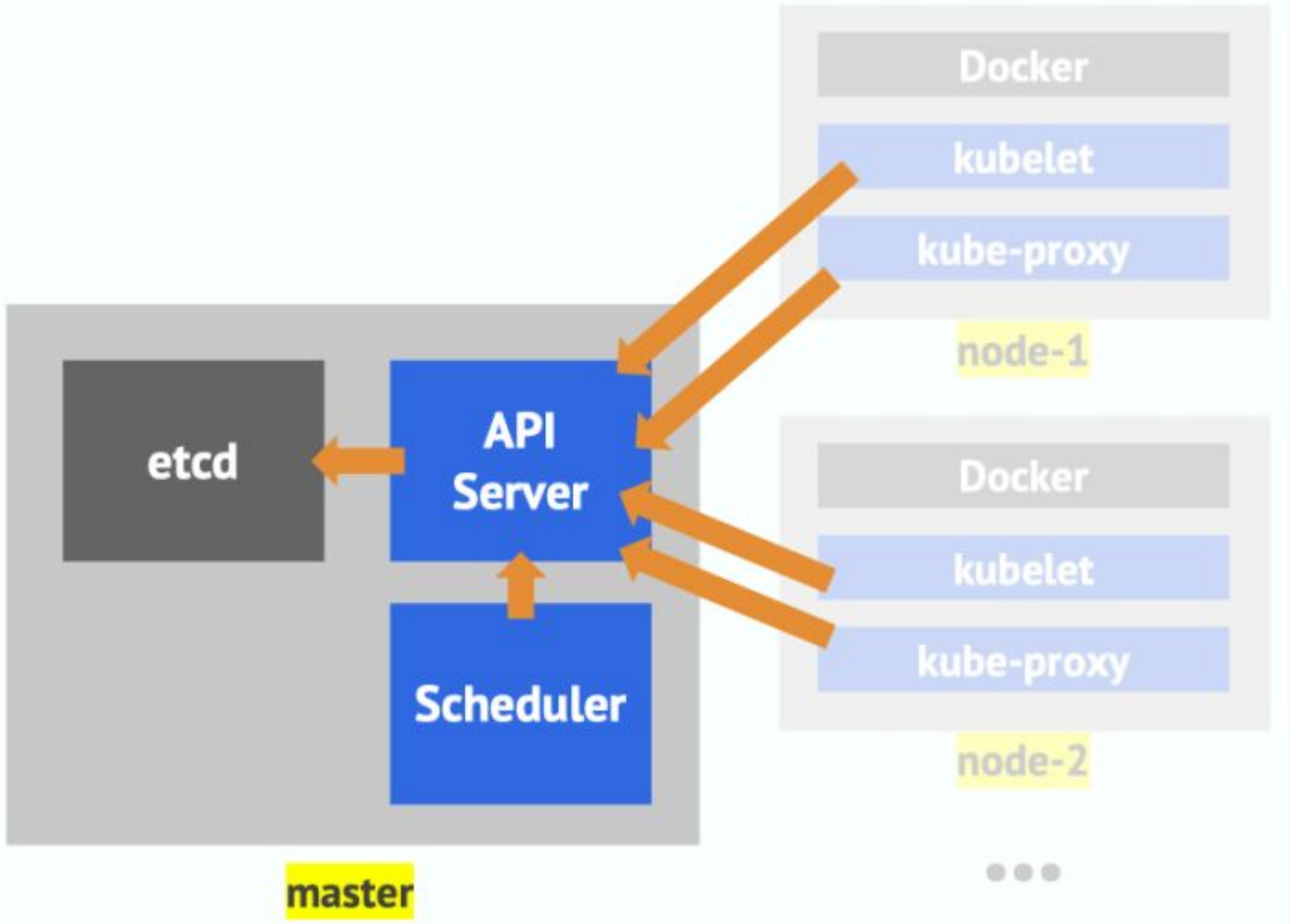
Для того, чтобы управлять агентами, которые расположены на рабочих нодах, нам необходим сервис, который по API будет с ним всеми общаться. Внутри kubernetes-а такой компонент называется API Server.

Мастер - etcd



Для того, чтобы где-то хранить текущую конфигурацию кластера, необходимо хранилище (отказоустойчивое, децентрализованное и консистентное). В kubernetes таким хранилищем является **etcd**.

