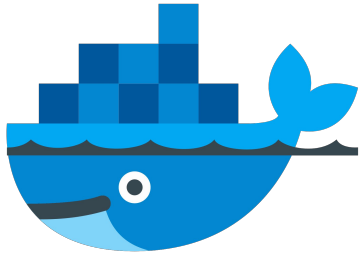
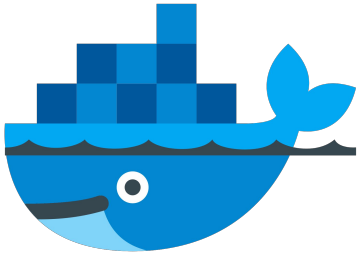
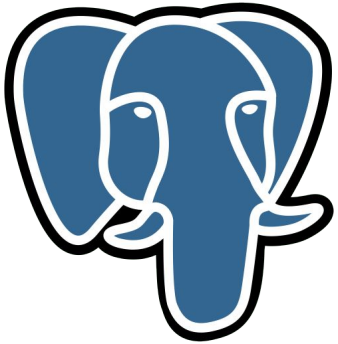
Кубер на своём железе - плюсы, минусы, котики.

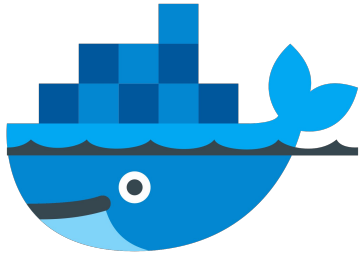
Дехтярёв Евгений

2GIS

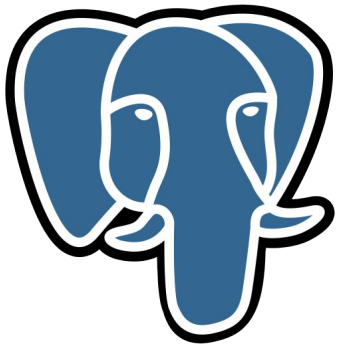


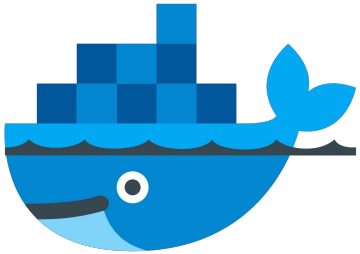




