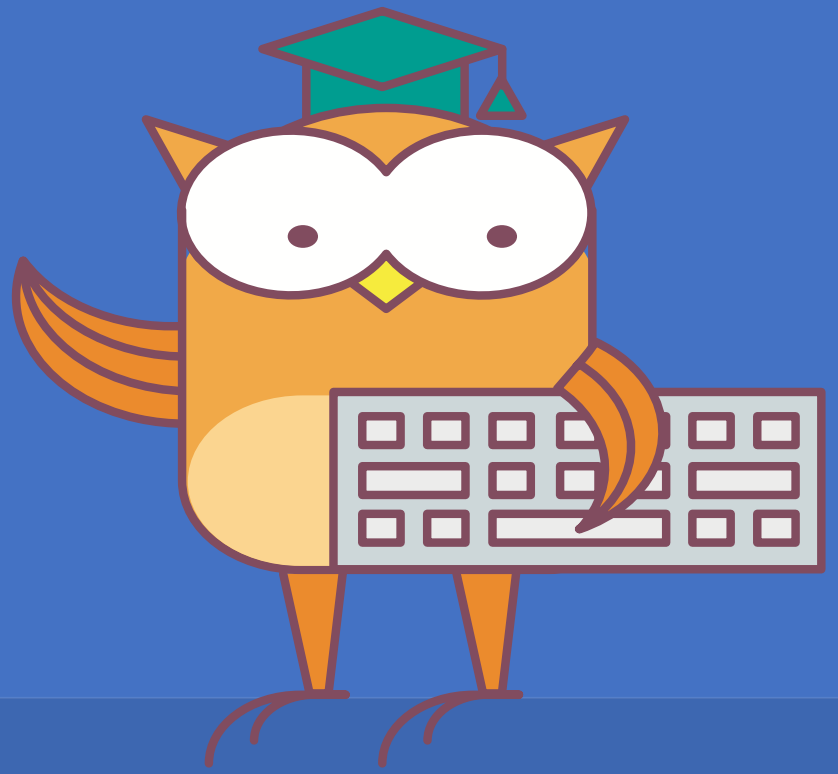
OTUS

Инфраструктурные паттерны и основы Kubernetes

Архитектор ПО



Щетинников Стас

Меня хорошо слышно && видно?

Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

Карта вебинара

- Основы Kubernetes

Какие проблемы возникают в микросервисной архитектуре?

Когда сервисов много, управление ими становится сложнее. Издержки на управление растут экспоненциально с количеством сервисов.

Ручное управление перестает работать, нужны инструменты для решения проблем:

- Как на одной машине запустить процессы с разным окружением?
- Как изолировать сервисы, которые работают на одной машине?
- Как можно автоматически перекидывать сервисы в случае аварии на машине?
- Как обеспечить беспрерывную поставку?
- Как сделать обнаружение сервисов, если сервисы могут с одной ноды переезжать на другую?
- Как выбирать куда и сколько инстансов сервисов деплоить?

И т.д. и т.п.

01

Kubernetes

Kubernetes

Kubernetes – это оркестратор, который решает следующие задачи:

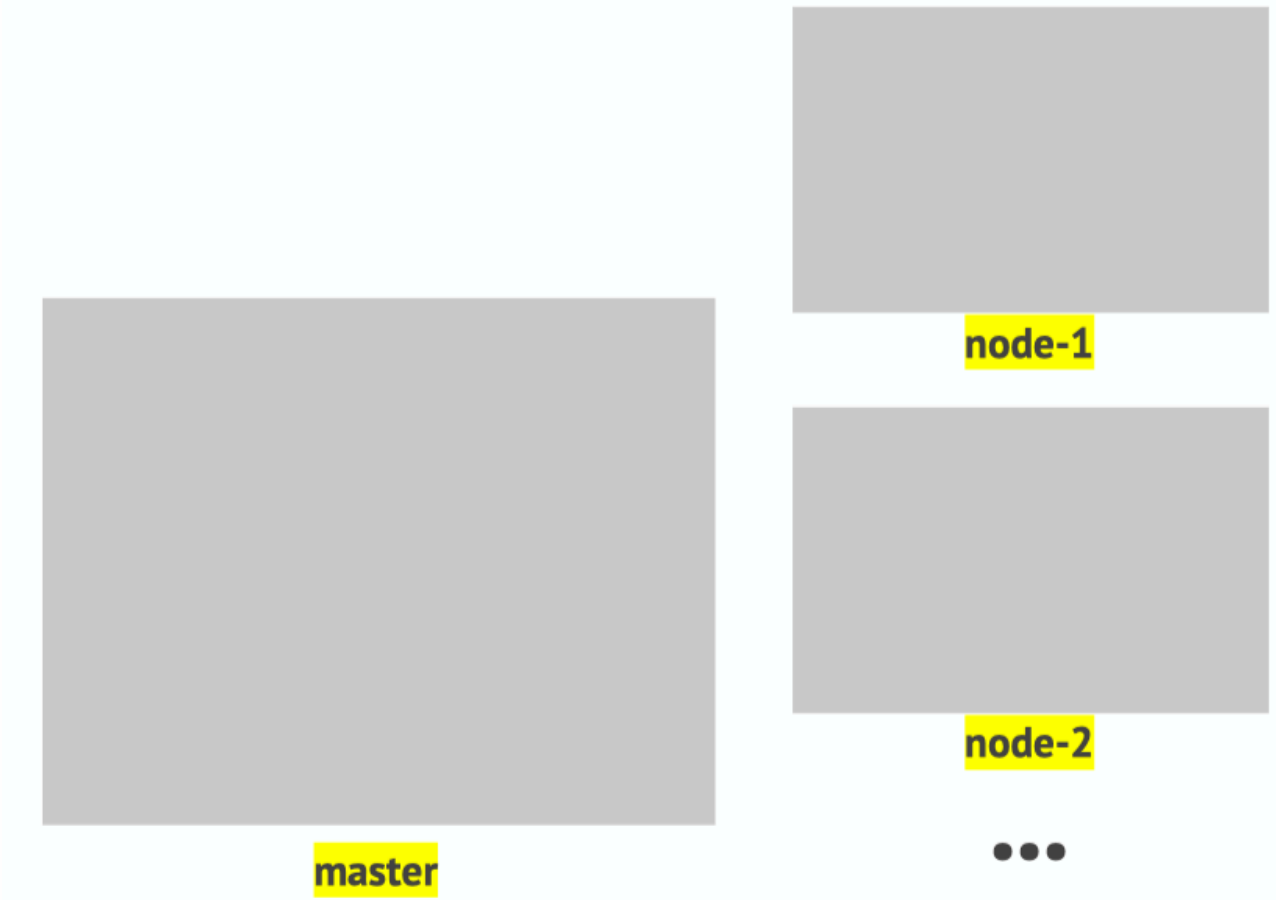
- Service discovery и балансинг
- Управление хранилищами
- Деплой новых версий и откат
- Распределение контейнеров по нодам
- Управление конфигурациями
- Средства self-healing

Kubernetes не решает (из коробки) такие задачи:

- Из коробки нет логгирования 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/

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

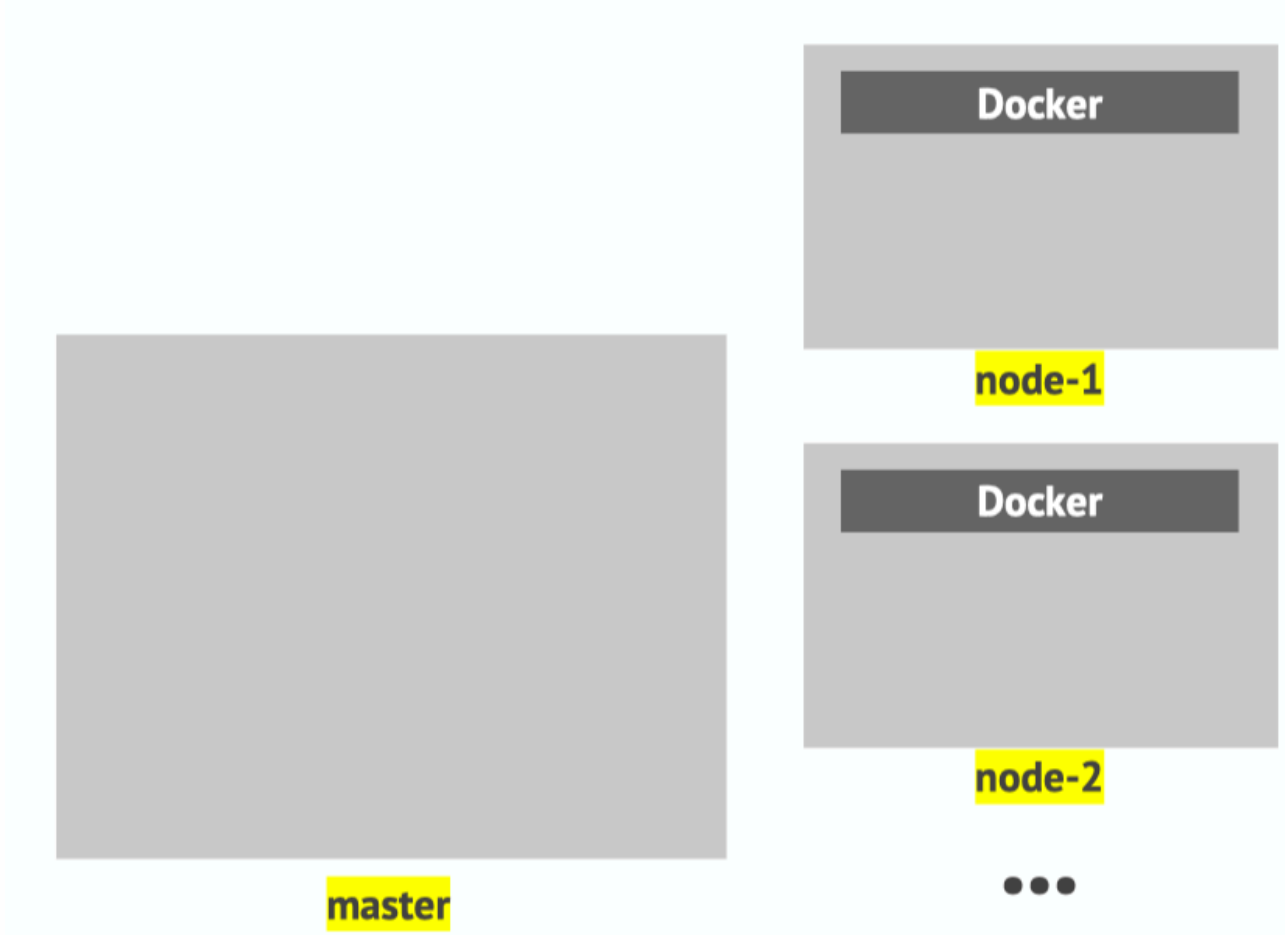
Архитектура Kubernetes

Чтобы запускать на рабочих нодах приложения, необходима 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).