Мастер - scheduler

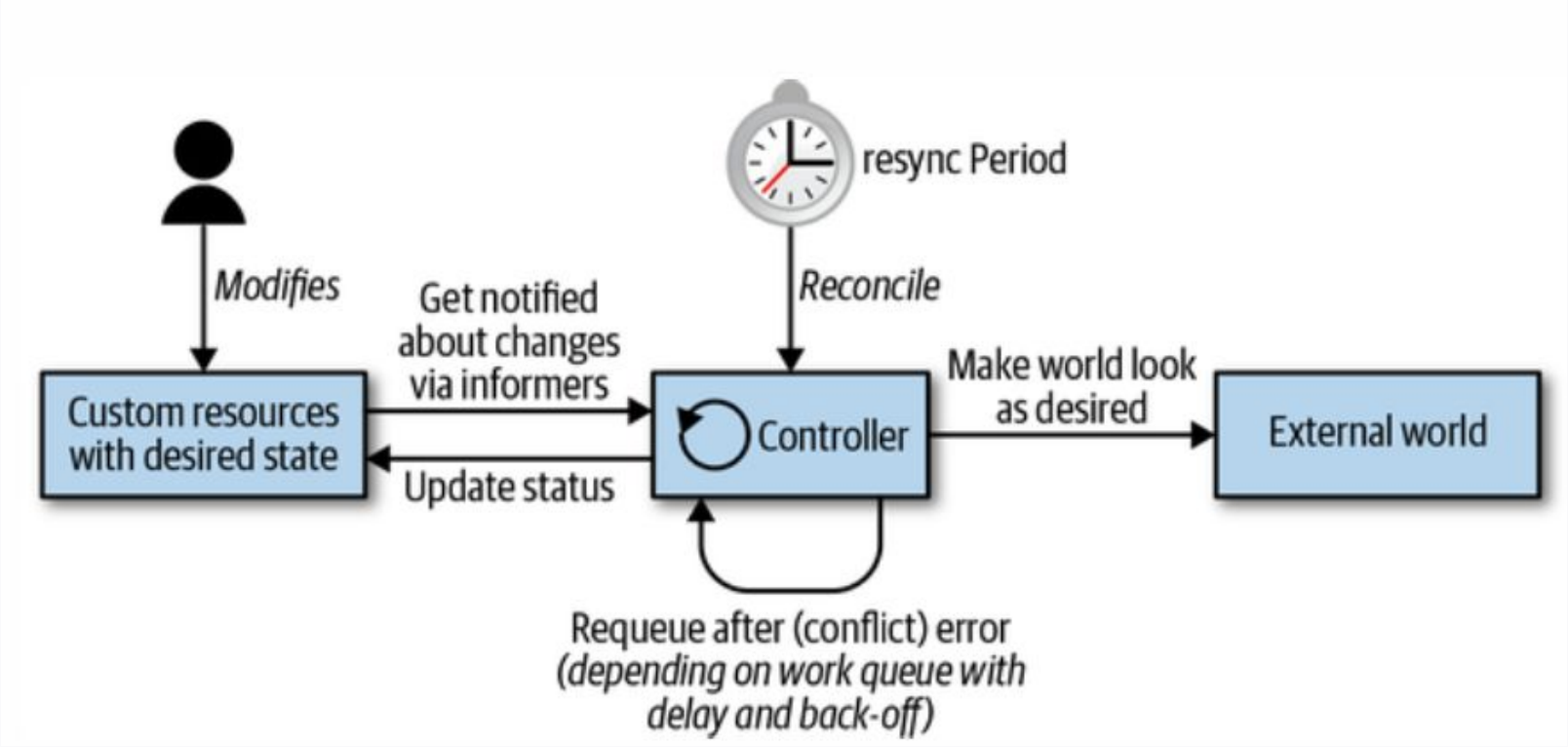


Так же есть отдельный компонент – **scheduler**, который отвечает за распределение рабочей нагрузки по нодам.

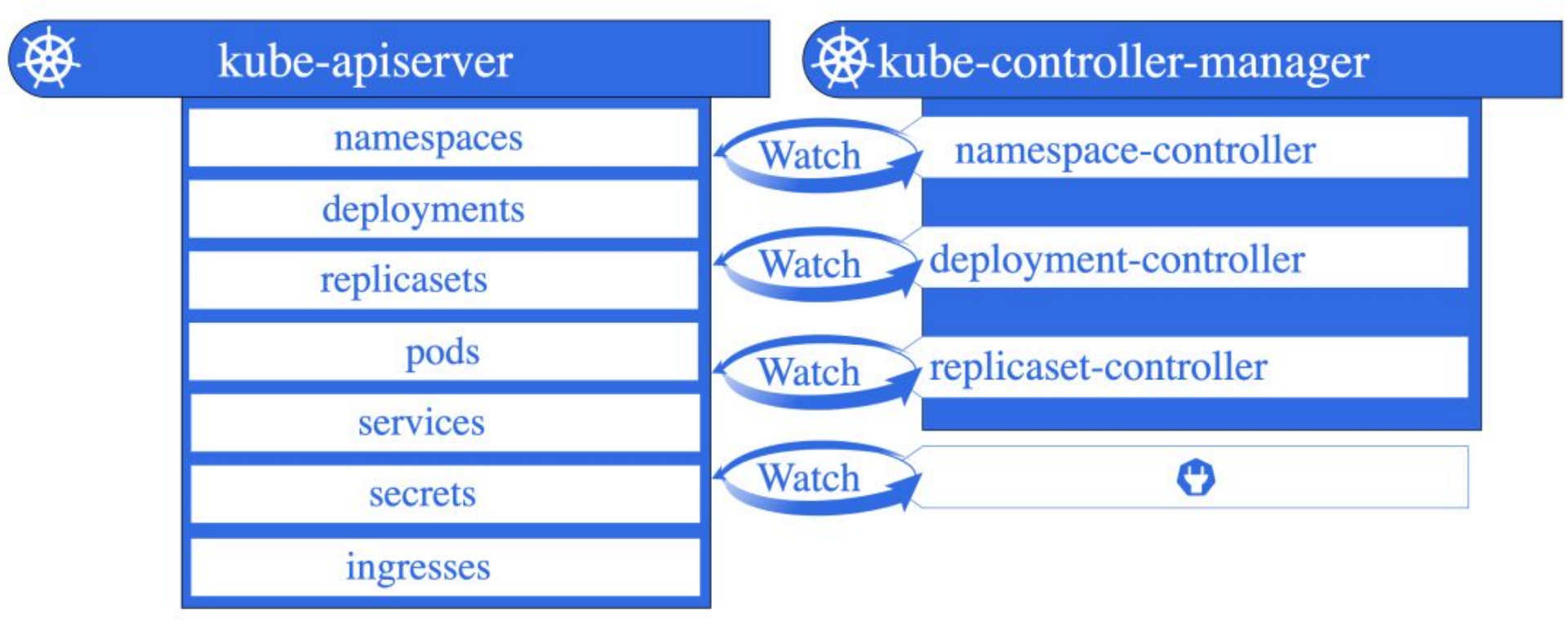
Kubernetes

K8s хранит декларативное описание желаемого состояния в виде набора ресурсов разных типов, а набор контроллеров (и операторов):
1. Обнаруживает несоответствия и делает все возможное, чтобы их устранить
2. Слушает изменения состояния и обновляет статус объектов

K8s – это конструктор, который позволяет хранить и работать с разными типами ресурсов и выбирать подходящий контроллер для достижения того или иного состояния.