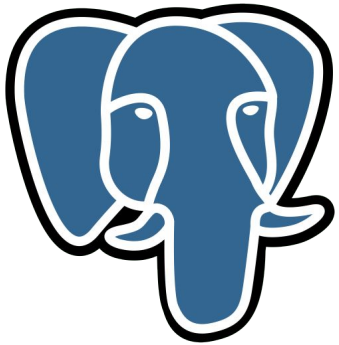


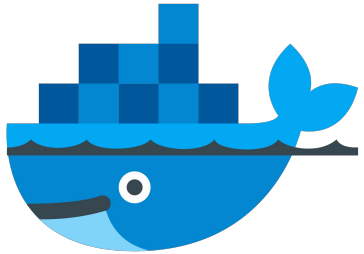

Stack



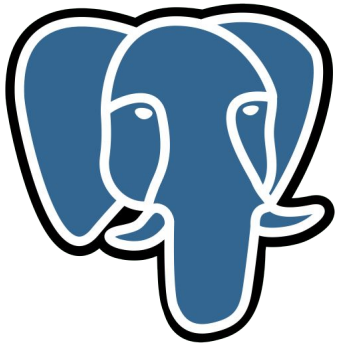


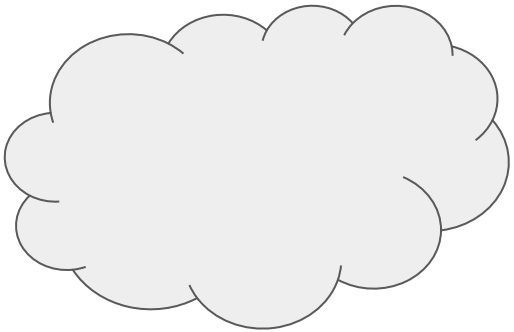

Stack

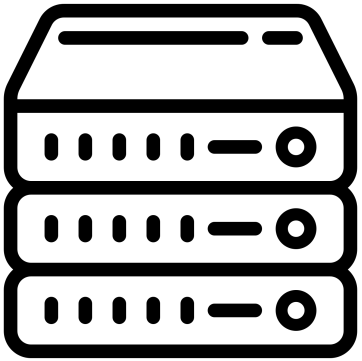


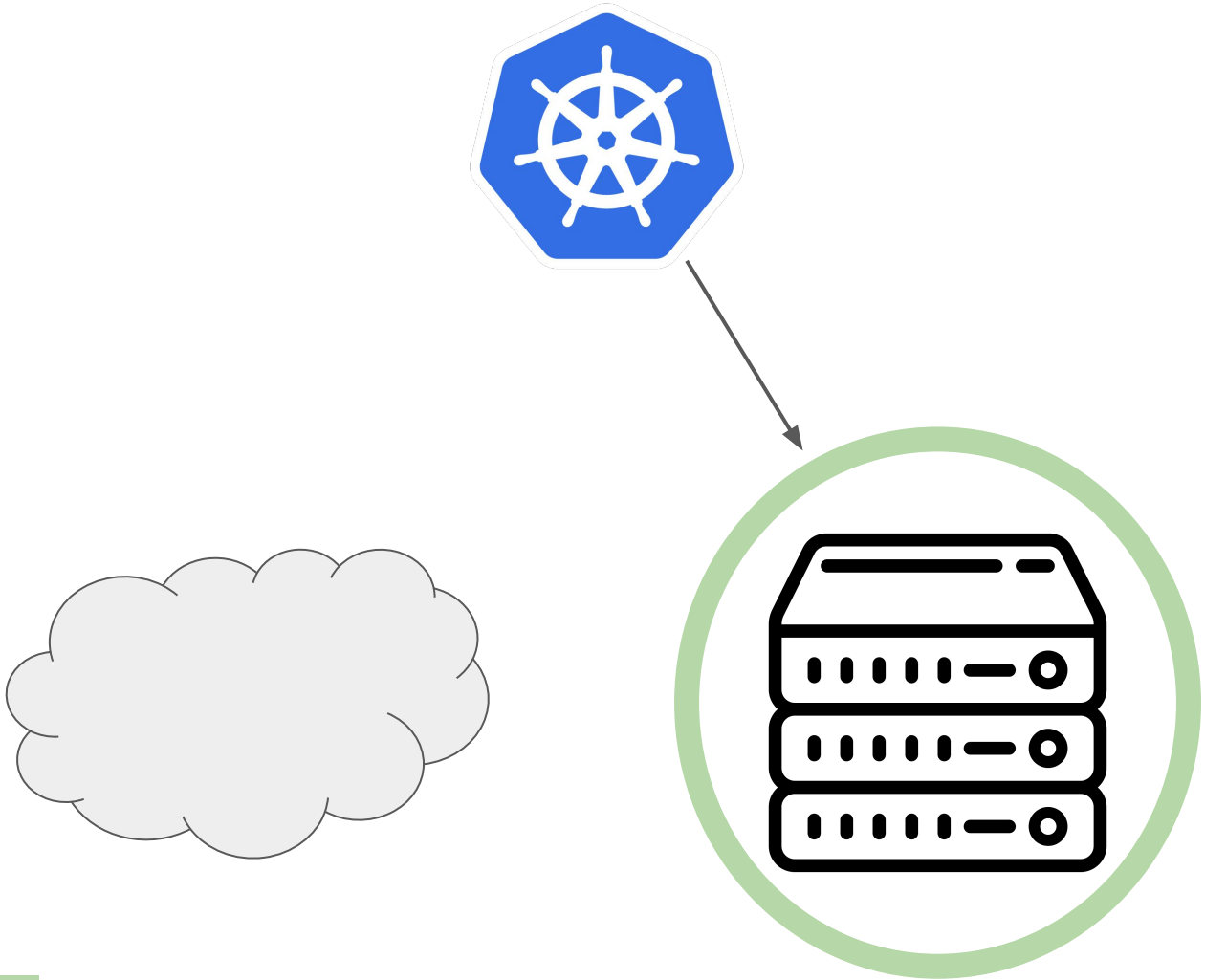



ELK
Stack

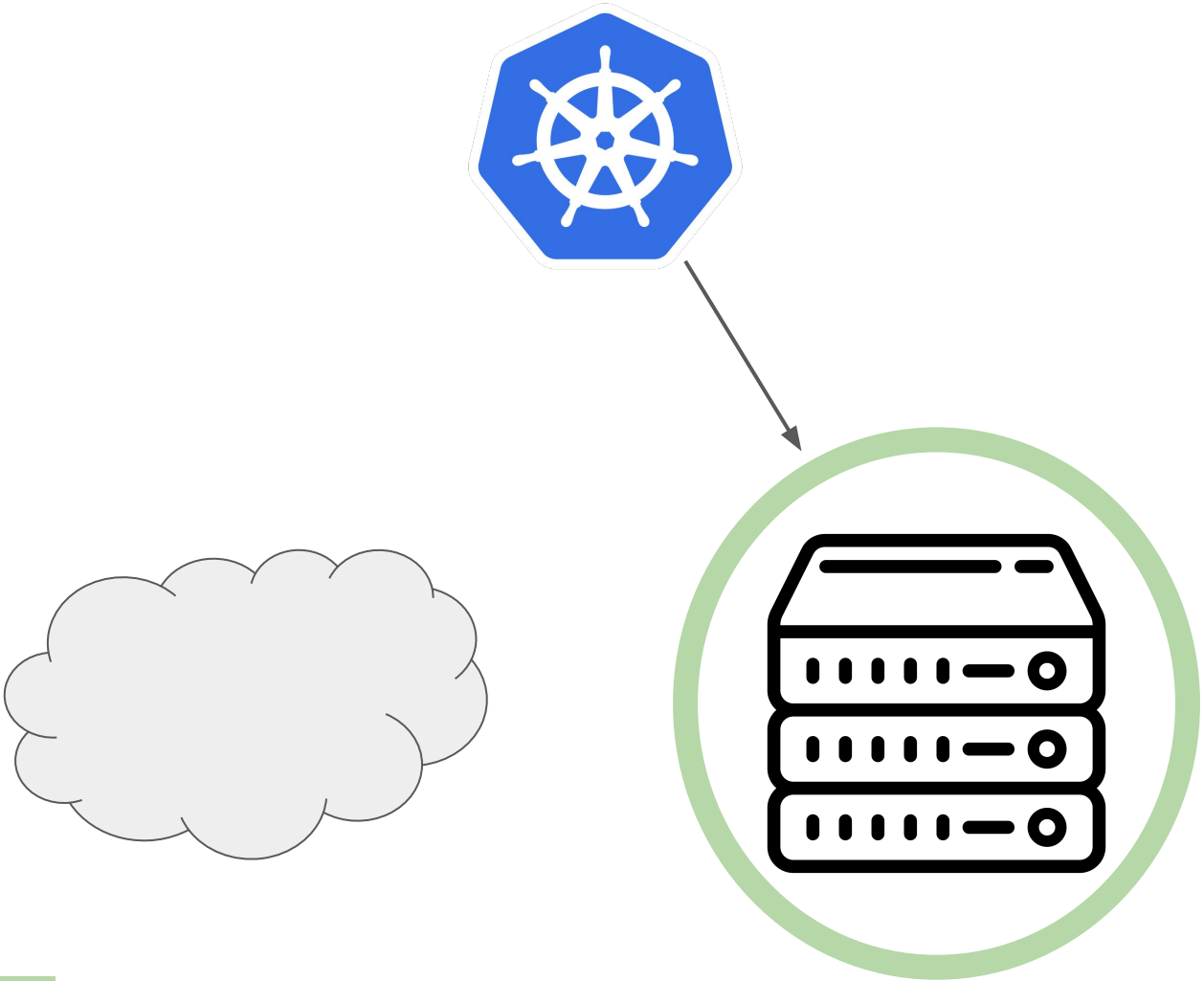