https://kubernetes.io/docs/concepts/containers/

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

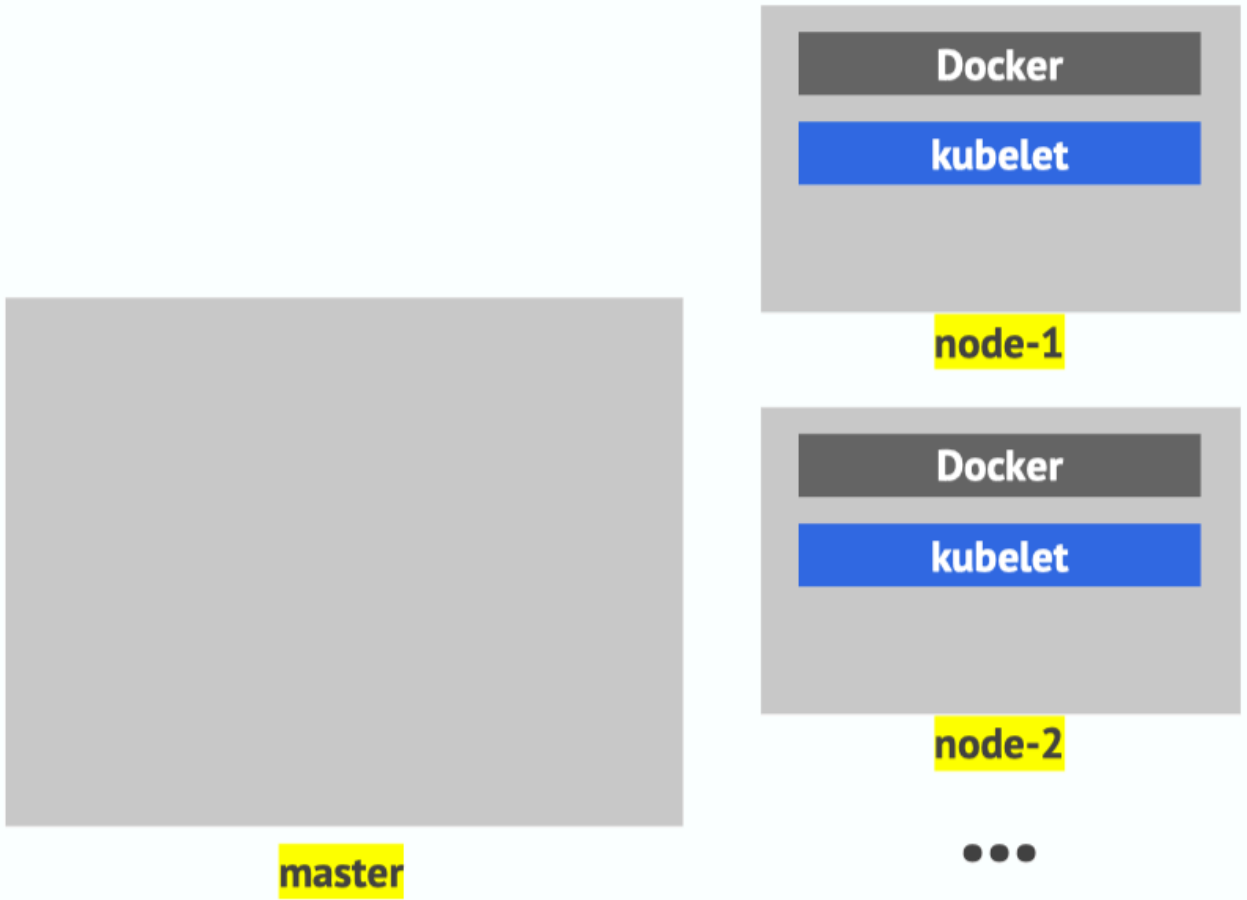
Архитектура Kubernetes

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**.

https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

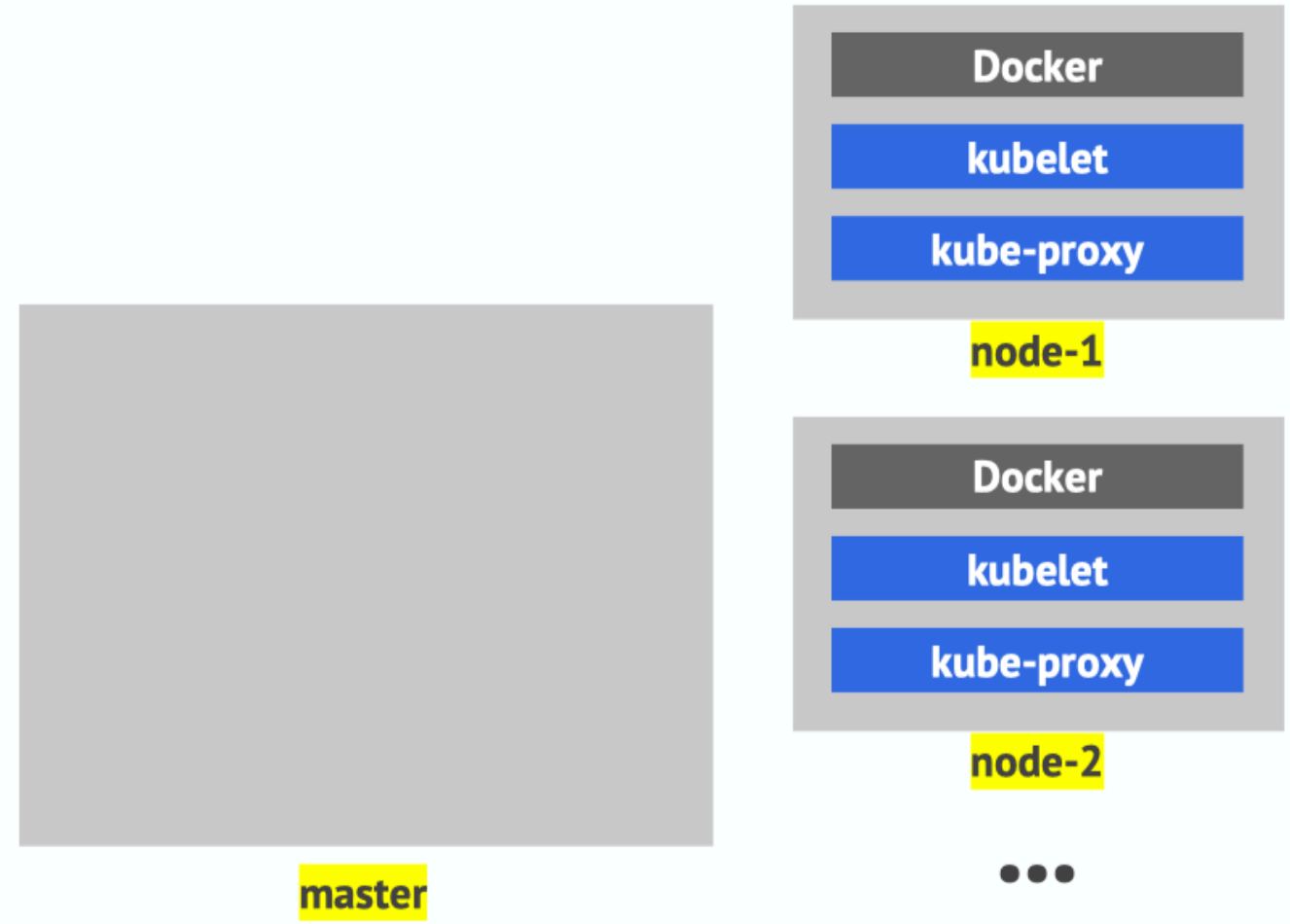
Архитектура Kubernetes

Для того, чтобы обеспечивать балансировку и service discovery, на каждой ноде необходимо прописывать соответствующие правила в сетевом файерволе (iptables, ipfw).

Агентом, который занимается на всех нодах, является **kube-proxy**.

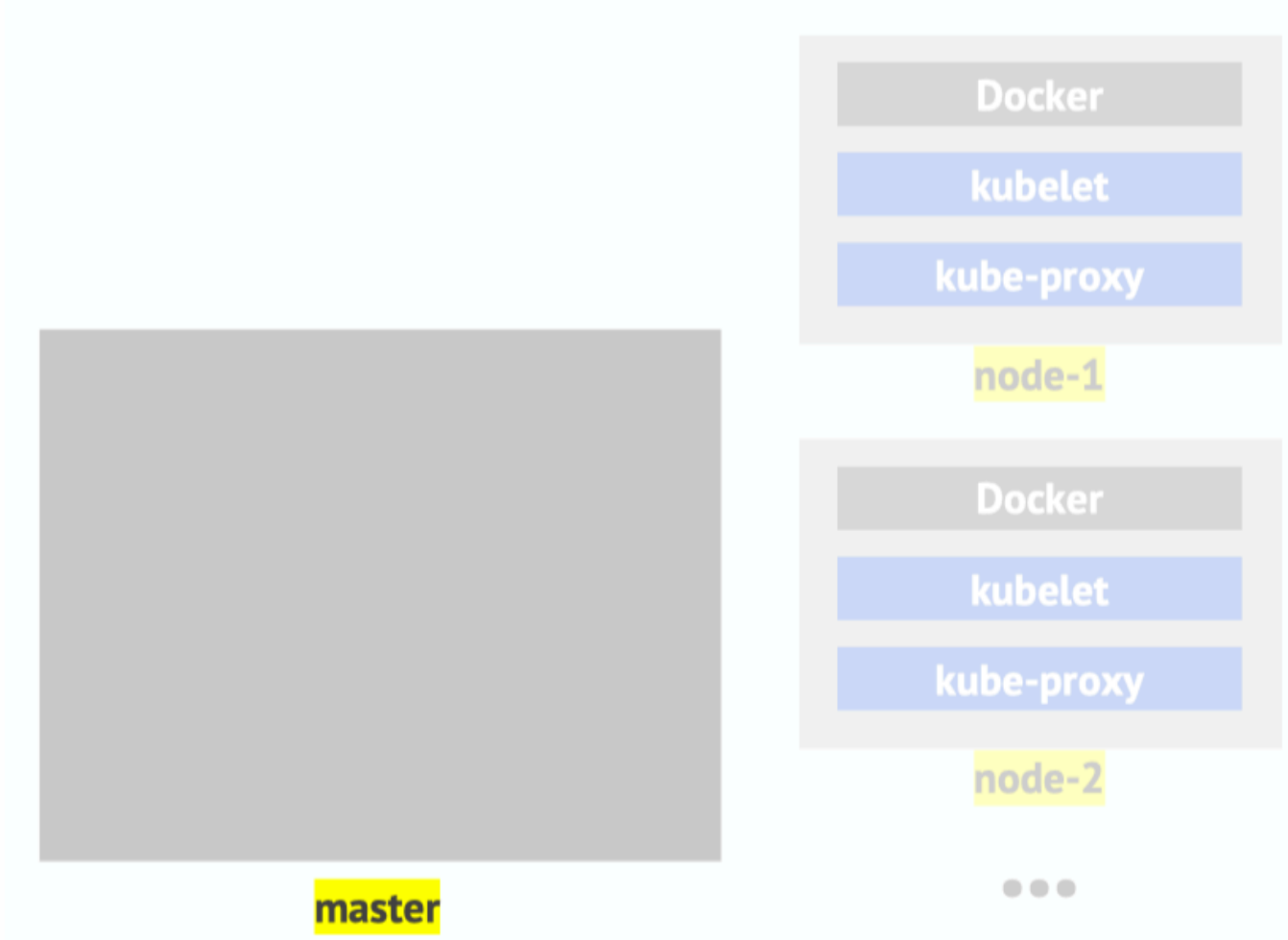
https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

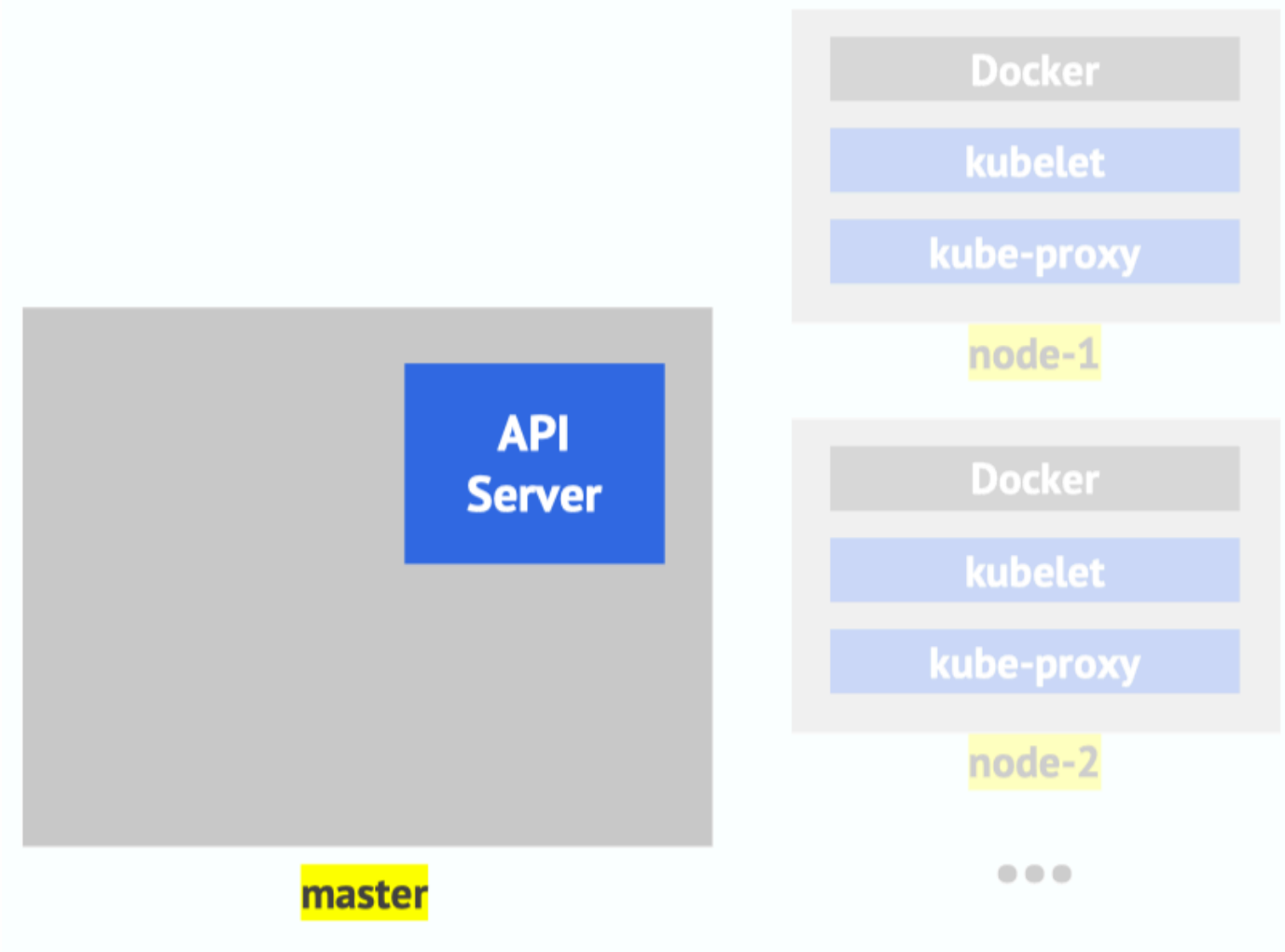
Архитектура Kubernetes

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**.