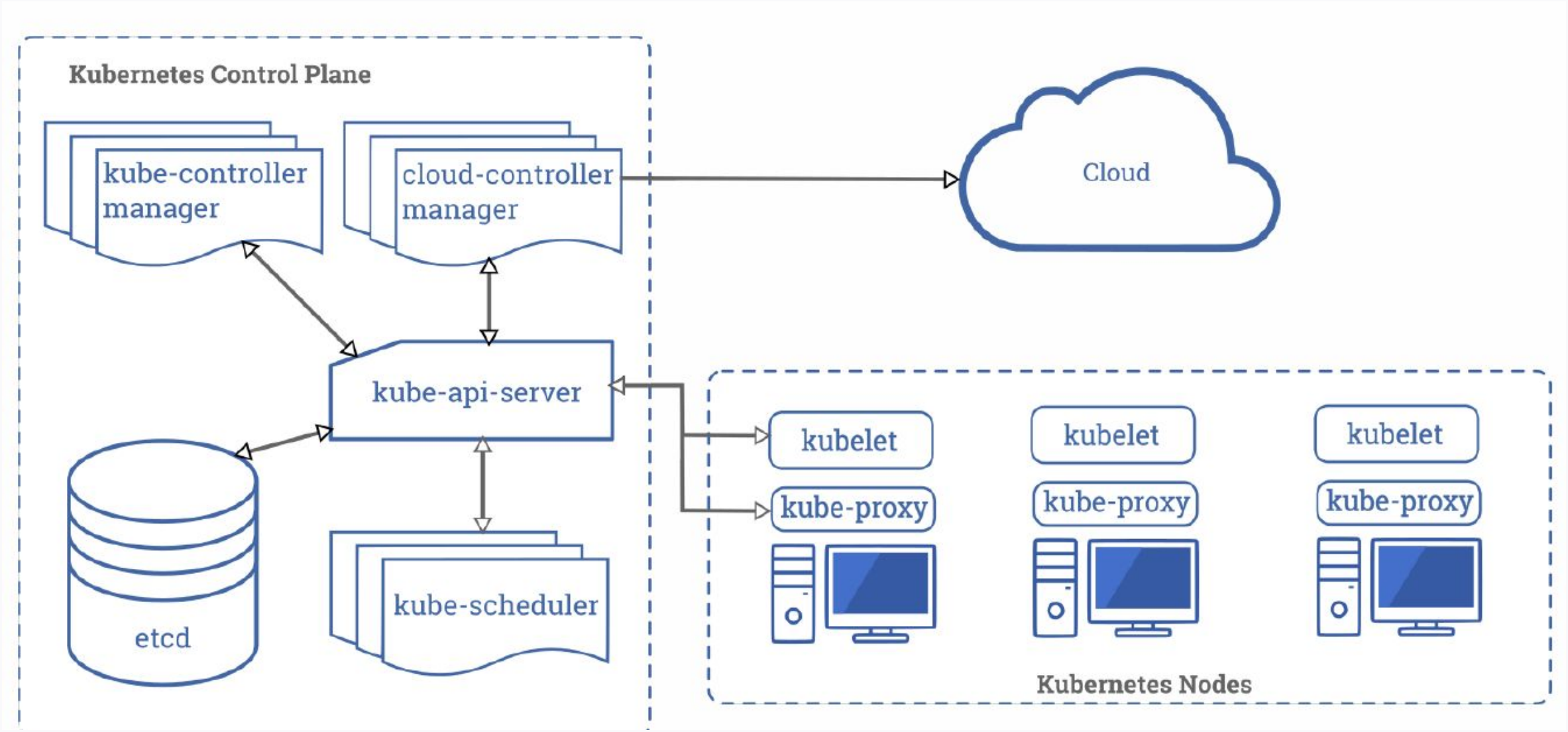
Kubernetes - controller



И есть отдельный компонент – controller-manager, который отвечает за

- Event loop внутри kubernetes
- Содержит встроенные контроллеры: например, deployment, replicaset и т.д

Kubernetes


Kubernetes Control Plane
kube-controller manager
cloud-controller manager
Cloud
kube-api-server
kubelet
kubelet
kubelet
kube-proxy
kube-proxy
kube-proxy
etcd
kube-scheduler
Kubernetes Nodes

Kubernetes. Состав

kube-apiserver - сервер принимающий апи команды для управления и мониторинга кластера

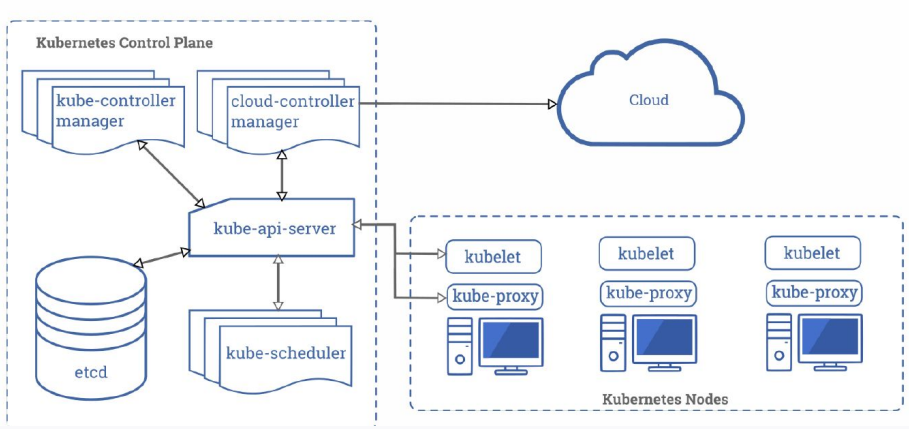
etcd - консистентное HA key-value хранилище для хранения состояния кластера

kube-scheduler - сервер для распределения созданных подов

kube-controller-manager - сервер для запуска и управления контроллерами

- Node controller: отвечает за ноды и информирование при их падении.
- Replication controller: отвечает за поддержание нужного количества реплицируемых объектов.
- Endpoints controller: соединяет поды и сервисы.
- Service Account & Token controllers: управляет аккаунтами и токенами для доступа к неймспейсам и объектам.