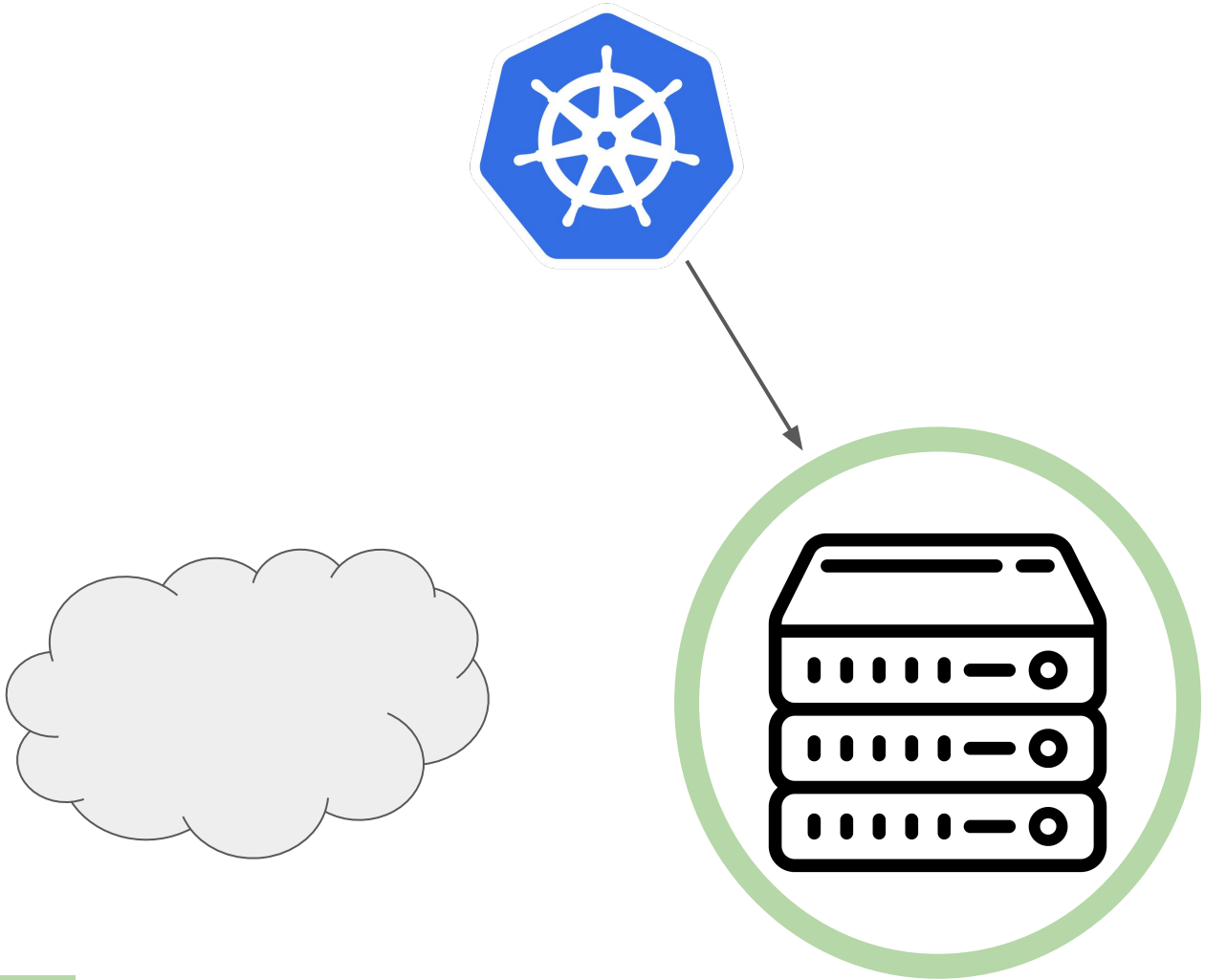


Поговорим



Поговорим

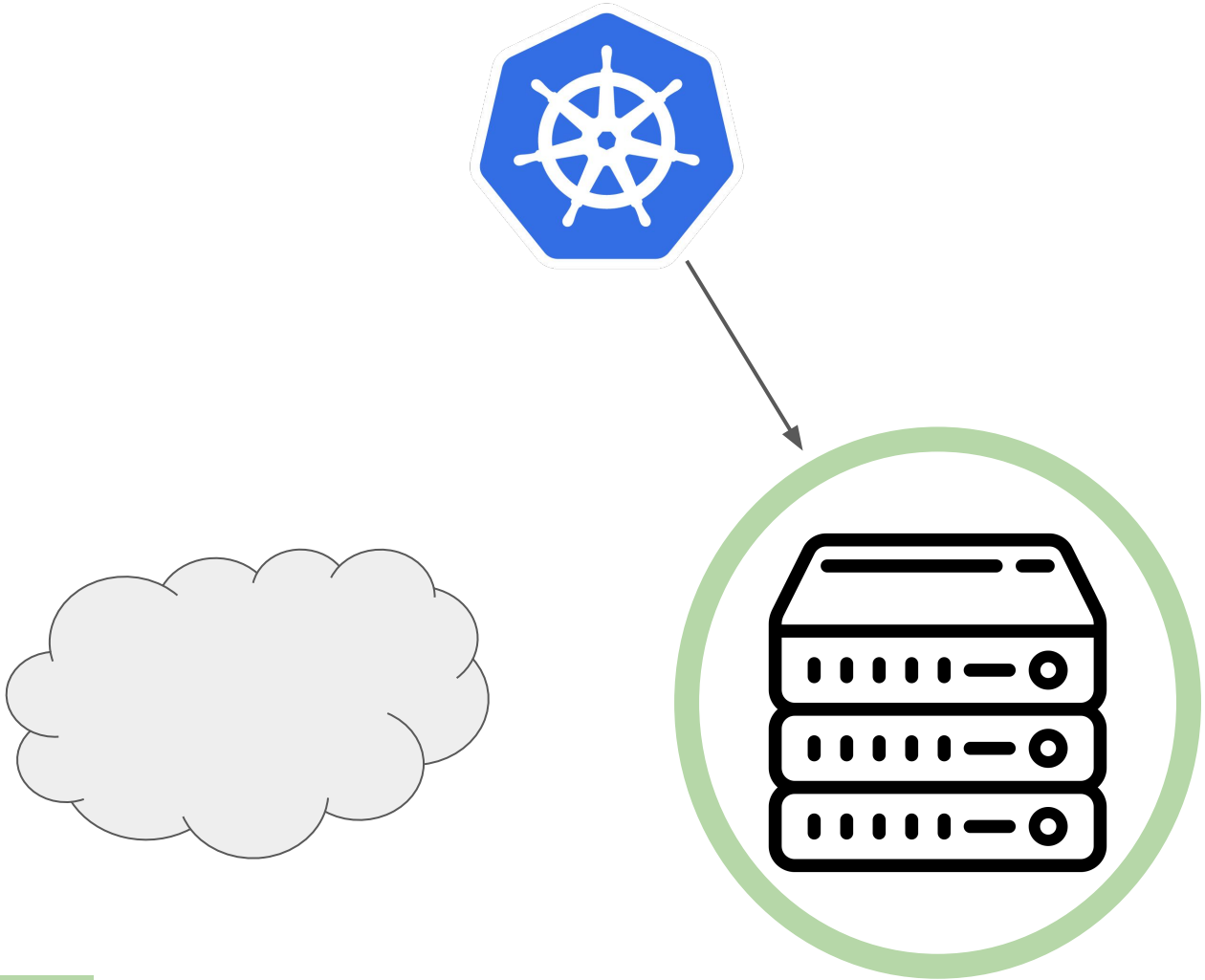
Наше железо



Поговорим

Наше железо

Проблемы железа

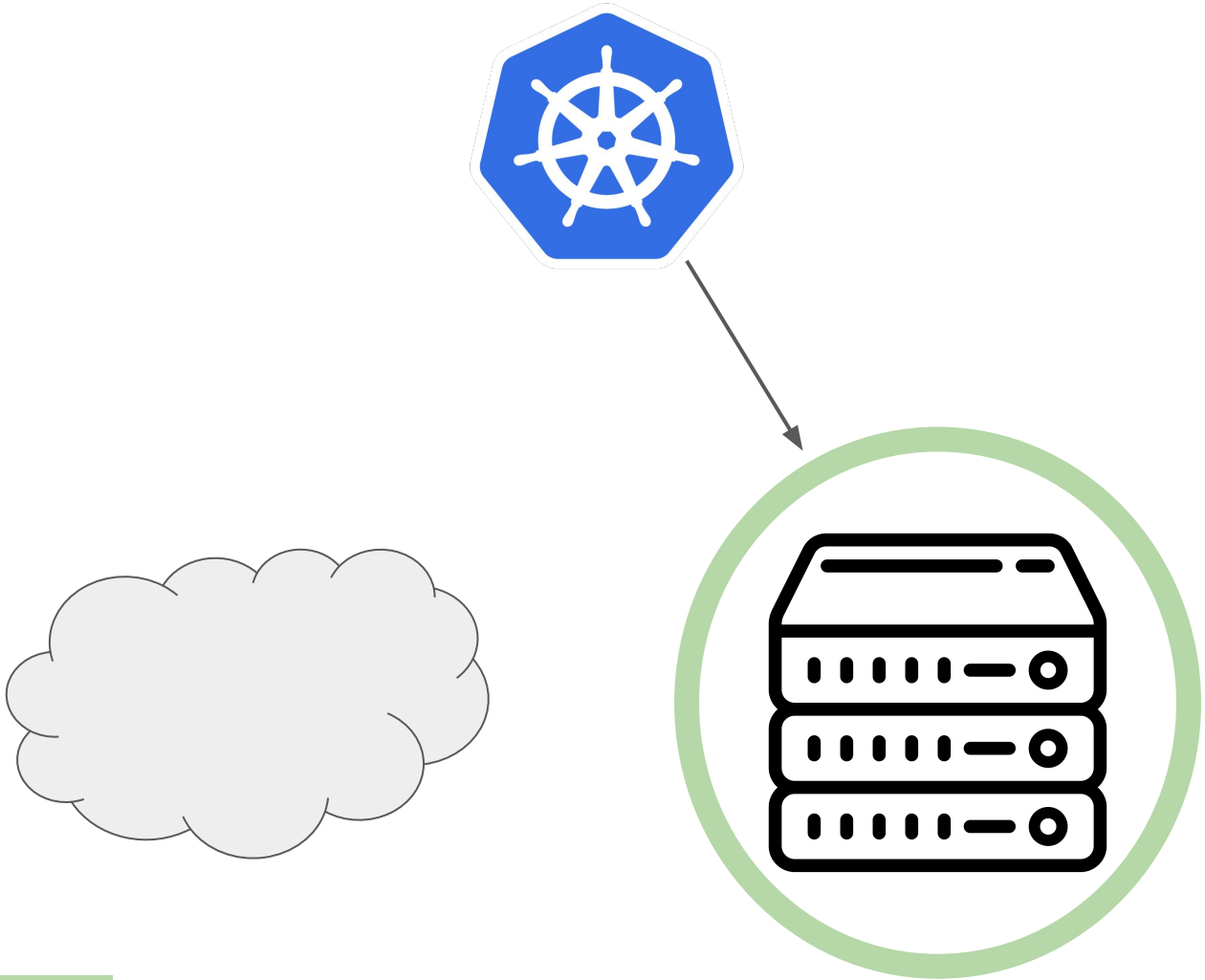


Поговорим

Наше железо

Проблемы железа

Как решать



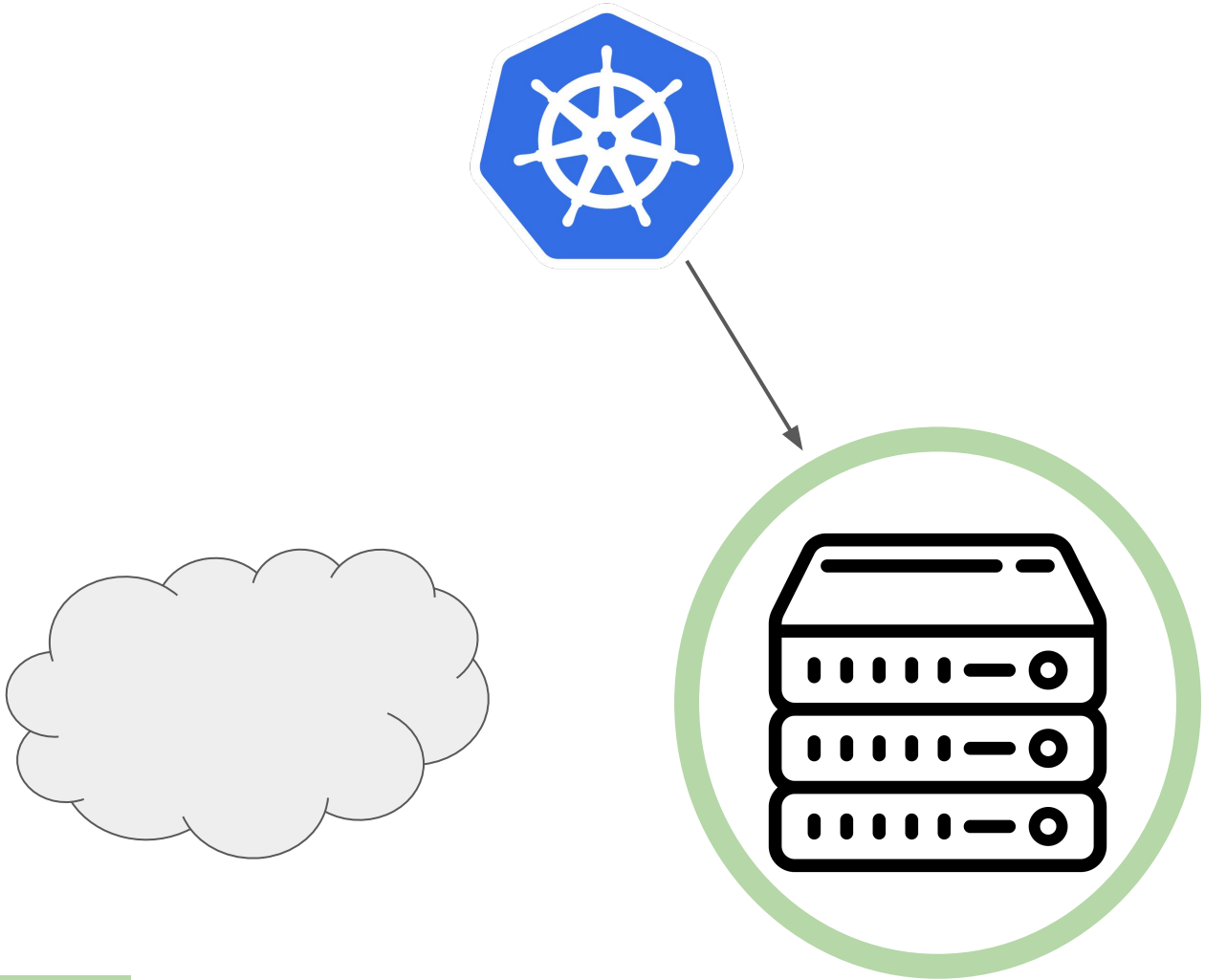
Поговорим

Наше железо

Проблемы железа

Как решать

Как планировать



Поговорим

Наше железо

Проблемы железа

Как решать

Как планировать

Как считать

Почему железо

Очень много пользователей за ~~МКАДом~~ Уралом (начинали с регионов)

- очень мало ДЦ за Уралом

Ист~~О~~**Е**рически сложилось

- + требования компании

Какое железо

Для облака(*Openstack*)