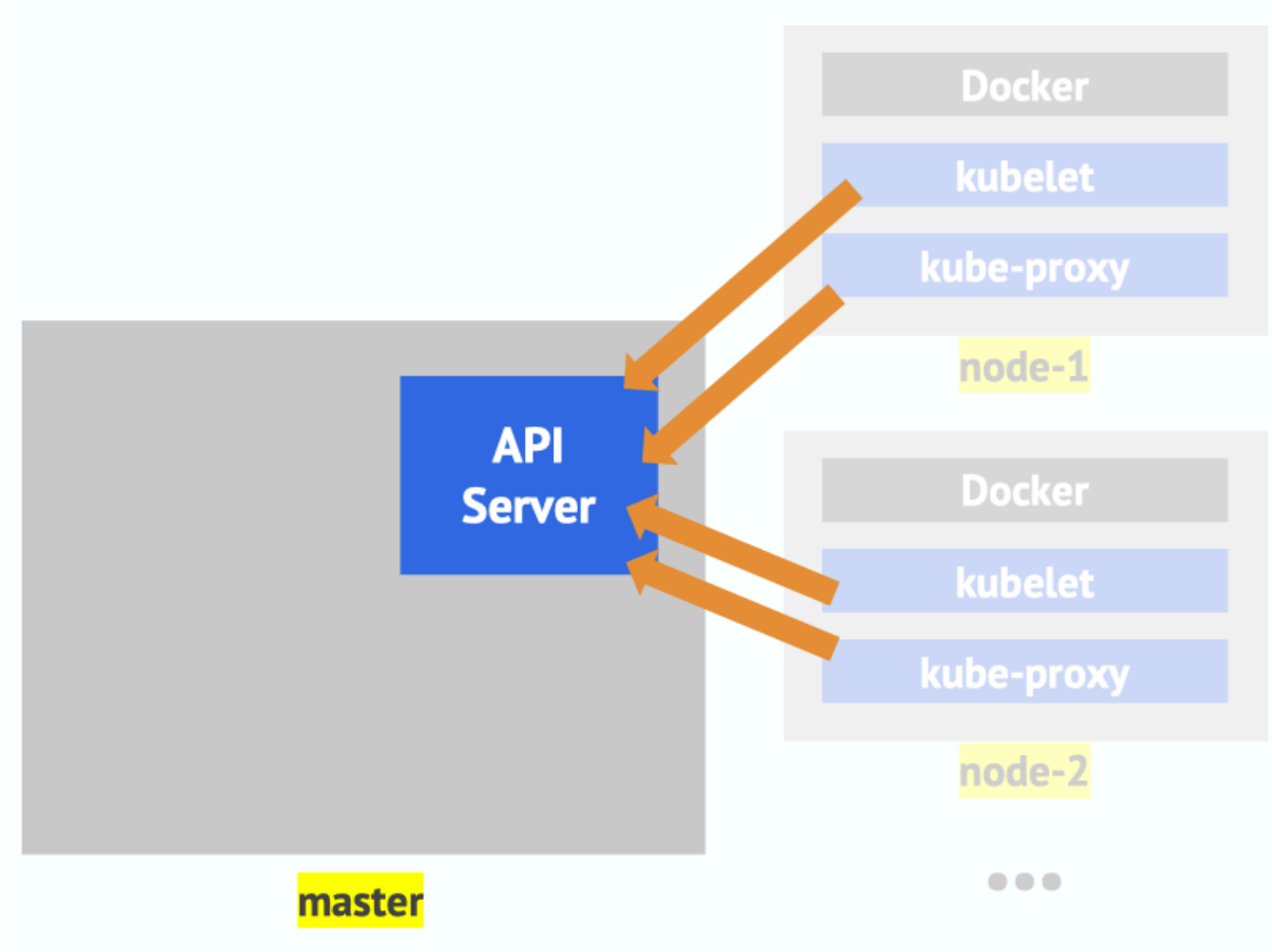
https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

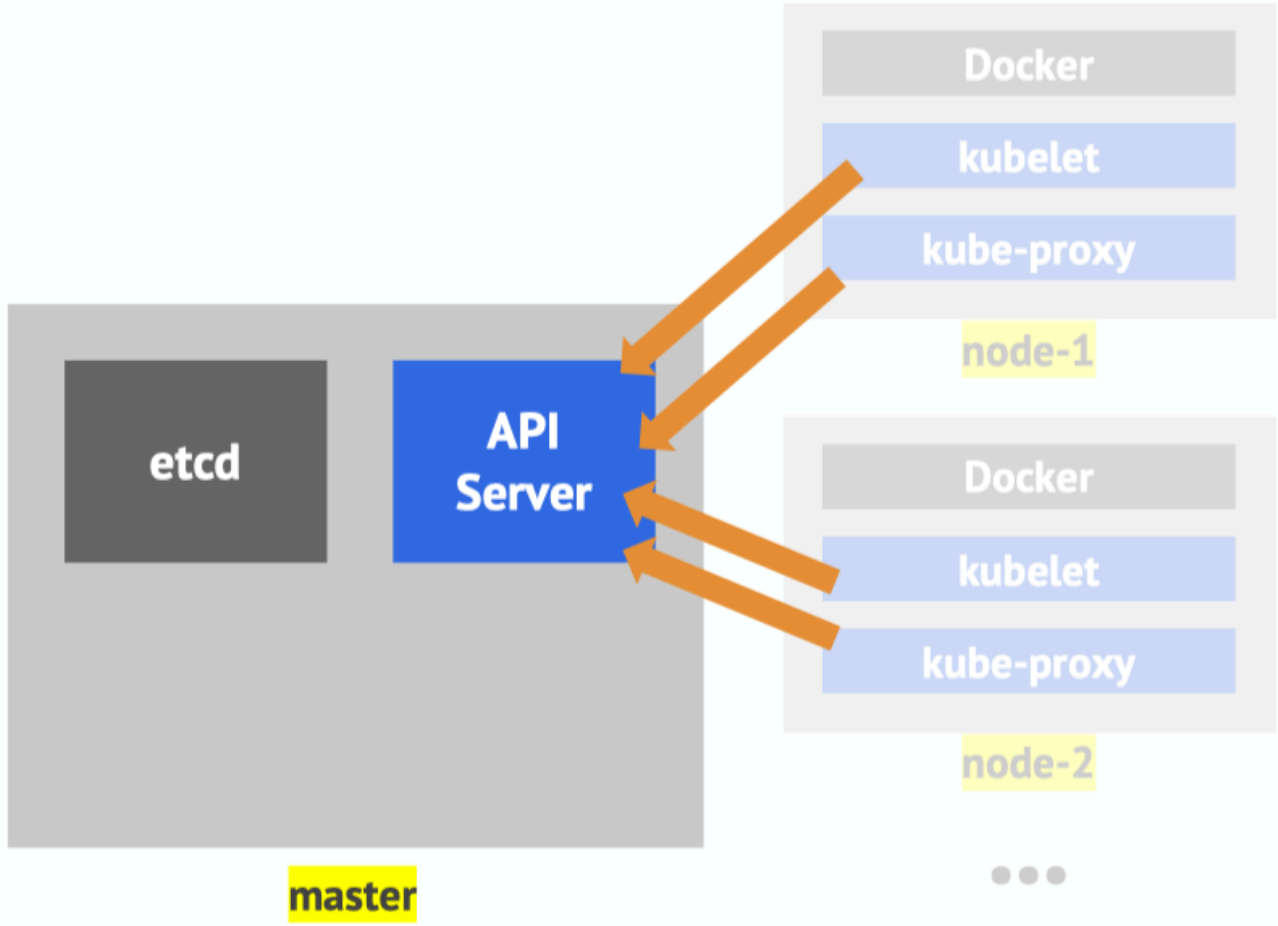
Архитектура Kubernetes

Для того, чтобы где-то хранить текущую конфигурацию кластера, необходимо хранилище (отказоустойчивое, децентрализованное и консистентное).

В kubernetes таким хранилищем является **etcd**.

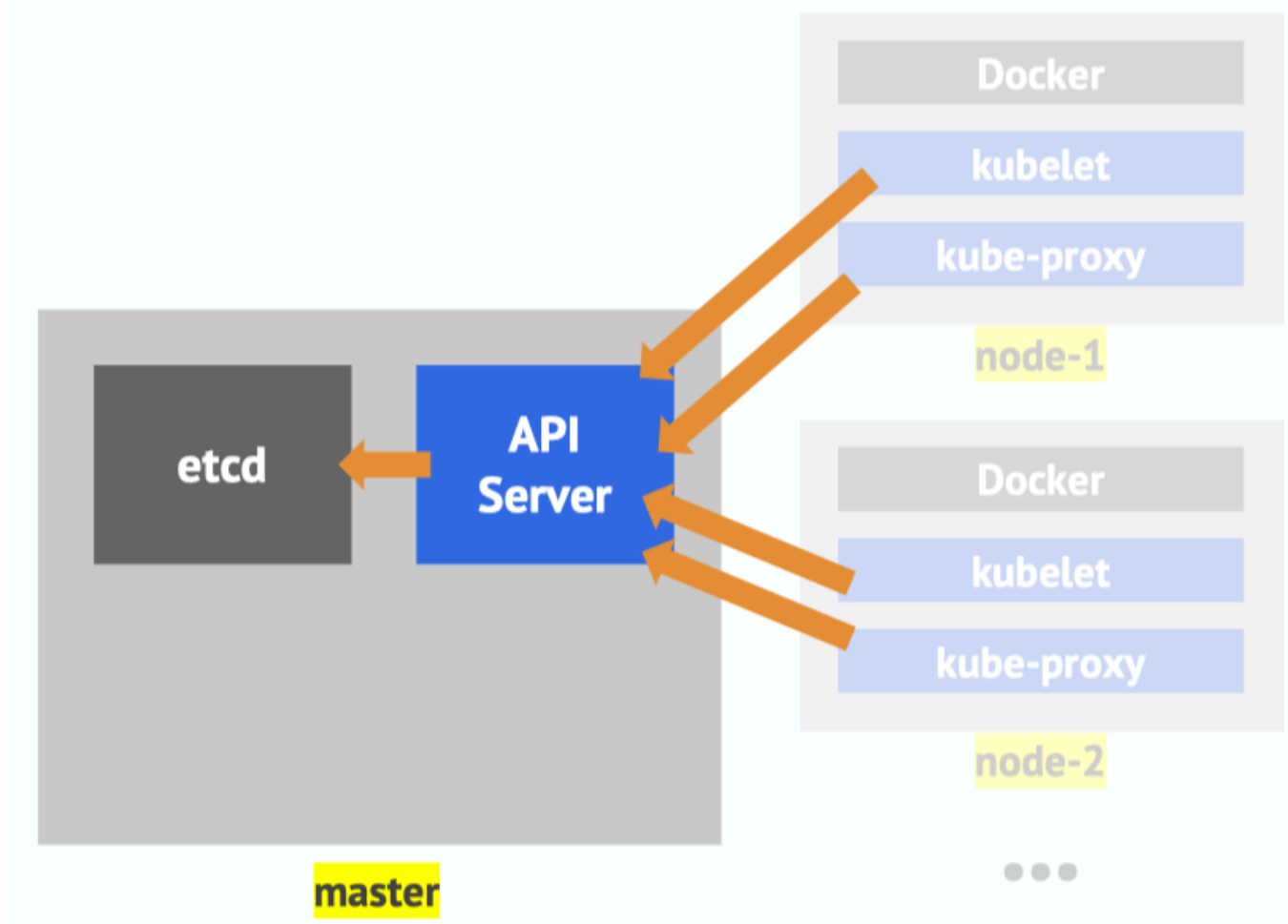
https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