cloud-controller-manager -
с облачной инфраструктурой



https://kubernetes.io/docs/concepts/overview/components/

Kubernetes

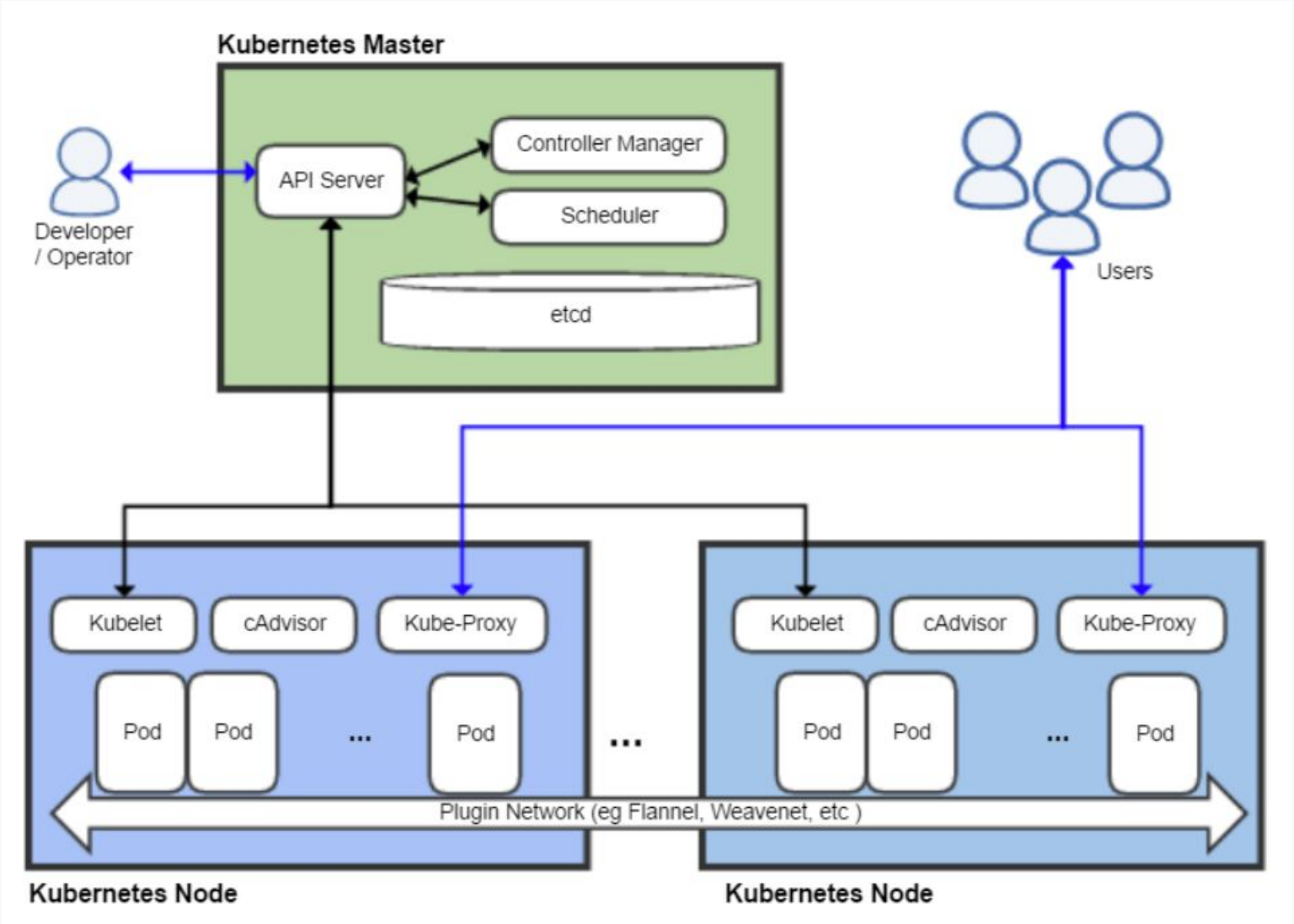
Node Components

на каждой ноде работают:

kubelet - запускает поды на этой ноде и следит за их работоспособностью

kube-proxy - отвечает за сетевое взаимодействие

Kubernetes

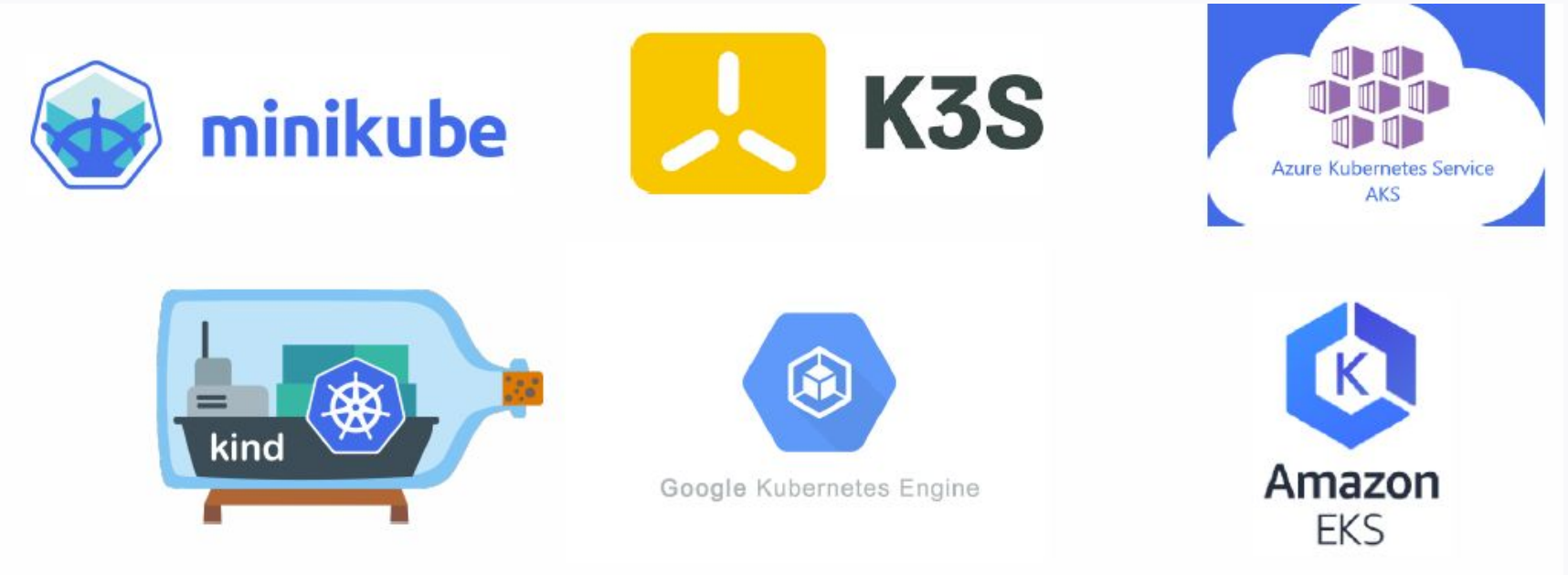

Kubernetes Master
API Server
Controller Manager
Scheduler
etcd
Developer
/ Operator
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

Kubernetes. Варианты развёртывания

BareMetal

Локальные инсталляции K8S:
- **minikube**
- k3s
- kind
- Micro8ks

Облачные инсталляция K8S:
- GKE (Google)
- AKS (Azure)
- EKS (Amazon)



Kubernetes. Minikube

Minikube – это локальная инсталляция K8S из одной мастер-ноды.

Удобно для проведения тестов и знакомства с кубиком.