- 40+ CPU / 512+ RAM / 2Tb SSD

Для K8s

- 8-16 CPU / 32-64 RAM / 2x480Gb SSD

Новое железо

Новый трафик

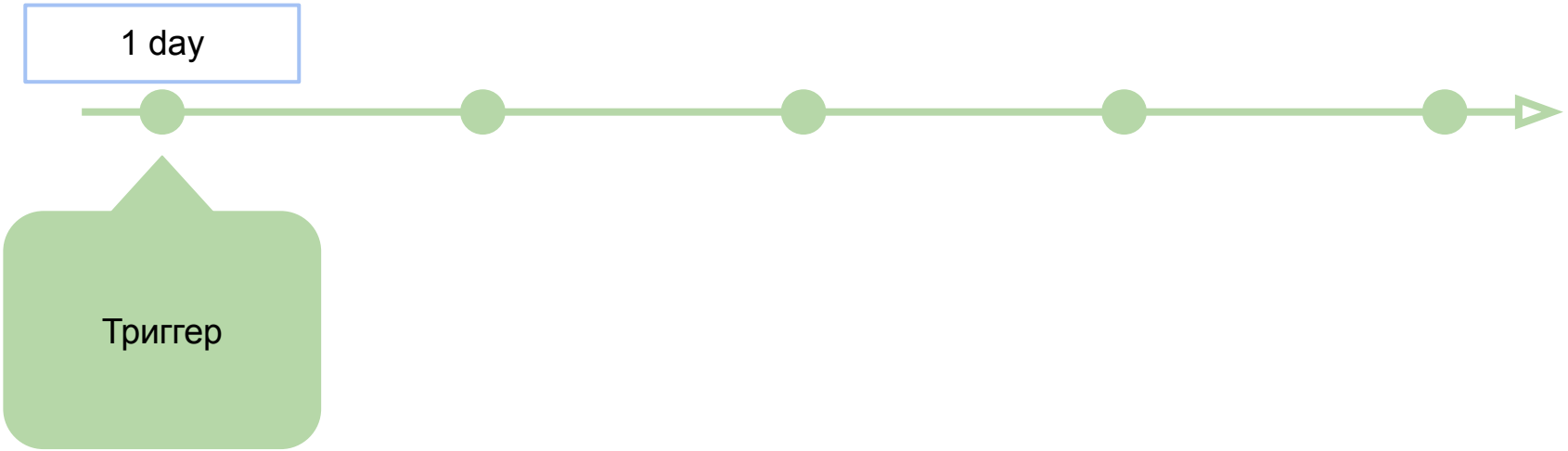
Новые приложения

Резервирование

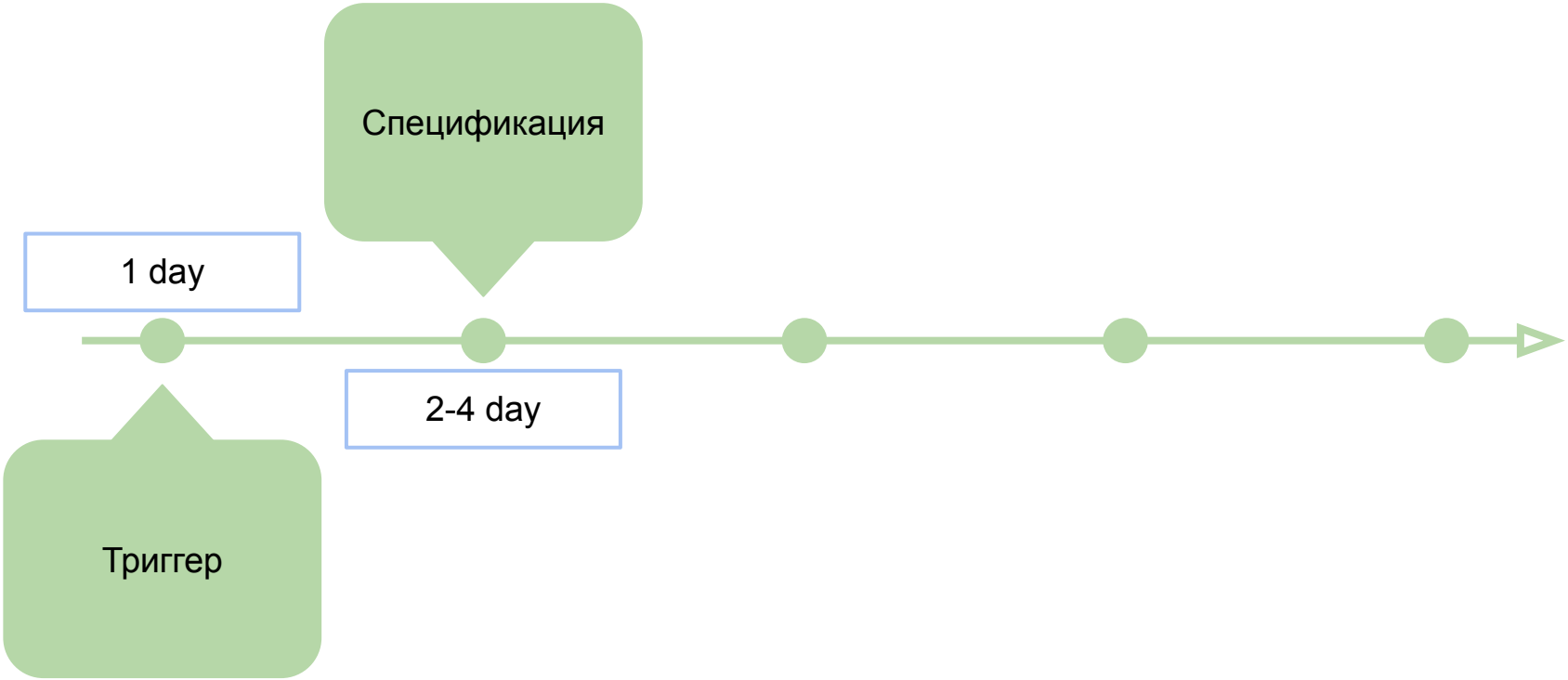
Проблема заказа

Новое железо - **когда пора?**

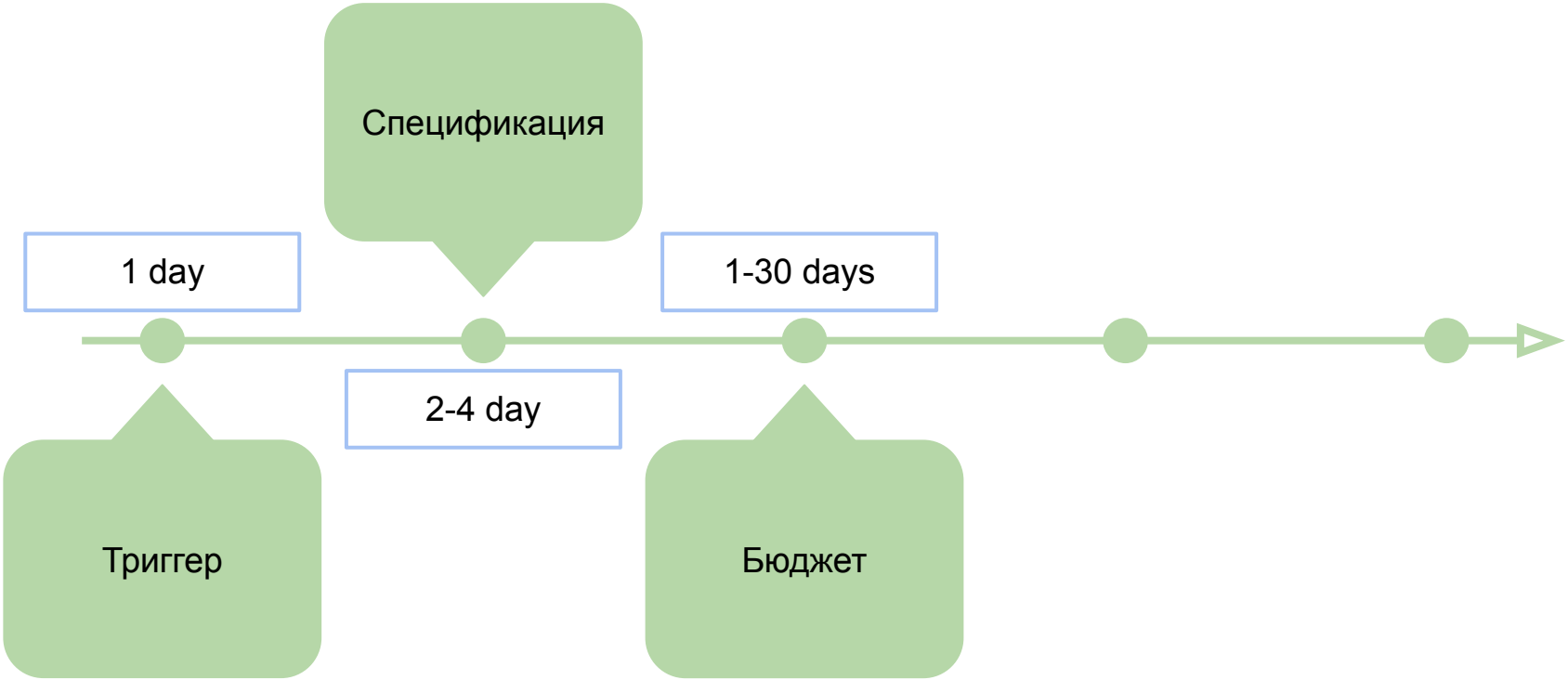
Новое железо - когда пора?