Архитектура Kubernetes



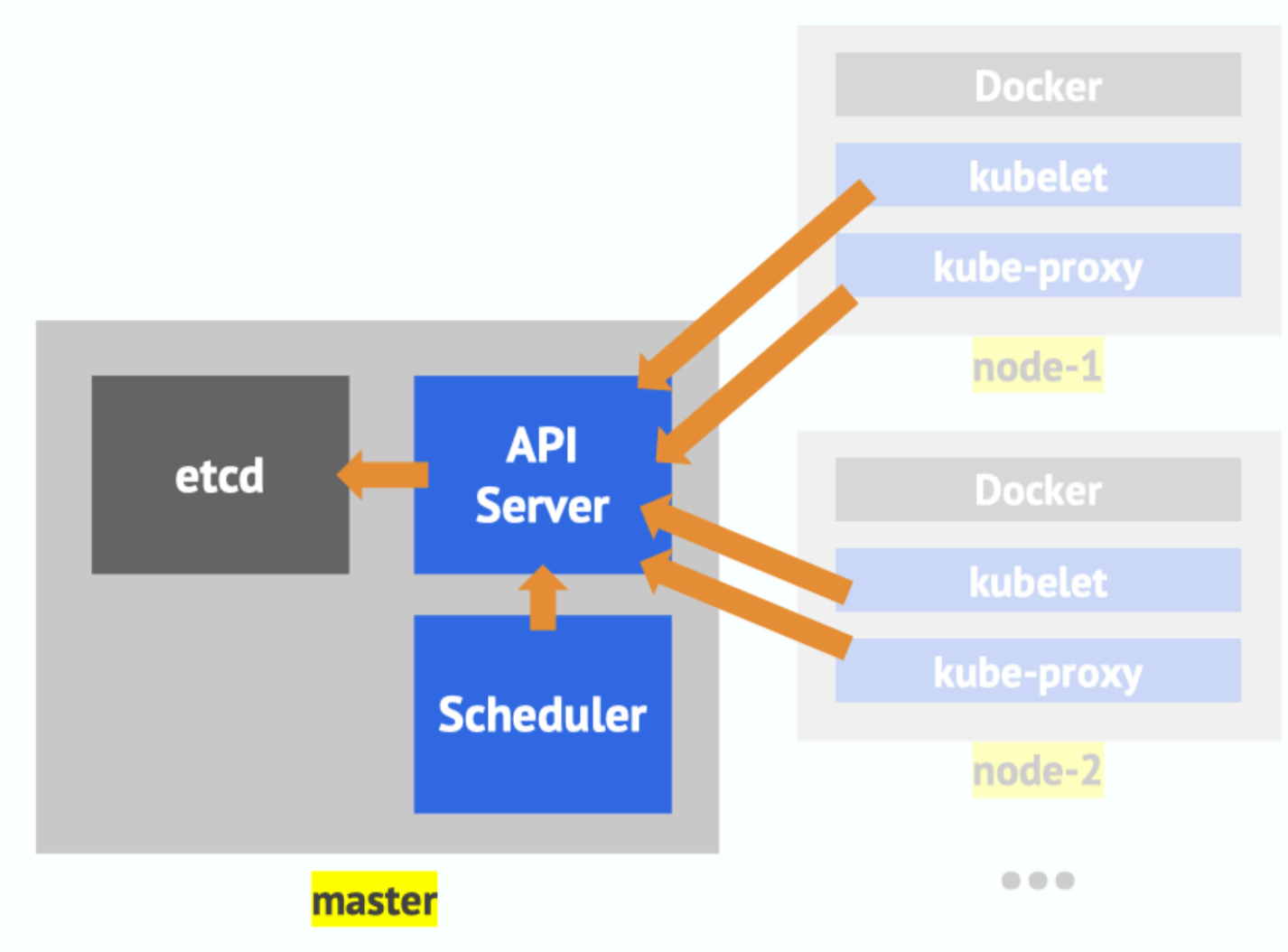
https://www.youtube.com/watch?v=CgCLPYJRxbU

Архитектура Kubernetes

Так же есть отдельный компонент – **scheduler** , который отвечает за распределение подов (контейнеров) по нодам.

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

Архитектура Kubernetes

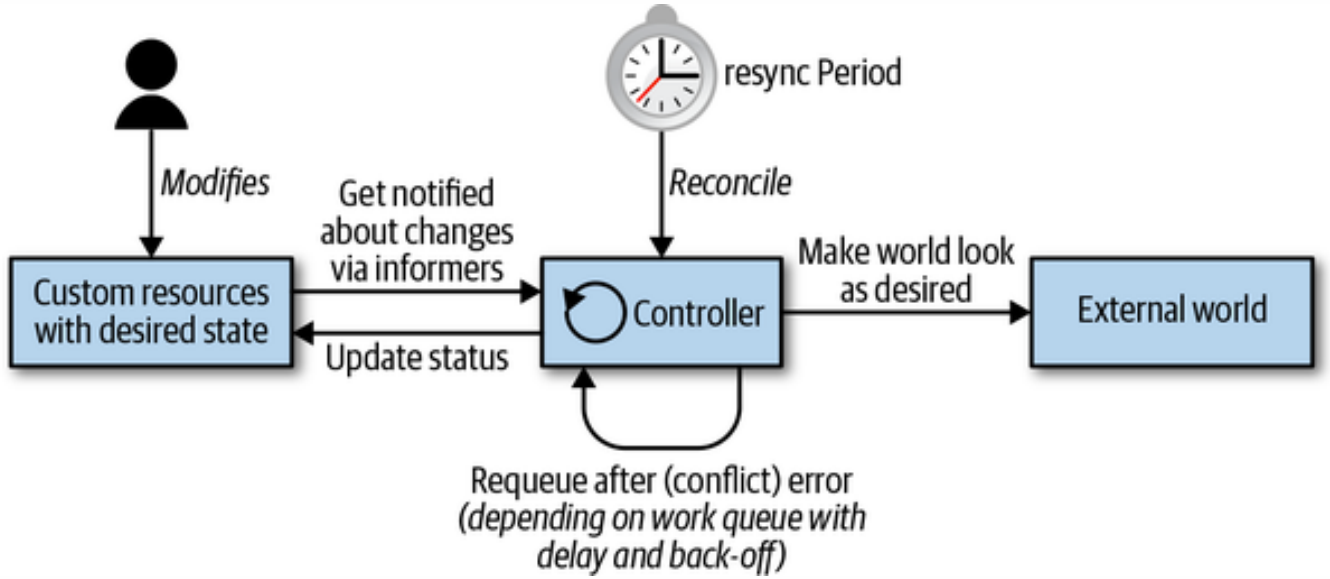


https://www.youtube.com/watch?v=CgCLPYJRxbU

Архитектура Kubernetes

Кубернетес 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

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

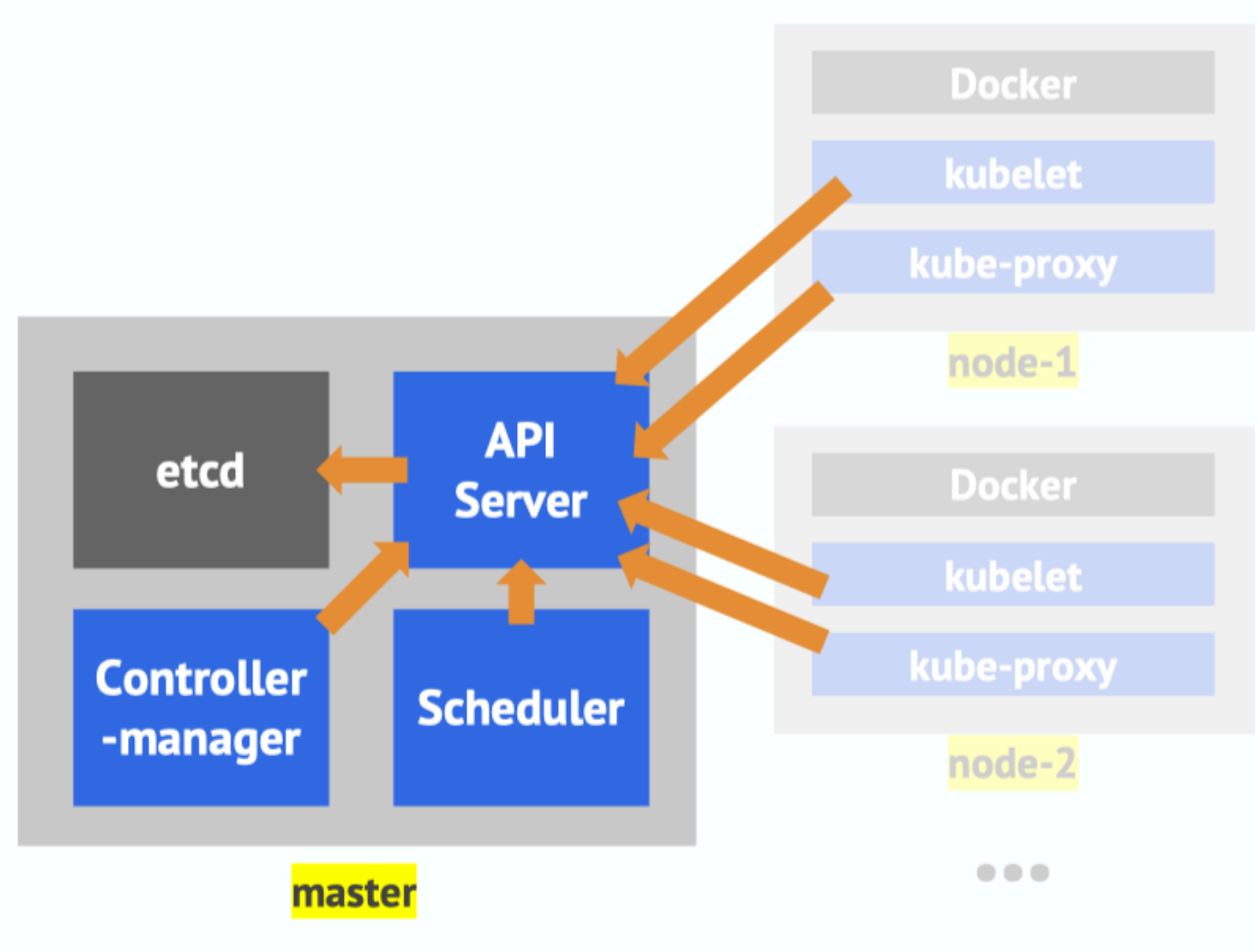


Архитектура Kubernetes

И есть отдельный компонент – **controller-manager**, который отвечает за контроллеры (deployment, replicaset и т.д) внутри Kubernetes.