- https://kubernetes.io/docs/tasks/tools/install-minikube/ - инструкция по установке

Объекты

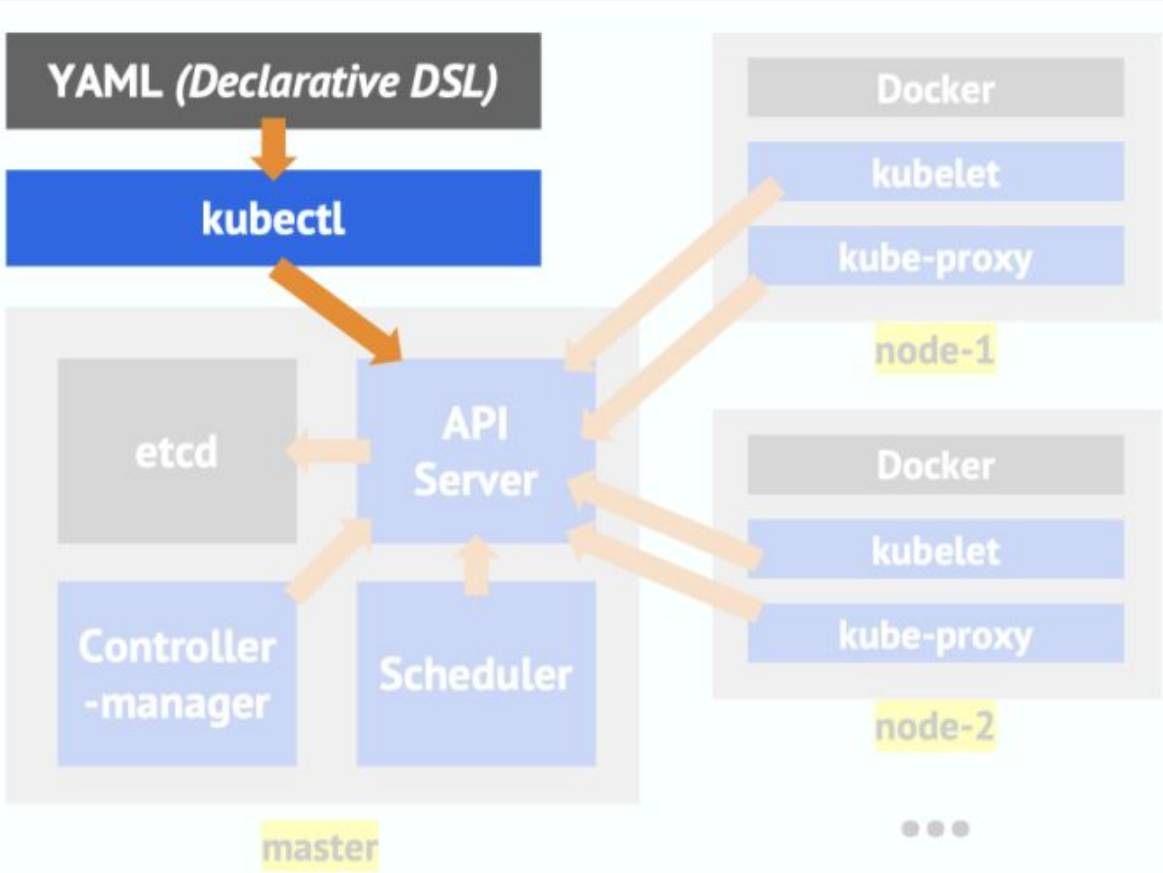
Объекты kubernetes

Объекты в Kubernetes – это хранящиеся внутри Kubernetes сущности (ресурсы). И Kubernetes их использует для представления состояния кластера.

Объекты могут описывать:
• Какие приложения сейчас запущены и работают
• Какие ресурсы доступны этим приложениям
• Политики поведения, такие как политика рестартов, обновлений и т.д.

Для того, чтобы изменить желаемое состояние, например:
• увеличить количество инстансов сервиса с 2 до 3
• полностью потушить деплой
• убрать сервис из балансировки
и т.д. - необходимо просто удалить или изменить объекты.

Объекты kubernetes



Внутри etcd информация и объекты Kubernetes хранятся в своем каком-то
формате, но извне чаще всего используется описание объектов yaml
через CLI инструменты типа kubectl

Внутри себя api-server использует json.

Все объекты в Kubernetes имеют общий формат описания в yaml

- kind – тип (класс) объекта
- apiVersion – группа api
- metadata – метаданные: имя, метки, аннотации
- spec – желаемое состояние объекта
- status - текущее состояние объекта

Namespace

Namespace - некий виртуальный кластер.

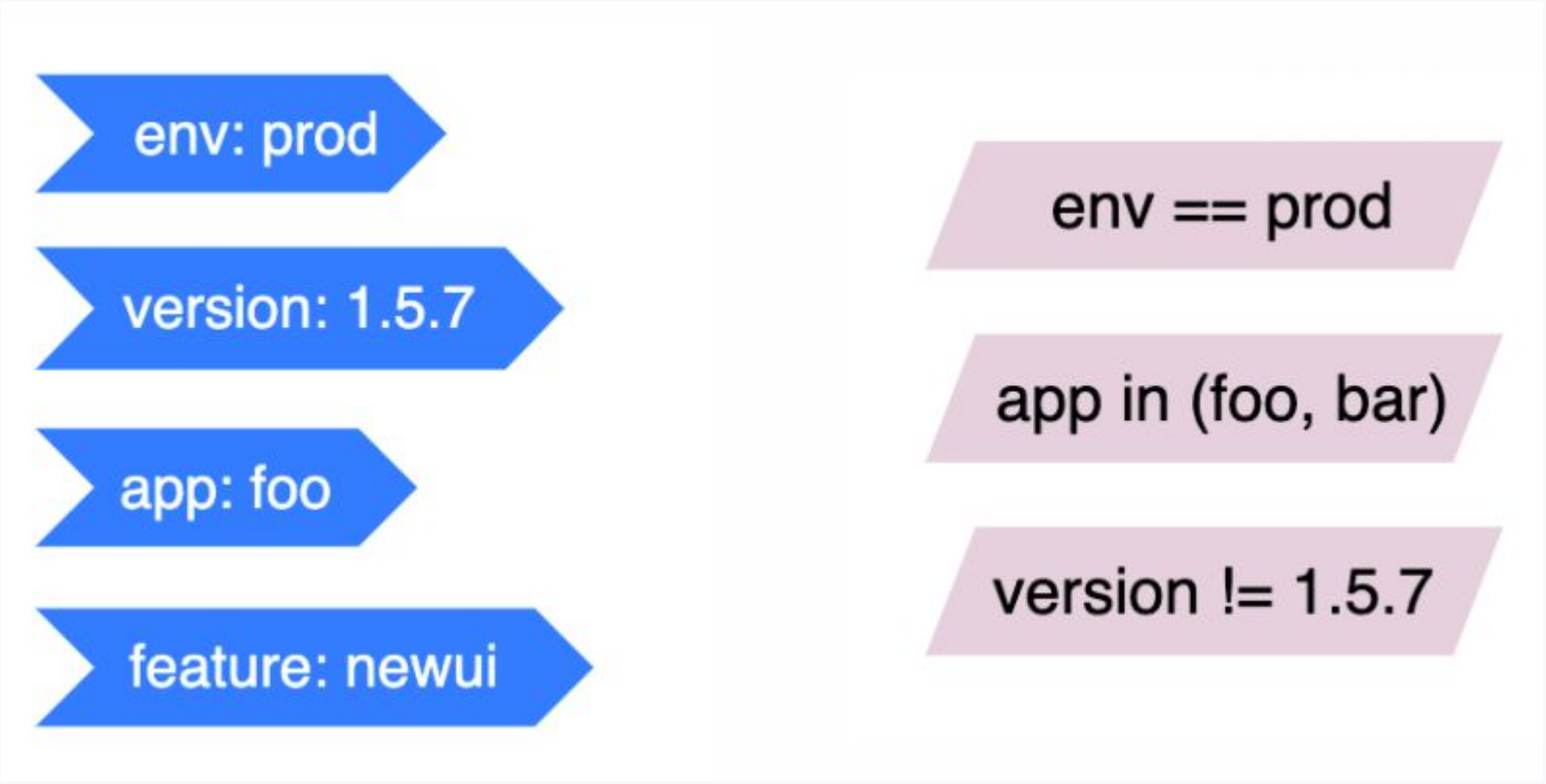
Объекты могут привязаны к namespace-у, могут быть не привязаны -
в зависимости от типа объекта. Например, deployment привязан, а node – нет.