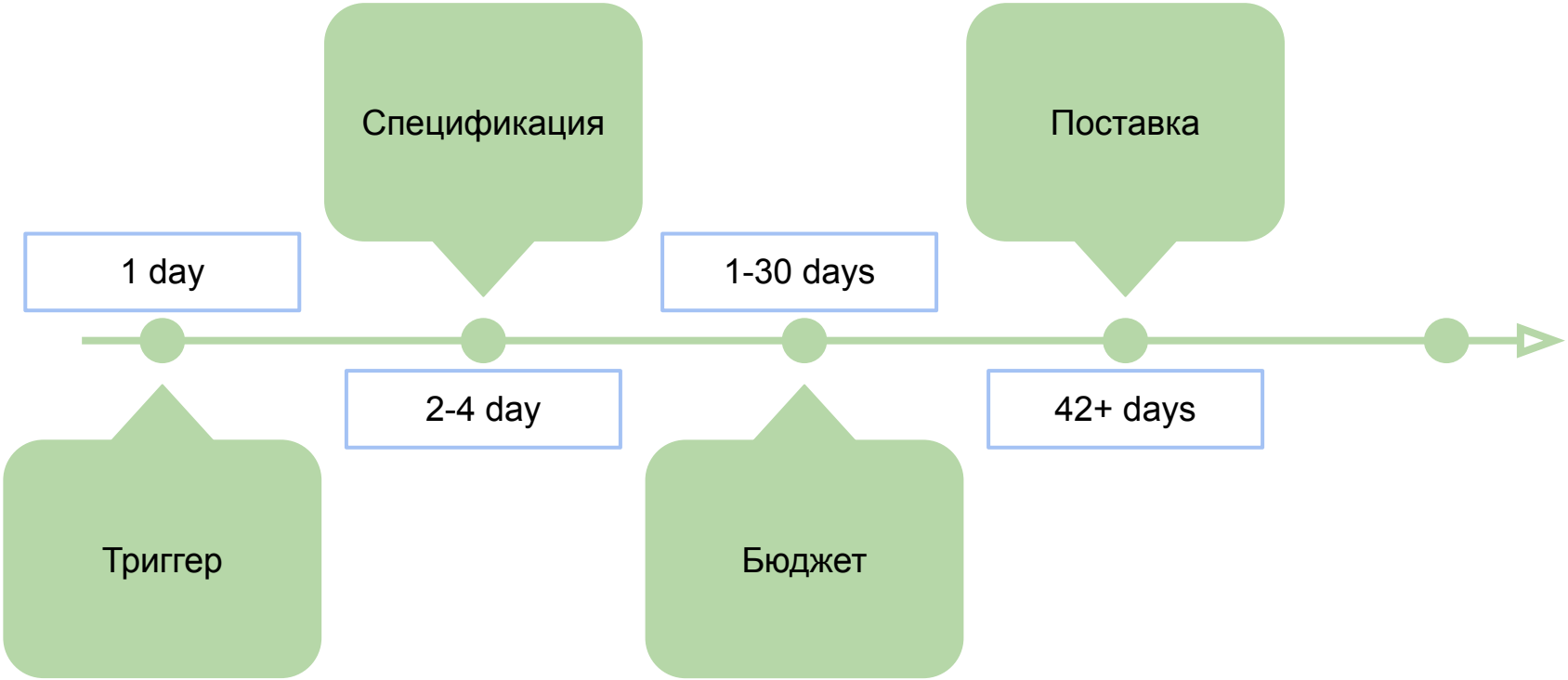
Новое железо - когда пора?



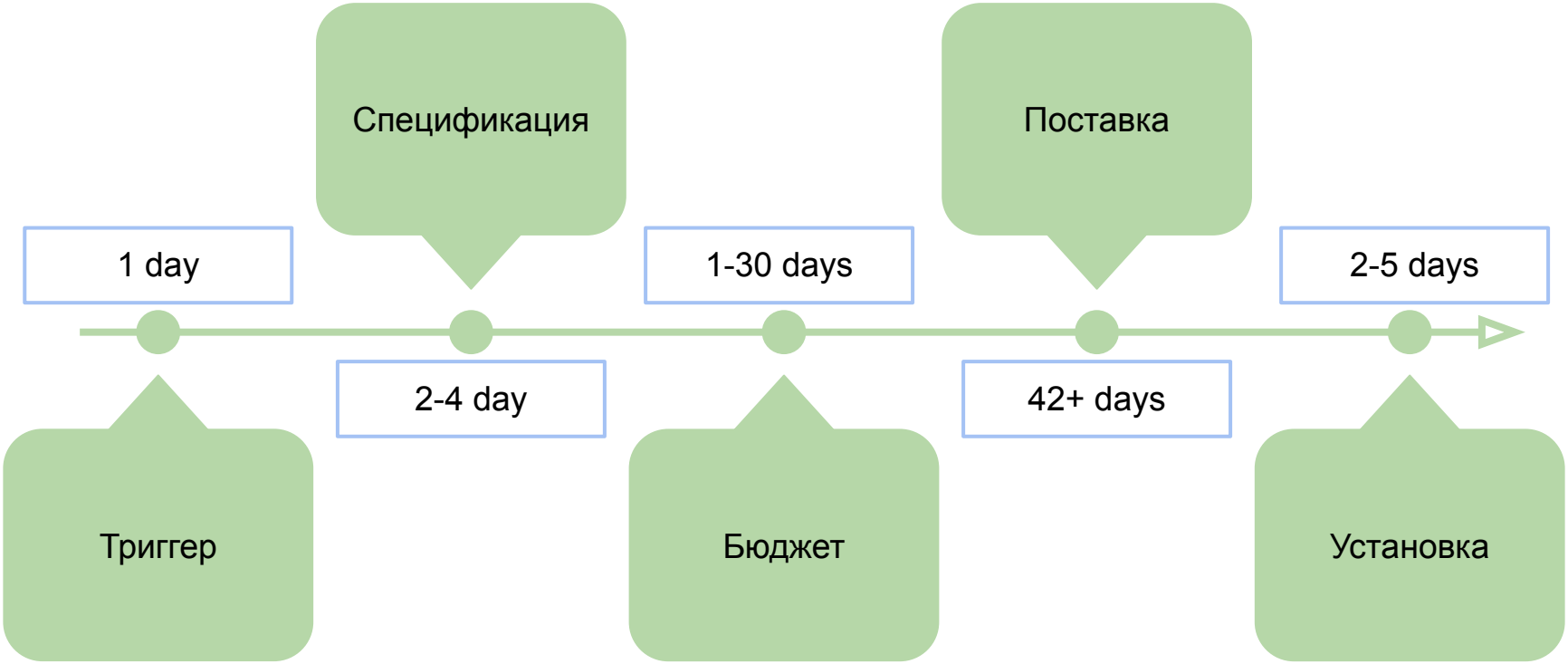
Новое железо - когда пора?