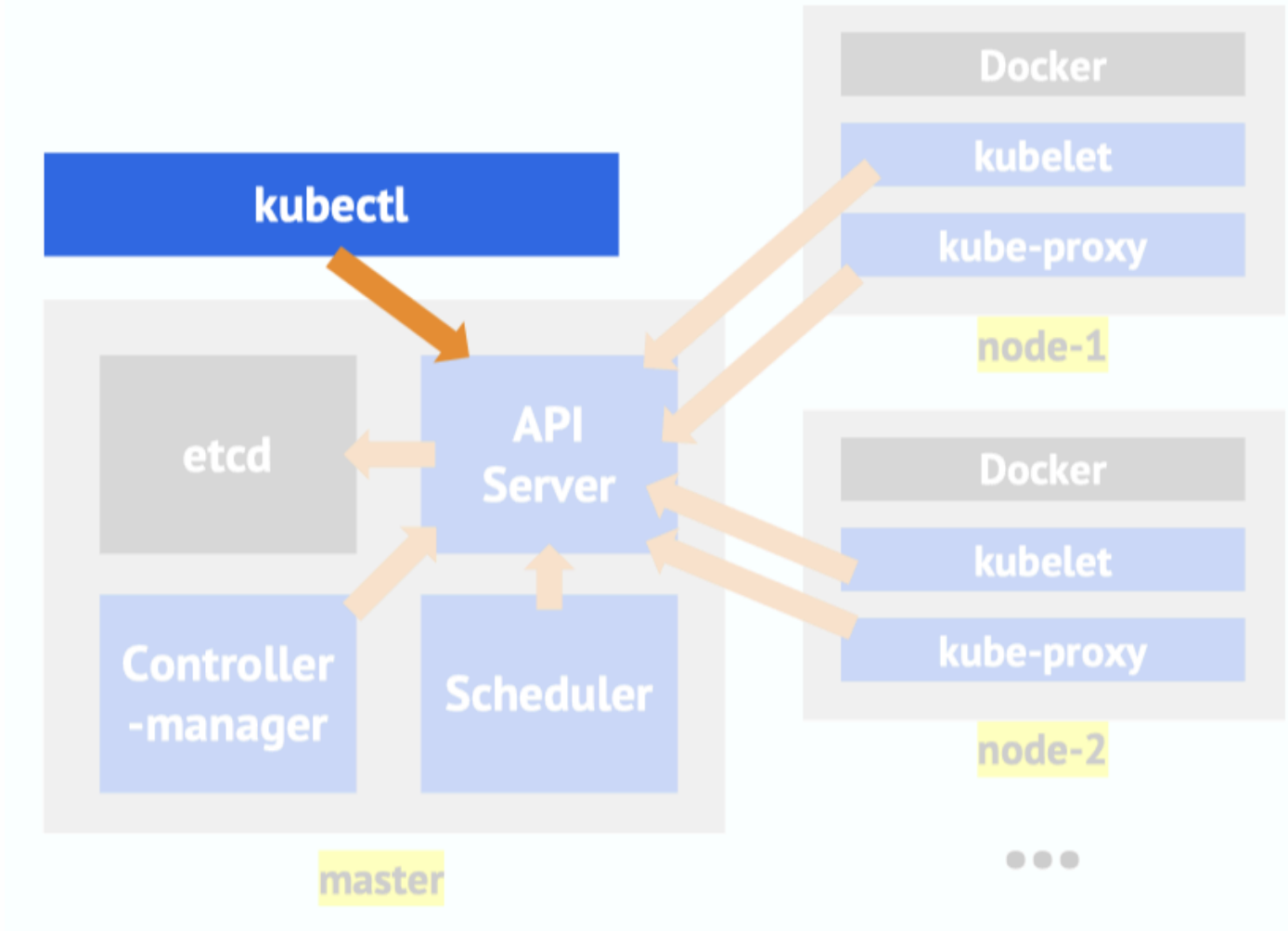
https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Архитектура Kubernetes



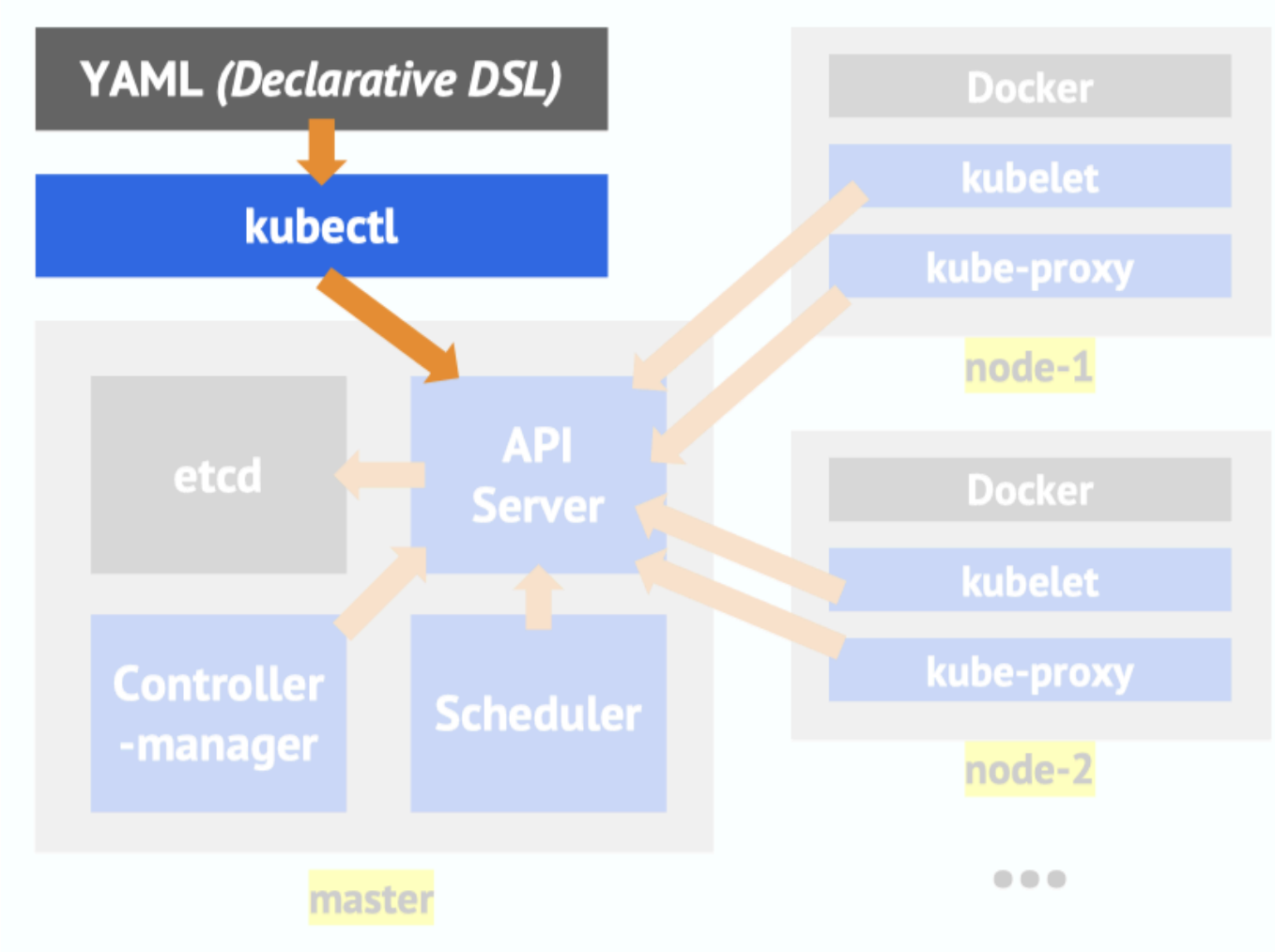
https://www.youtube.com/watch?v=CgCLPYJRxbU

Архитектура Kubernetes



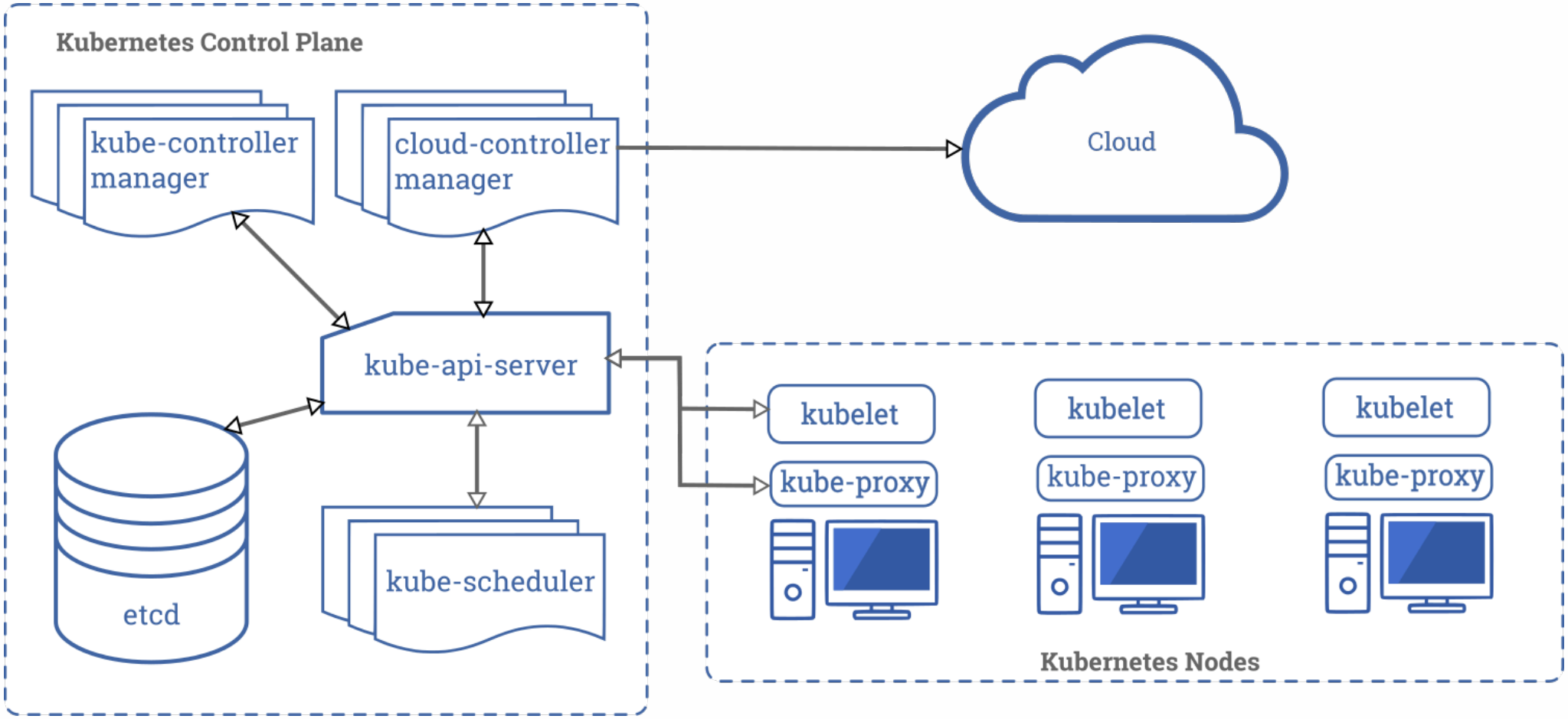
https://www.youtube.com/watch?v=CgCLPYJRxbU

Архитектура Kubernetes



https://www.youtube.com/watch?v=CgCLPYJRxbU

Архитектура Kubernetes



https://kubernetes.io/docs/concepts/architecture/nodes/

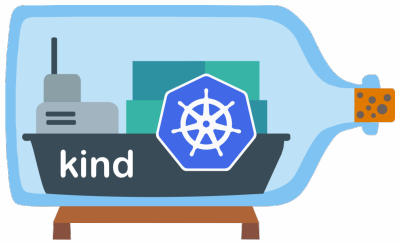
Kubernetes options

Локальные инсталляции K8S:

- minikube
- k3s
- kind
- Micro8ks

Облачные инсталляция K8S:

- GKE (Google)
- AKS (Azure)
- EKS (Amazon)



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

02

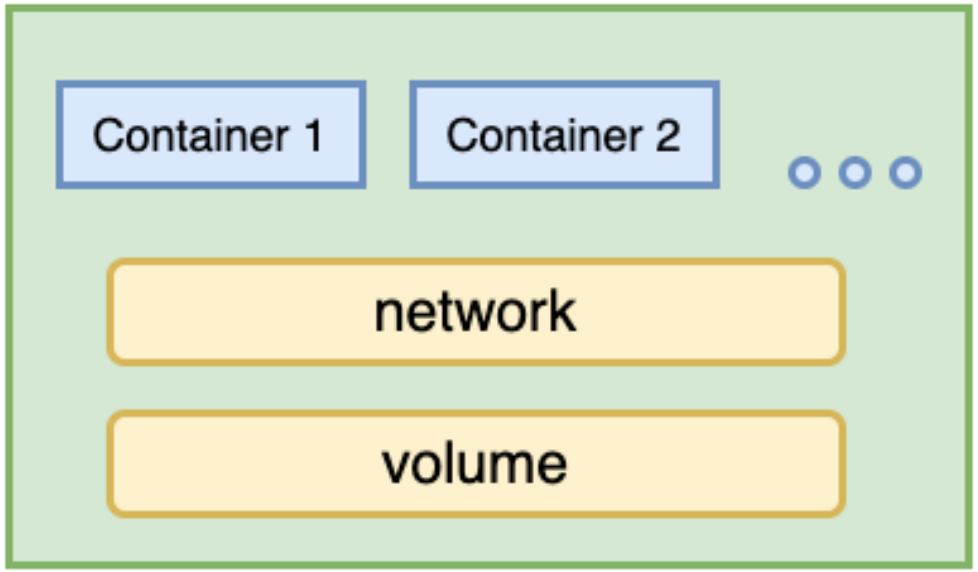
Основные сущности kubernetes

Основные сущности Kubernetes

Pod

Pod – это один или несколько контейнеров, которые разделяют единый сетевой интерфейс и шарят общие ресурсы.