Namespaces – это способ отделить объекты друг от друга и разделить
права доступа и ресурсы между разными командами (например)

Метки и селекторы

Многие контроллеры работают не с одним объектом, а с несколькими.
Для того, чтобы ссылаться используются метки и селекторы.

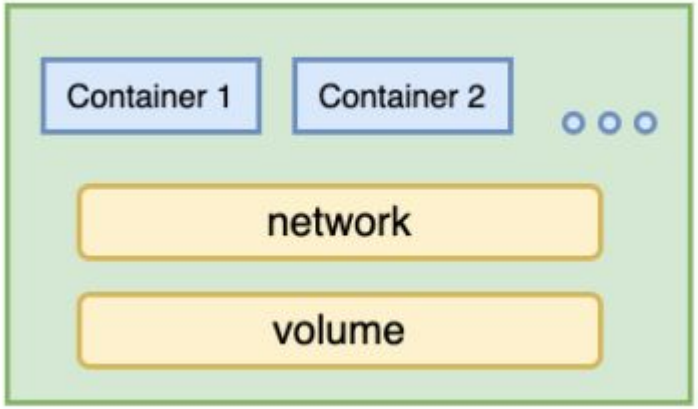
Метки (labels) – это значения типа key: value
Селекторы – это выражения, позволяющие выбрать объекты по меткам




POD

Pod

Pod – это объект Kubernetes, который является атомарной единицей рабочей нагрузки.
Pod можно воспринимать, как запущенный инстанс сервиса или приложения
При этом внутри пода может быть один или несколько контейнеров.
Внутри пода контейнеры разделяют сетевой интерфейс и имеют общее локальное хранилище
Kubernetes запускает, удаляет и обновляет контейнеры, описанные в Pod-е как единое целое



ReplicaSet

ReplicaSet

Под-ы в чистом виде практически никто не использует, потому что за подами надо приглядывать. Если под упадет, он сам по себе не поднимется. Кроме того, для отказоустойчивости мы захотим иметь несколько инстансов одного и того же сервиса, и если упадет нода (например), нам нужно, чтобы количество инстансов осталось прежним.

Эту проблему в некотором смысле решает такая сущность (ресурс) кубернетеса, как **replicaset**

ReplicaSet

ReplicaSet – это объект Kubernetes, который описывает в скольких количествах экземпляров должен быть запущен под (с шаблоном).

Следит за этим соответствующий встроенный контроллер, который обеспечивает, что поды запущены ровно в указанном количестве экземпляров.

Если количество подов больше, контроллер убивает лишние поды. Если количество подов меньше запускает еще один под по шаблону из спецификации.


Deployment

Deployment

Deployment – это объект Kubernetes, который описывает в скольких экземплярах запущен сервис, а также стратегию обновления на новую версию.

Следит за этим встроенный контроллер, который позволяет не только следить за количеством реплик, но также позволяет обновлять версию и в случае ошибок откатить или остановить раскладку.