Новое железо - когда пора?



Новое железо - когда пора?



Новое железо - когда пора?

Железо закончилось - сервисы нет

Железо будет через 2 месяца, а продакт принёс тебе 50 микросервисов.
Что делать?

- **Потушить пожар в моменте**
- **Postmortem, чтобы определить дальнейшие шаги**

Тушим пожар

Определить

- Что закончилось?
- Сколько требуется?

Вместимость(Capacity) кластера k8s зависит от квоты на requests(cpu/mem)

- CPU - самый дефицитный ресурс
- RAM - дешевле и её меньше требуется для микросервисов
    - *У нас много Python/Golang/JS приложений и меньше на Java/Scala*

Конкретно тушим пожар #1

Быстрый способ - **воспользоваться заначкой** (*Openstack*)

Деплой готов, очень быстро, поднять ещё воркеров k8s

- Строка в деплое == 1 воркер k8s

Помните - это временное решение

- Кризис миновал --> вернули ресурсы в заначку

Конкретно тушим пожар #2

Ревизия выданных requests.cpu

- 100m requests.cpu на контейнер в 2016

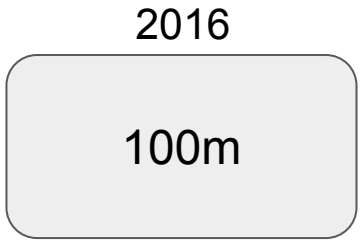
Это оверхед, как 5% резерва на рутовом диске при 4Тб SSD