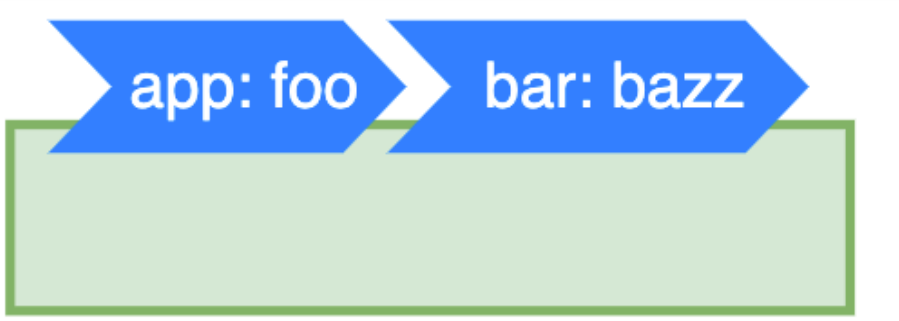
Контейнеры внутри пода могу обращаться друг к другу через localhost.



Labels & Selectors

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

Метки – это значения типа key: value
Селекторы – это выражения, позволяющие выбрать сущности по меткам



https://kubernetes.io/ru/docs/concepts/overview/working-with-objects/labels/

Labels & Selectors

env == prod

app in (foo, bar)

version != 1.5.7

https://kubernetes.io/ru/docs/concepts/overview/working-with-objects/labels/

ReplicaSet

ReplicaSet – это контроллер, который обеспечивает, что поды запущены ровно в указанном количестве экземпляров.

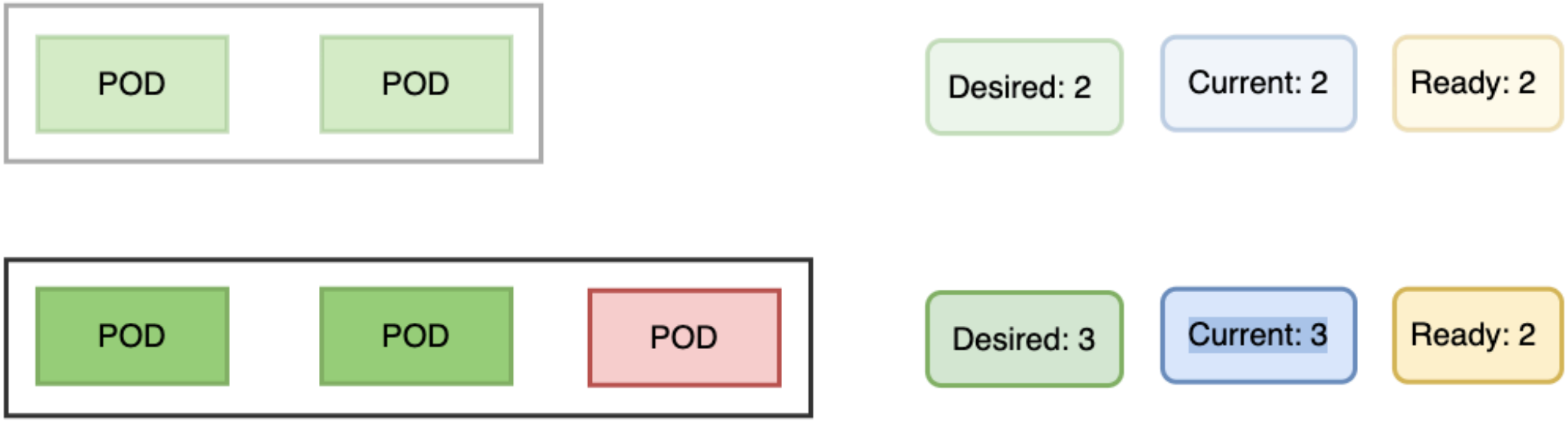
ReplicaSet

Desired: 2

Current: 2

Ready: 2

ReplicaSet

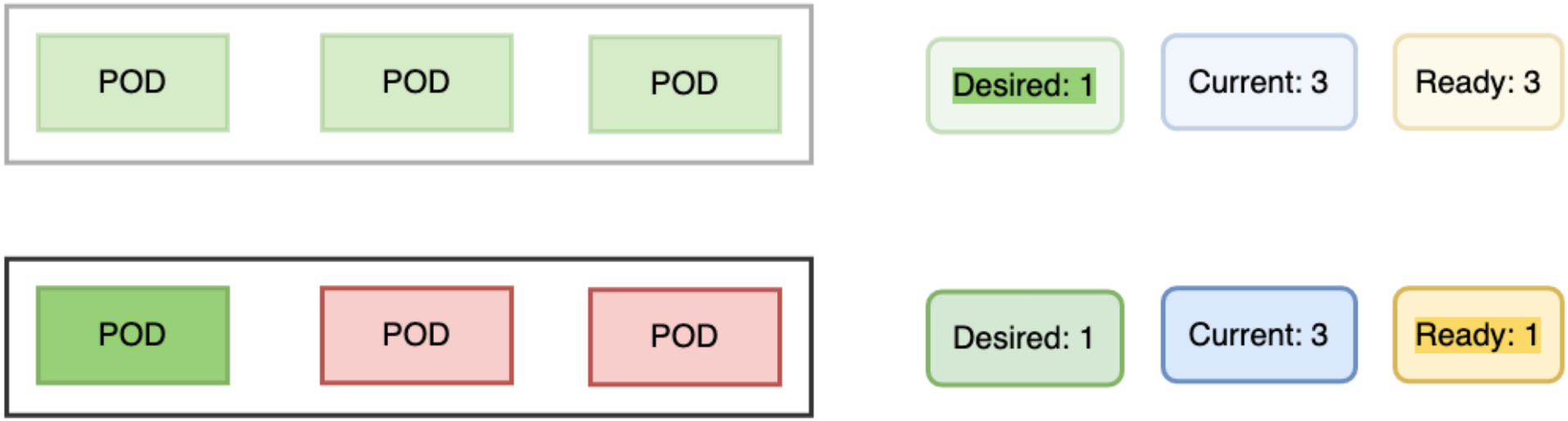

POD
POD
Desired: 2
Current: 2
Ready: 2
POD
POD
POD
Desired: 3
Current: 3
Ready: 2

ReplicaSet

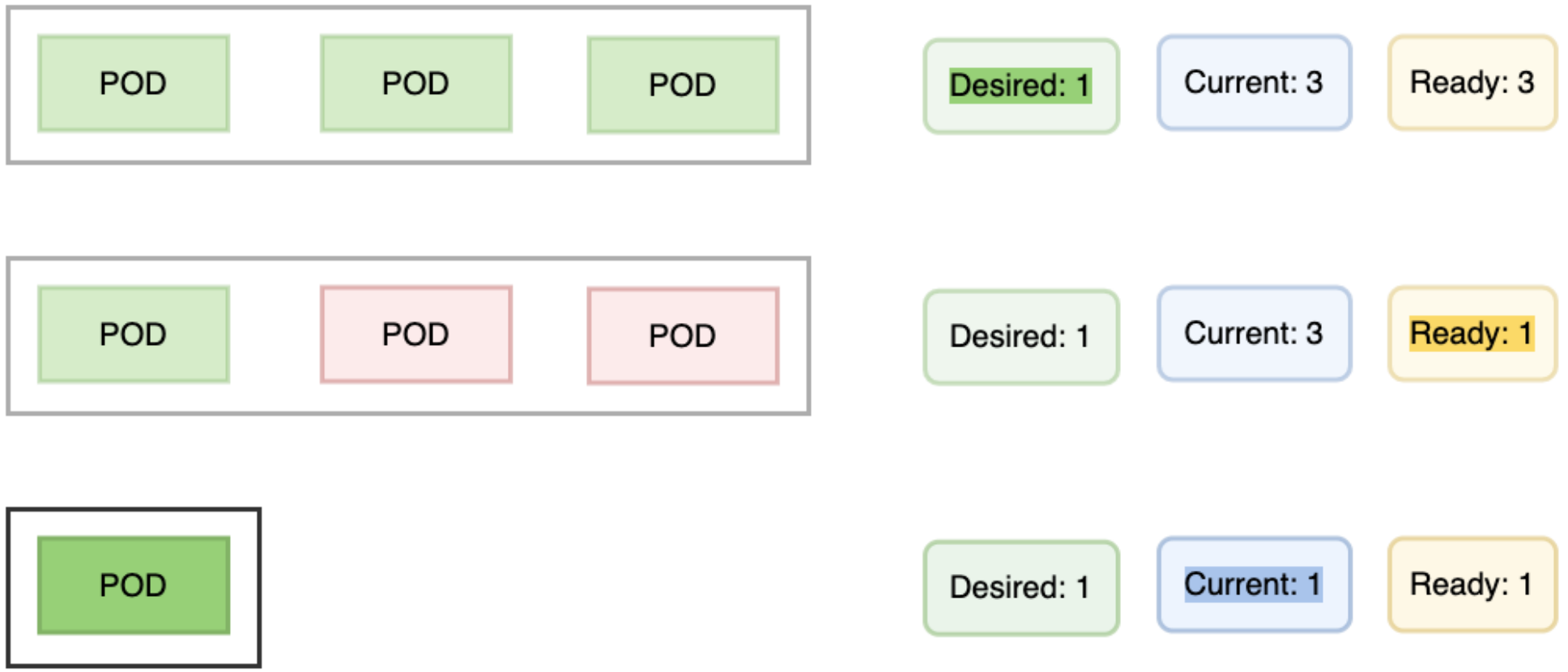

POD
POD
Desired: 2
Current: 2
Ready: 2
POD
POD
POD
Desired: 3
Current: 3
Ready: 2
POD
POD
POD
Desired: 3
Current: 3
Ready: 3

ReplicaSet

ReplicaSet


POD
POD
POD
Desired: 1
Current: 3
Ready: 3
POD
POD
POD
Desired: 1
Current: 3
Ready: 1

ReplicaSet


POD
POD
POD
Desired: 1
Current: 3
Ready: 3
POD
POD
POD
Desired: 1
Current: 3
Ready: 1
POD
Desired: 1
Current: 1
Ready: 1

Deployment

Deployment – это контроллер, который позволяет не только следить за количеством реплик, но также позволяет обновлять версию и в случае ошибок откатить или остановить раскладку.

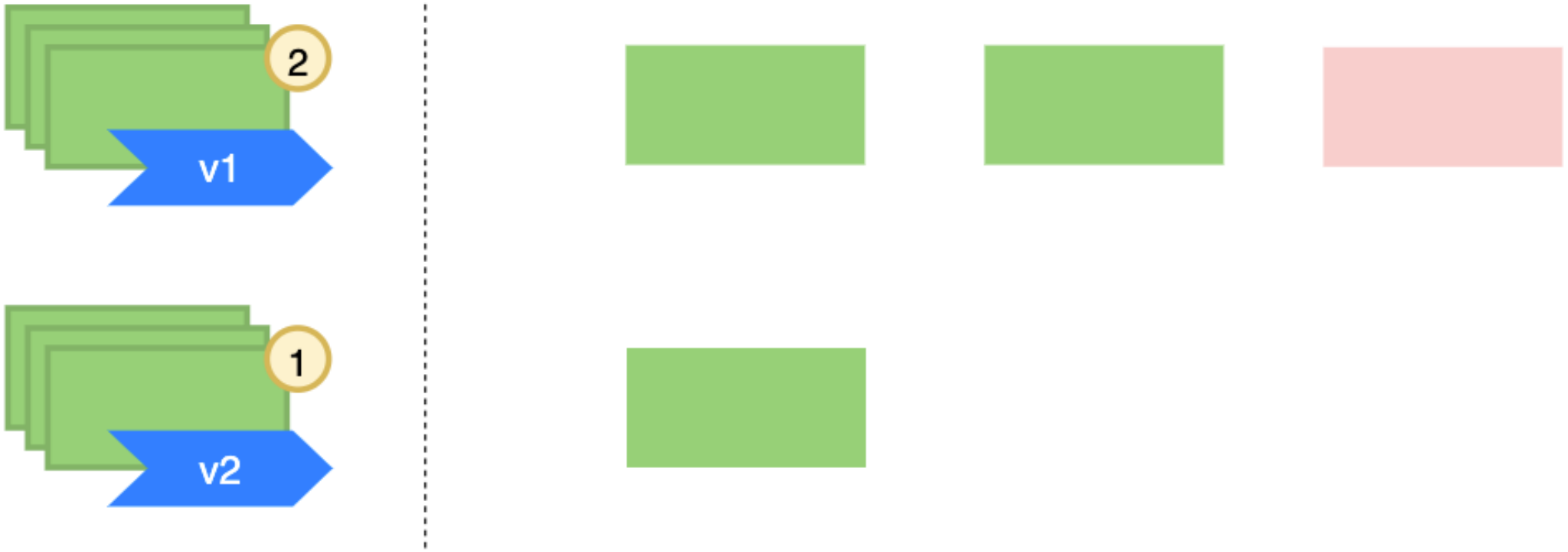
https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Deployment