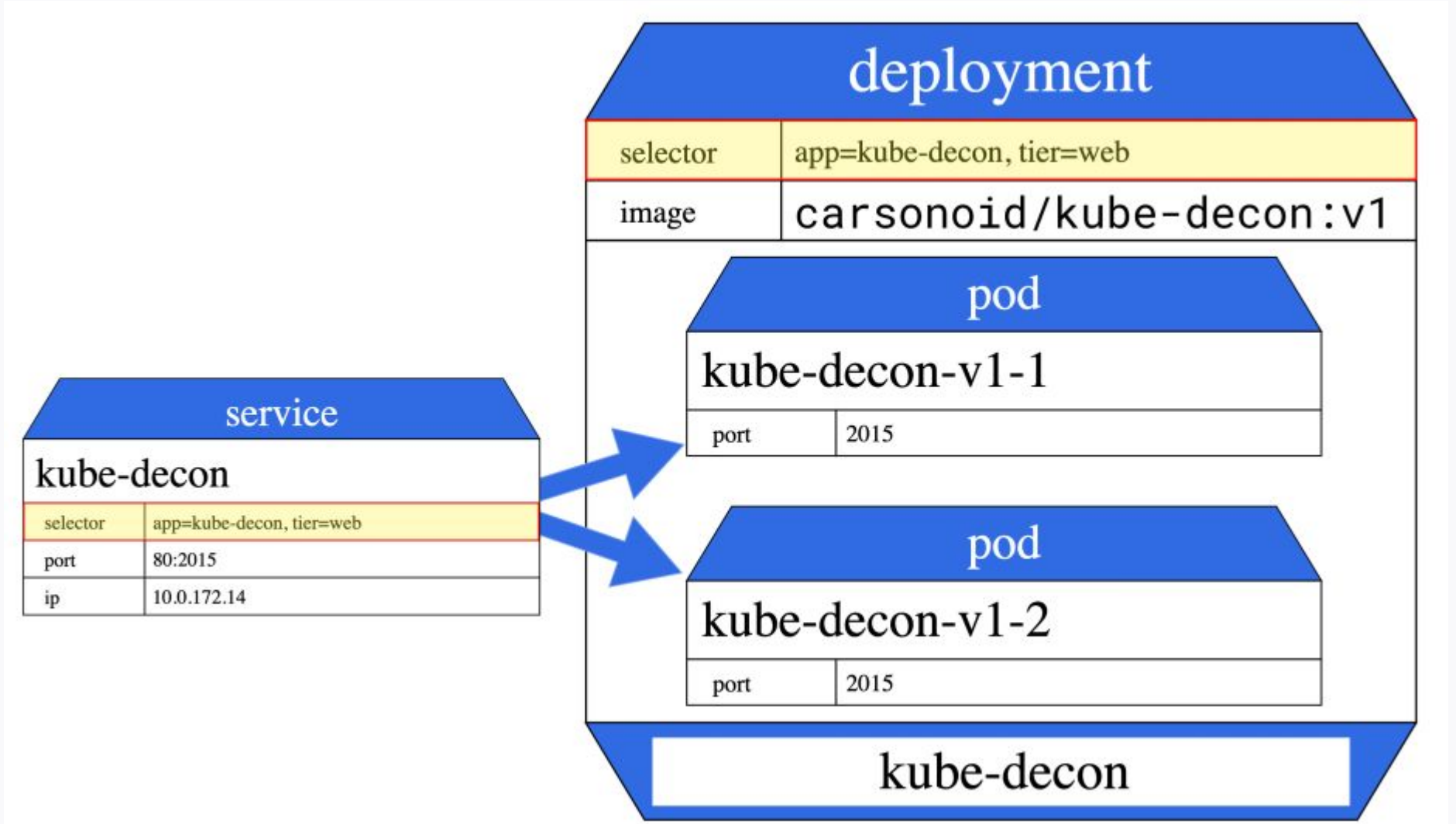
Service

Service

Service – это объект Kubernetes, который определяет некоторое количество подов в абстракцию сервиса.

Встроенный контроллер позволяет с помощью объекта Service организовать роутинг и балансировку.

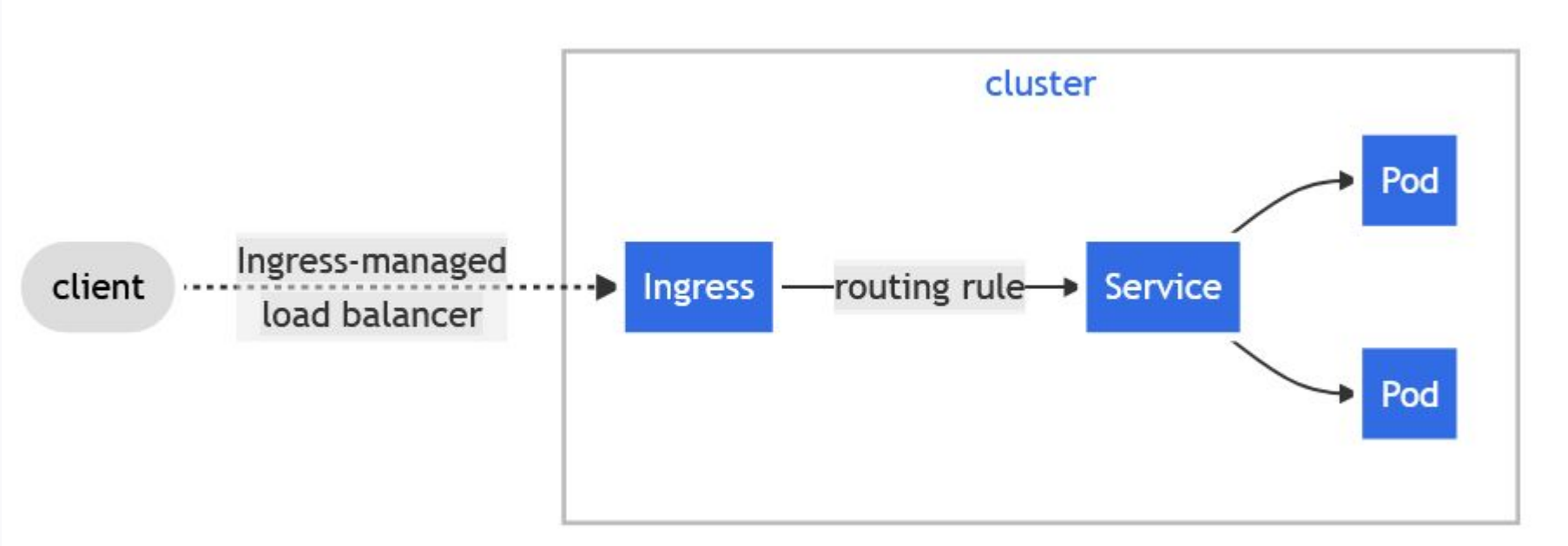
Service

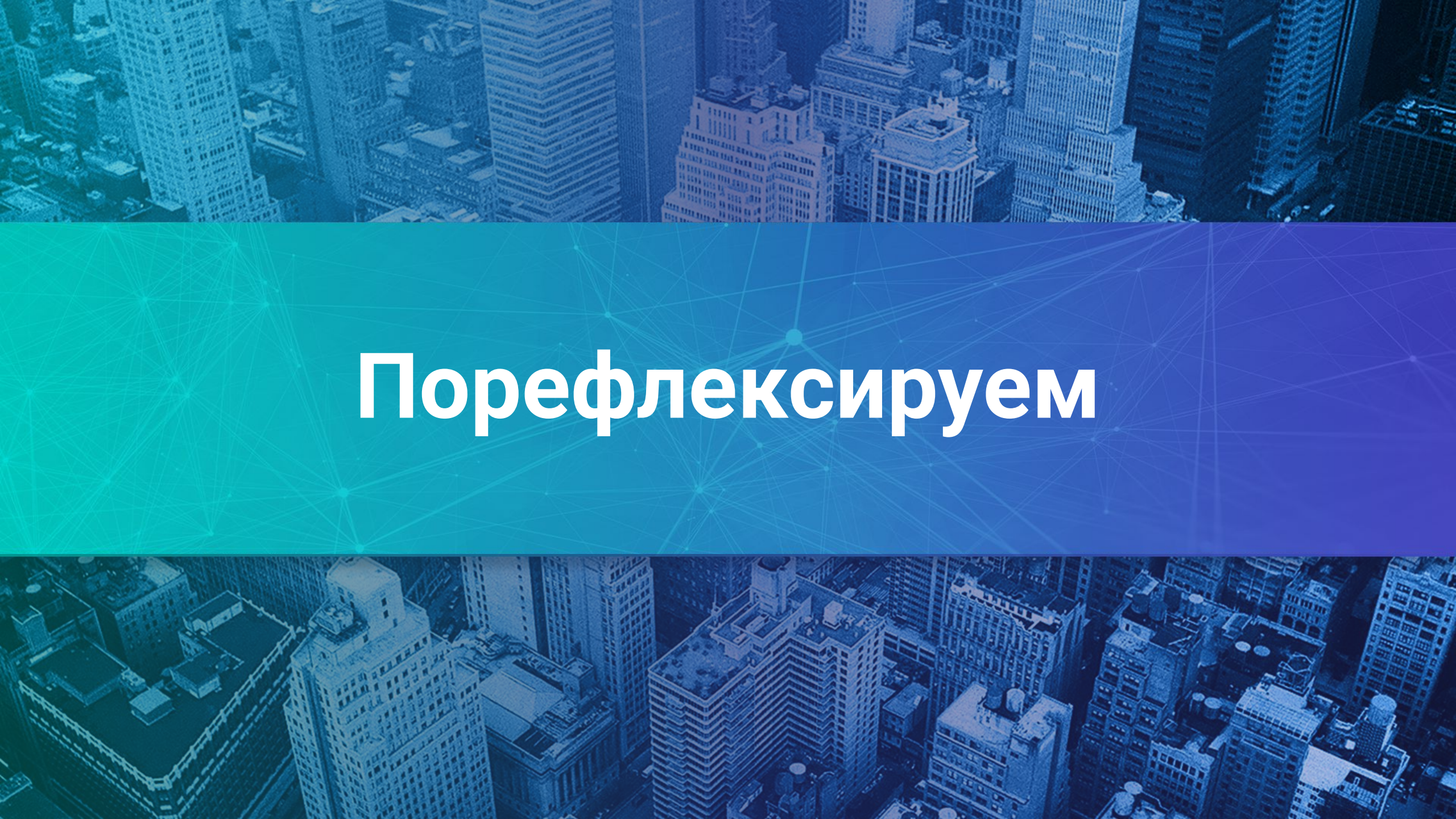

deployment
selector
app=kube-decon, tier=web
image
carsonoid/kube-decon:v1
pod
kube-decon-v1-1
port
2015
pod
kube-decon-v1-2
port
2015
kube-decon
service
kube-decon
selector
app=kube-decon, tier=web
port
80:2015
ip
10.0.172.14


Ingress

Ingress

Ingress – это объект Kubernetes, который описывает как роутится внешний трафик.




Порефлексируем

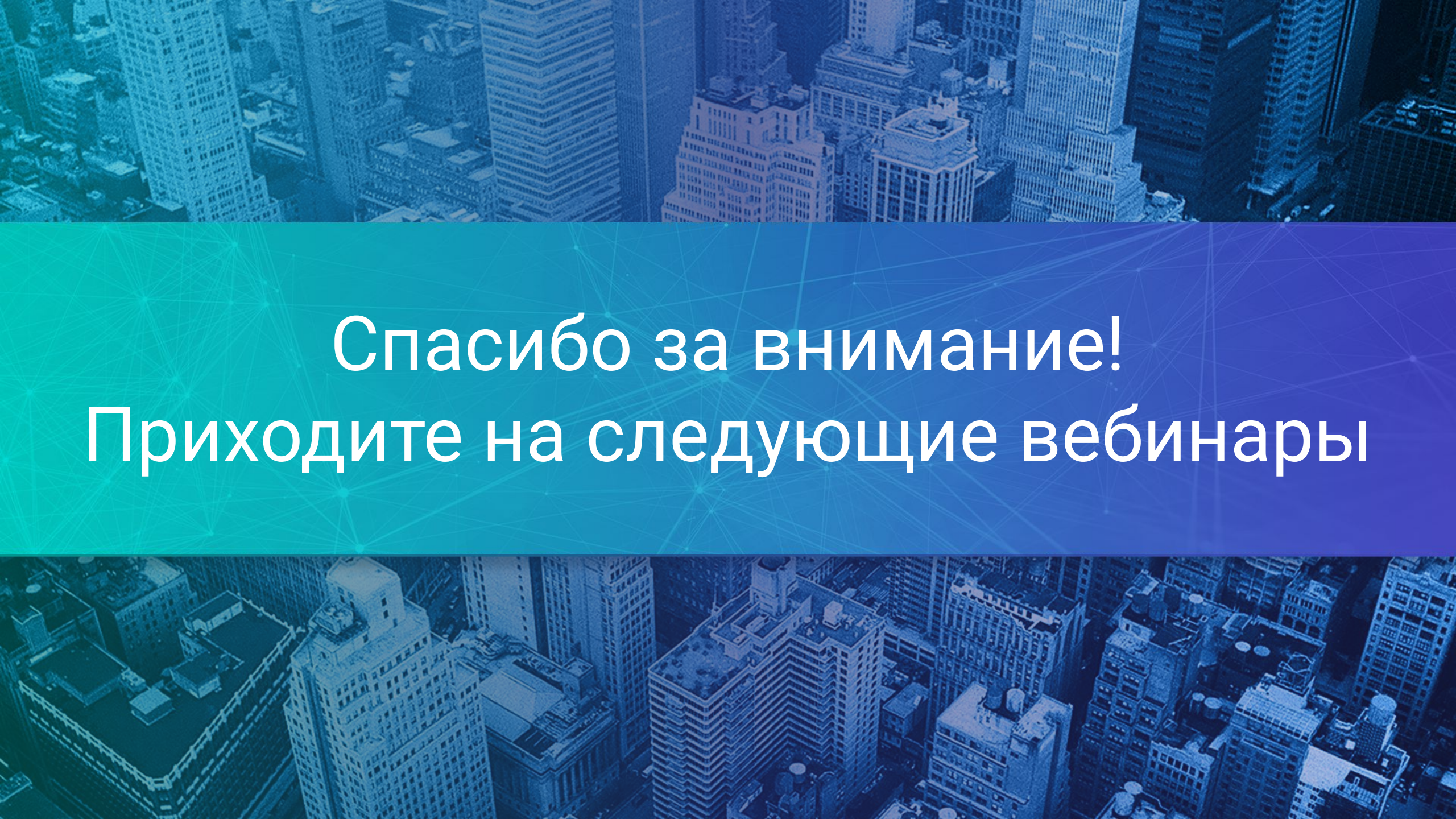
Опрос и минитест

https://otus.ru/polls/32378

https://forms.gle/HKm5MjsmHds84oBs9

Вопросы?

- **Pod**
- **ReplicaSet**
- **Deployment**
- **Service**
- **Ingress**


Спасибо за внимание!
Приходите на следующие вебинары