Deployment

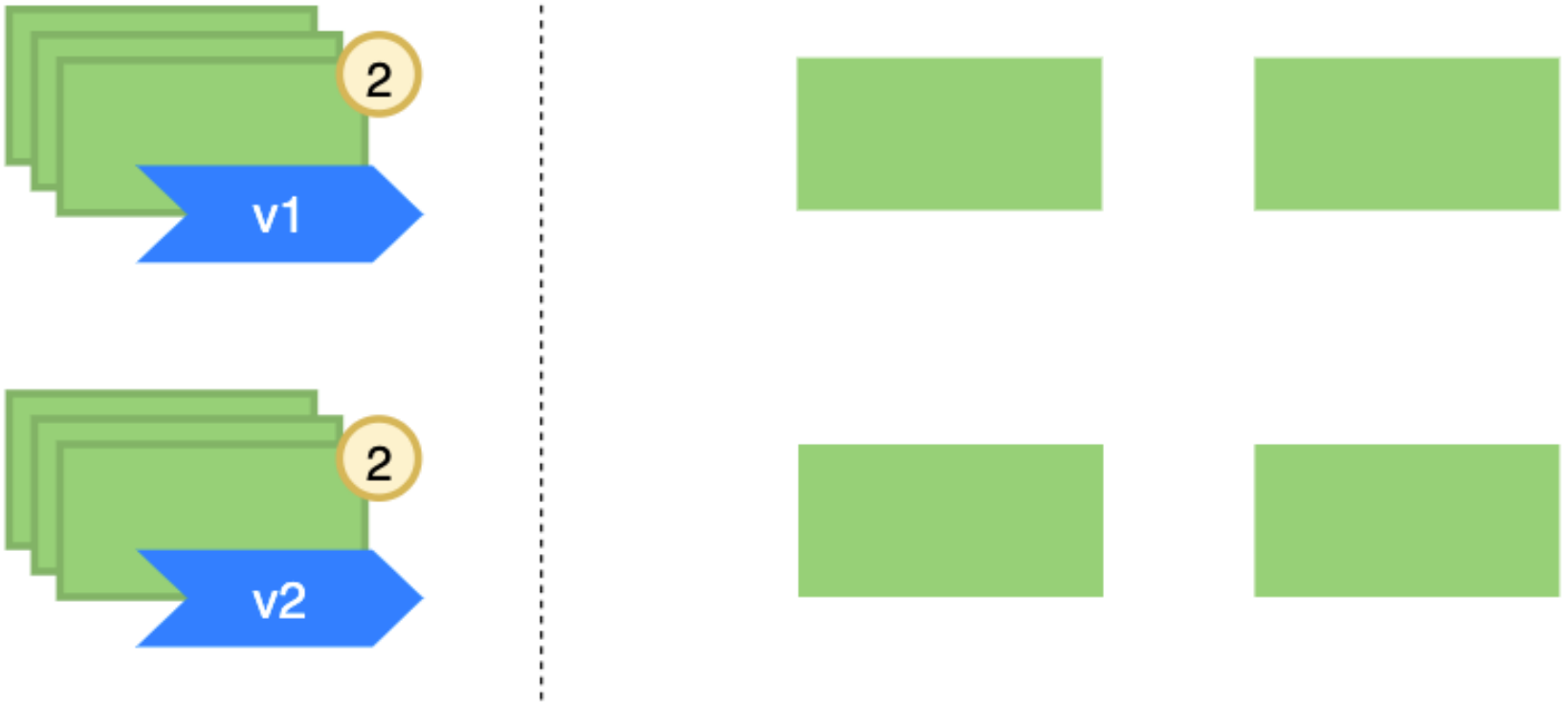
Deployment

Deployment

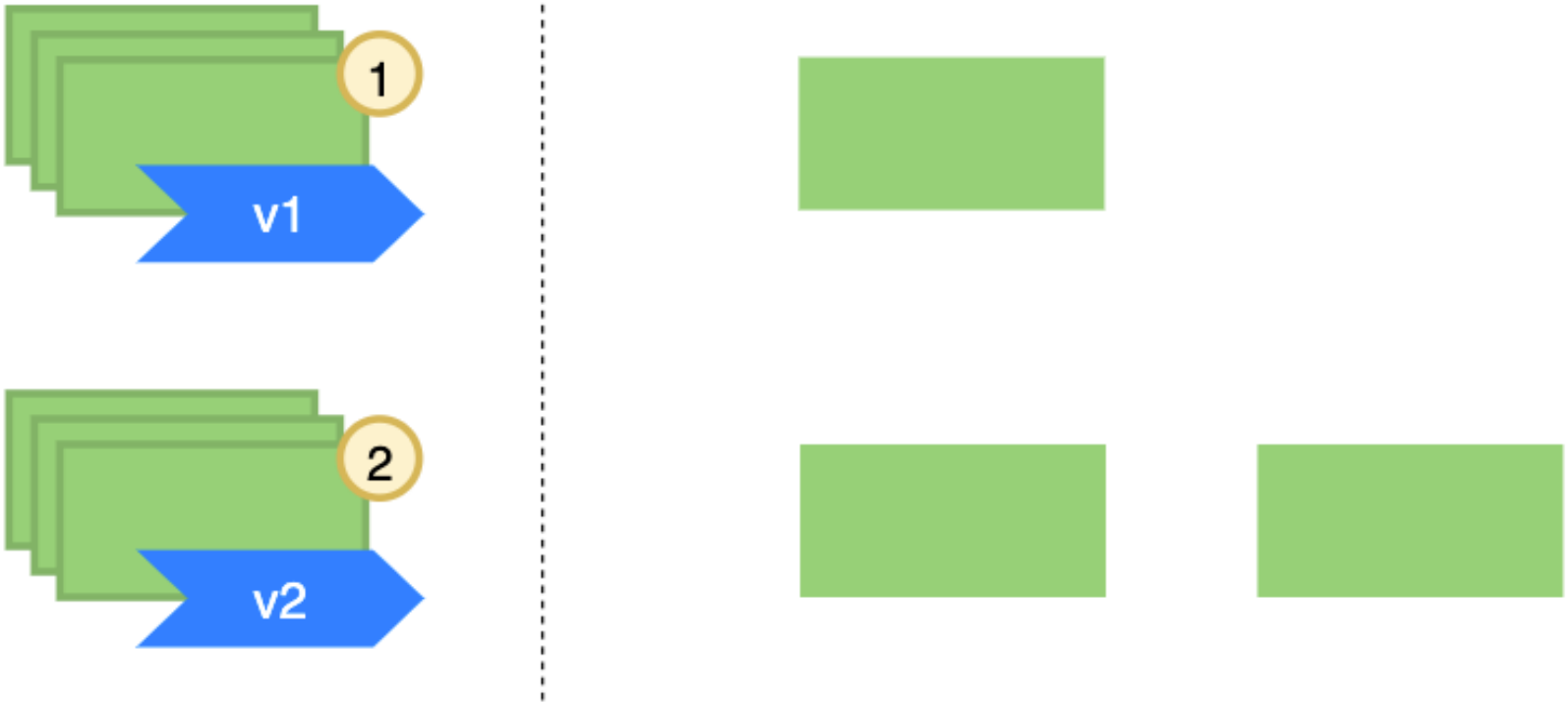


Deployment

Deployment



Deployment



Deployment

Deployment

Deployment

Service

Service – это ресурс (абстракция), которая позволяет обращаться к набору работающих подов, как к сетевому сервису.

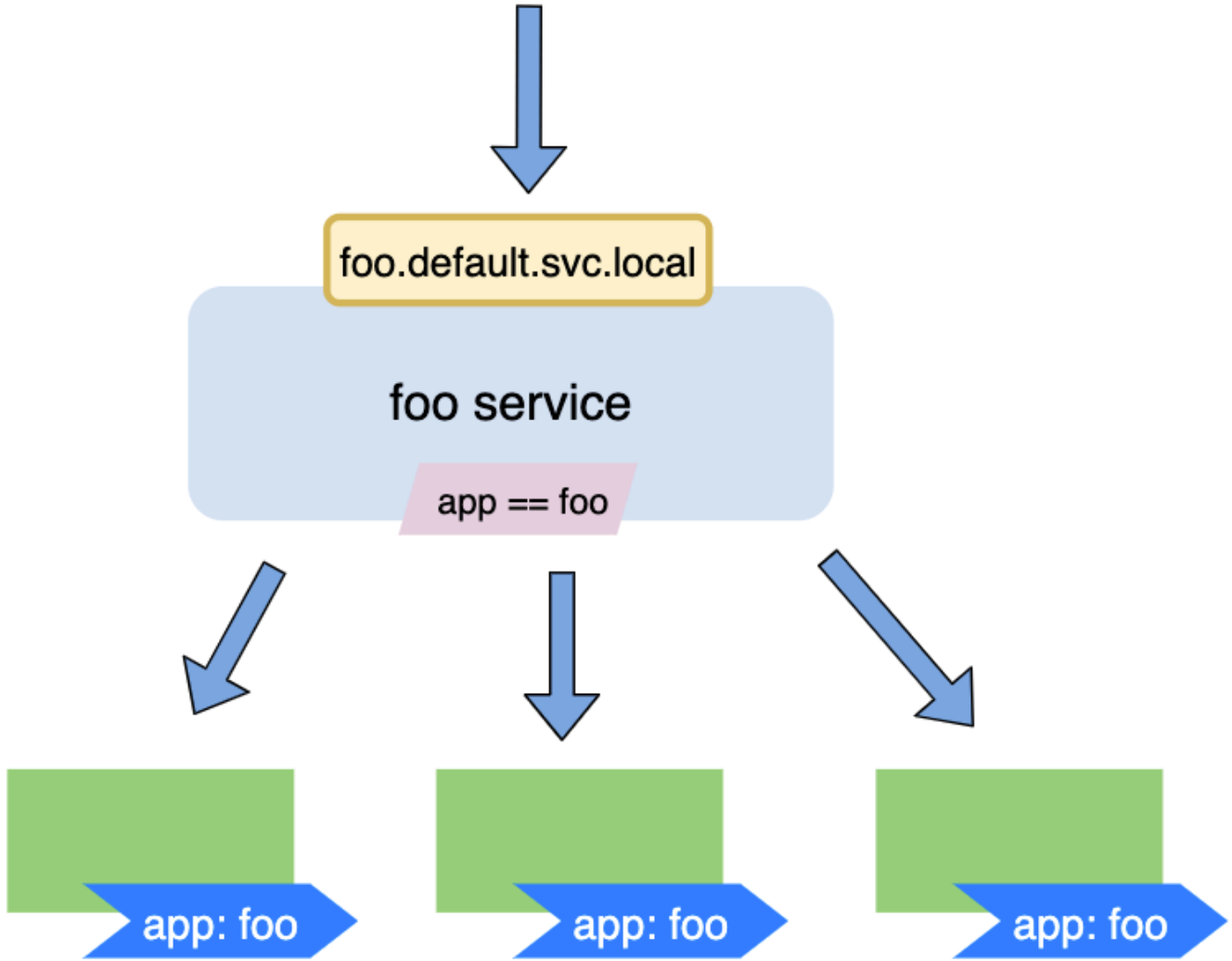
На каждый сервис выделяется конкретный IP адрес и набор портов, доступных для обращения

Каждому сервису выделяется локальное DNS имя.

Набор подов определяется селектором.

https://kubernetes.io/docs/concepts/services-networking/service/

Service



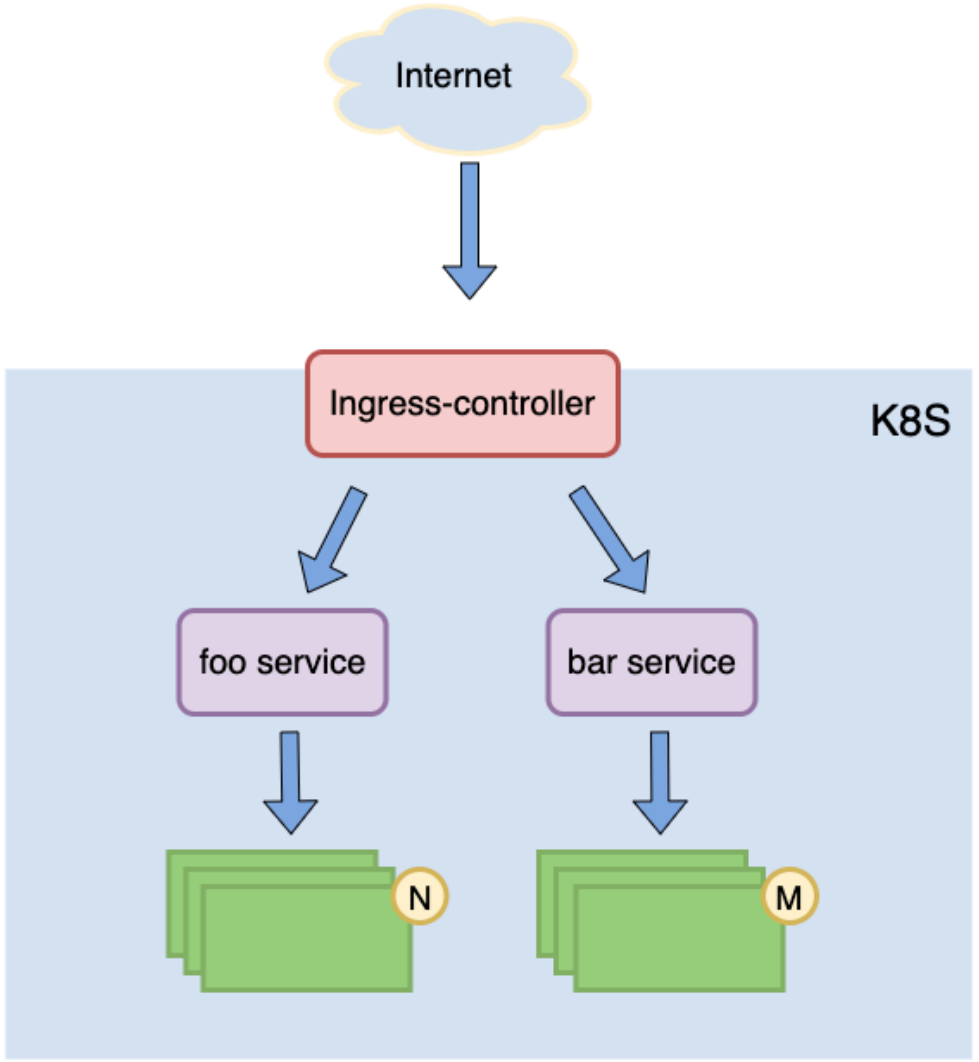
https://kubernetes.io/docs/concepts/services-networking/service/

Ingress

Ingress – это ресурс (абстракция), которая определяет каким образом внешний клиентский трафик роутится на сервисы.

https://kubernetes.io/docs/concepts/services-networking/ingress/

Ingress



https://kubernetes.io/docs/concepts/services-networking/ingress/

PersistentVolume

Persistent Volume – это ресурс, который описывает доступный для использования внутри кластера k8s раздел с хранилищем.

Persistent Volume Claim – это ресурс, который описывает запрос на получение доступа к хранилищу.

Provisioner – это контроллер, который слушает создание новых запросов на хранилище, и создает persistent volume, которые нужны хранилищу.

https://kubernetes.io/docs/concepts/services-networking/ingress/

Volume

Volume имеет обычно следующие характеристики:

- Размер (size)
- Тип доступа (acessType): readOnly, readWriteOnce
- Класс хранилища (storageClass): slow, normal, fast

https://kubernetes.io/docs/concepts/services-networking/ingress/

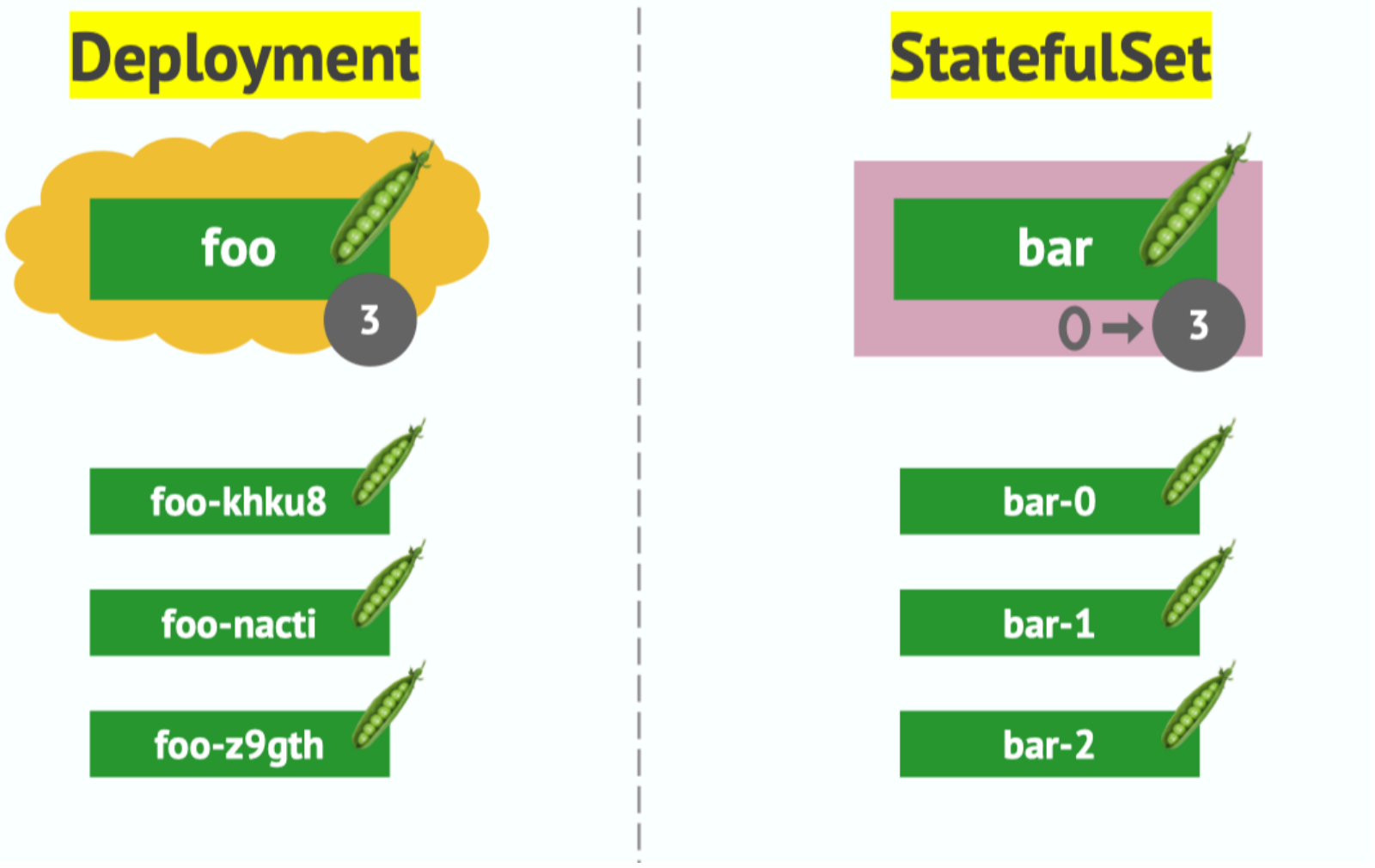
StatefulSet

StatefulSet – это ресурс (контроллер), который дает некоторые гарантии, необходимые для работы stateful приложений.

- Порядок создания. Поды в statefulset-е создаются последовательно от 0 до N-1. При удалении удаляются в последовательно в обратном порядке от N-1 до 0.
- Числовой идентификатор пода. Все поды в statefulset-е пронумерованы от 0 до N-1.
- Имя пода. У подов всегда стабильные имена с шаблоном $(statefulset name)-$(ordinal)
- Каждому поду выделяется отдельное хранилище.
- К каждому поду можно обращаться по DNS.
- В случае удаления пода, хранилище остается.

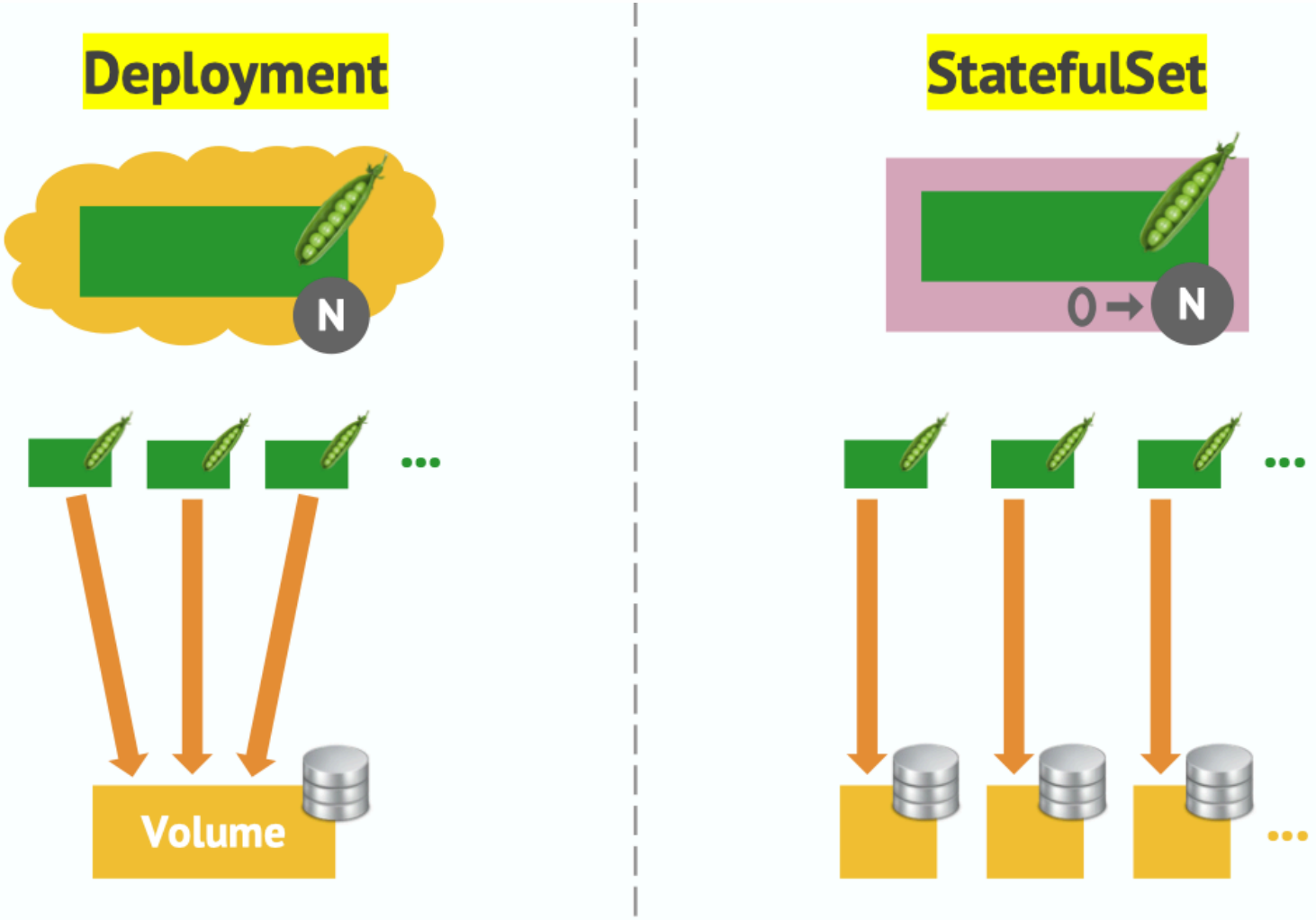
https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

StatefulSet



https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

StatefulSet



https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

Job

Job – это ресурс (контроллер), который дает возможность выполнения одноразовых задач (например, миграция)

Основные параметры job-а

- backoffLimit – это лимит рестартов задачи

https://kubernetes.io/docs/concepts/workloads/controllers/job/

Спасибо
за внимание!