Конкретно тушим пожар #2

Ревизия выданных requests.cpu

- 100m requests.cpu на контейнер в 2016

Это оверхед, как 5% резерва на рутовом диске при 4Тб SSD

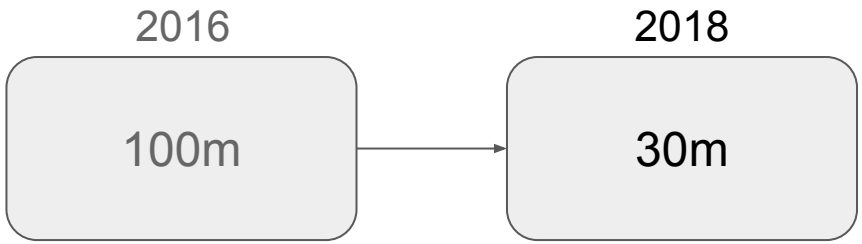


Конкретно тушим пожар #2

Ревизия выданных requests.cpu

- 100m requests.cpu на контейнер в 2016

Это оверхед, как 5% резерва на рутовом диске при 4Тб SSD



Конкретно тушим пожар #2

Ревизия выданных requests.cpu

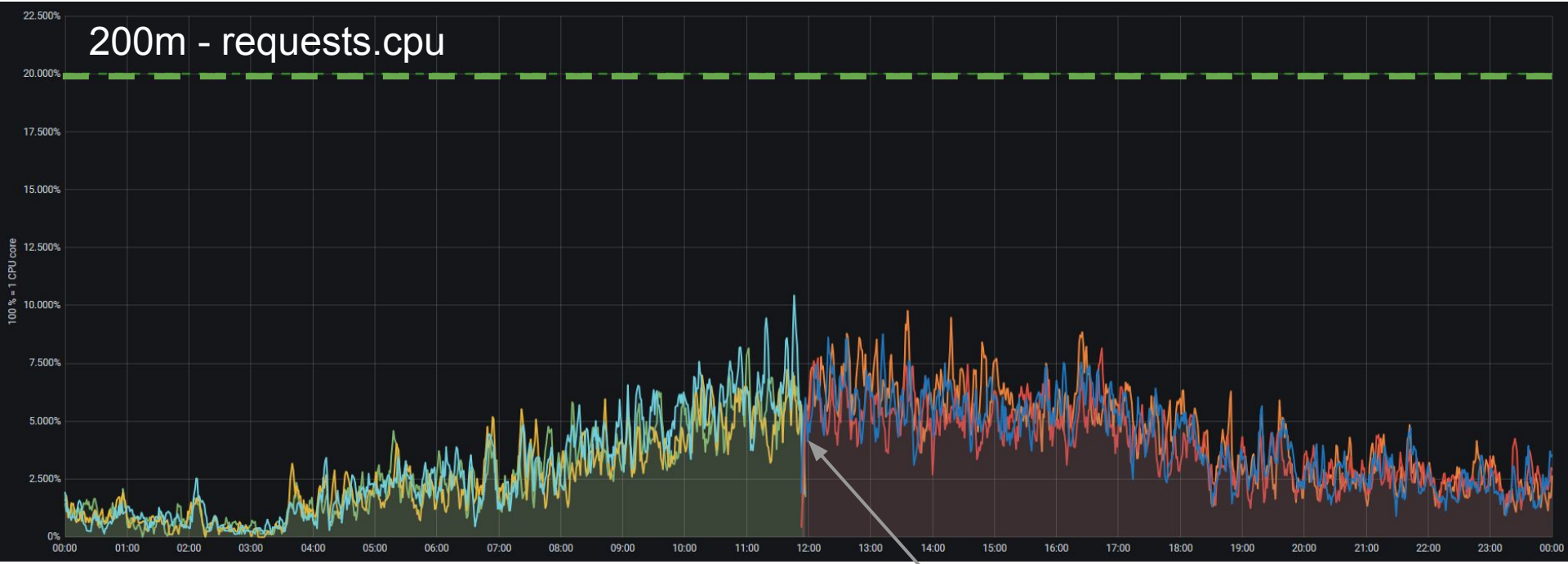
- 100m requests.cpu на контейнер в 2016

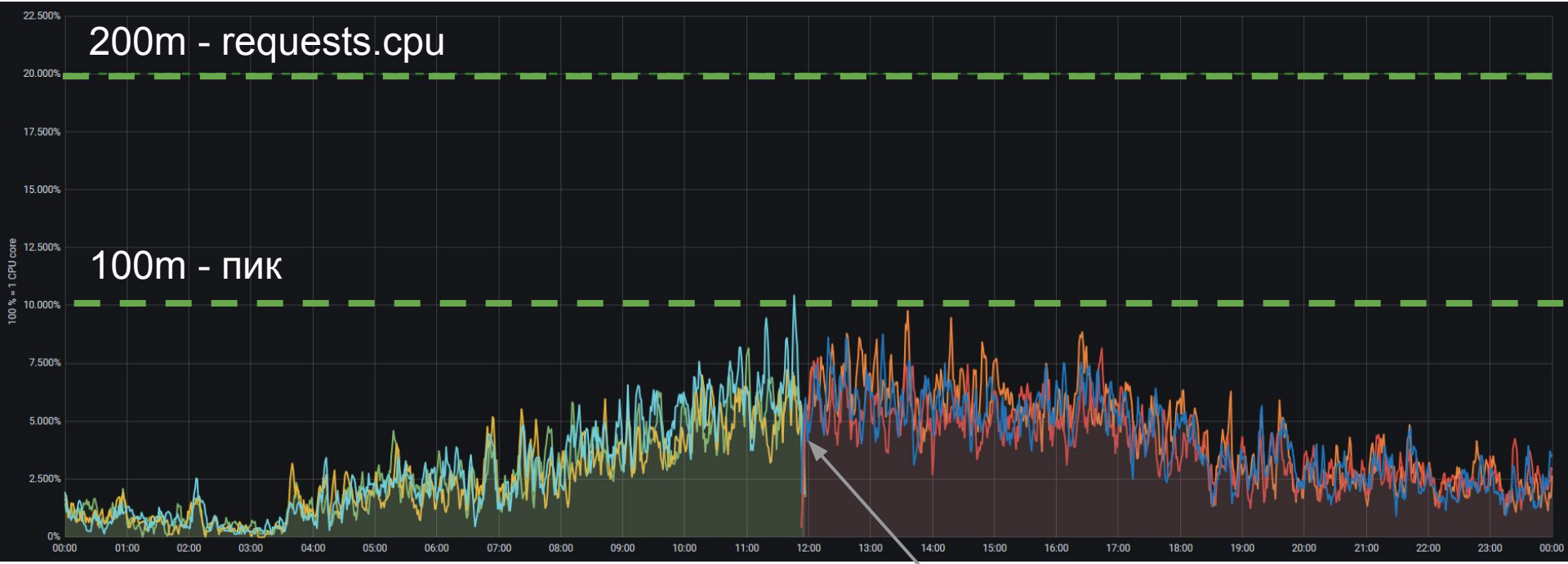
Это оверхед, как 5% резерва на рутовом диске при 4Тб SSD

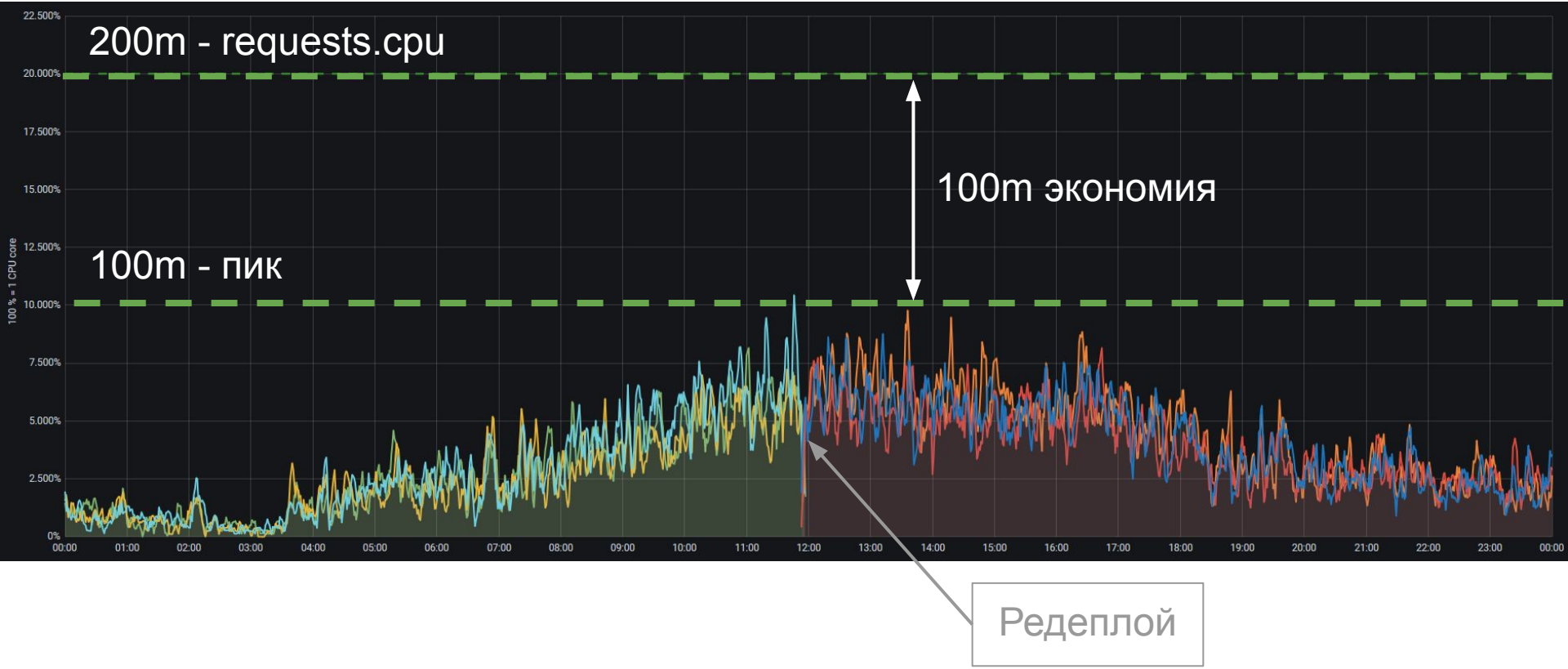
Конкретно тушим пожар #3

Запрошенное ≠ потребляемое

“*Экономим*” на оптимизации реквестов к приложениям.


200m - requests.cpu
22.500%
20.000%
17.500%
15.000%
12.500%
10.000%
7.500%
5.000%
2.500%
0%
100 % = 1 CPU core
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00
Редеплой


200m - requests.cpu
100m - пик
100 % = 1 CPU core
22.500%
20.000%
17.500%
15.000%
12.500%
10.000%
7.500%
5.000%
2.500%
0%
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00
Редеплой


200m - requests.cpu
100m экономия
100m - пик
Редеплой
22.500%
20.000%
17.500%
15.000%
12.500%
10.000%
7.500%
5.000%
2.500%
0%
100 % = 1 CPU core
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00

Вертикальный автоскейлинг приложений в k8s

Ручное изменение ресурсов у приложений - скучный процесс

Хотели изобрести свой велосипед, но нашли ванильный автоскейлер
https://github.com/kubernetes/autoscaler/tree/master/vertical-pod-autoscaler

Всё равно форкнули и дописали нужную нам специфику

Вертикальный автоскейлинг - форк

requests.cpu	limits.cpu	
100m	1000m	Заданные ресурсы
		Ванильный VPA
		Форк VPA

Вертикальный автоскейлинг - форк

requests.cpu	limits.cpu	
100m	1000m	Заданные ресурсы
10m	100m	Ванильный VPA
		Форк VPA

Вертикальный автоскейлинг - форк

requests.cpu	limits.cpu	
100m	1000m	Заданные ресурсы
10m	100m	Ванильный VPA
10m	1000m	Форк VPA

Горизонтальный автоскейлинг

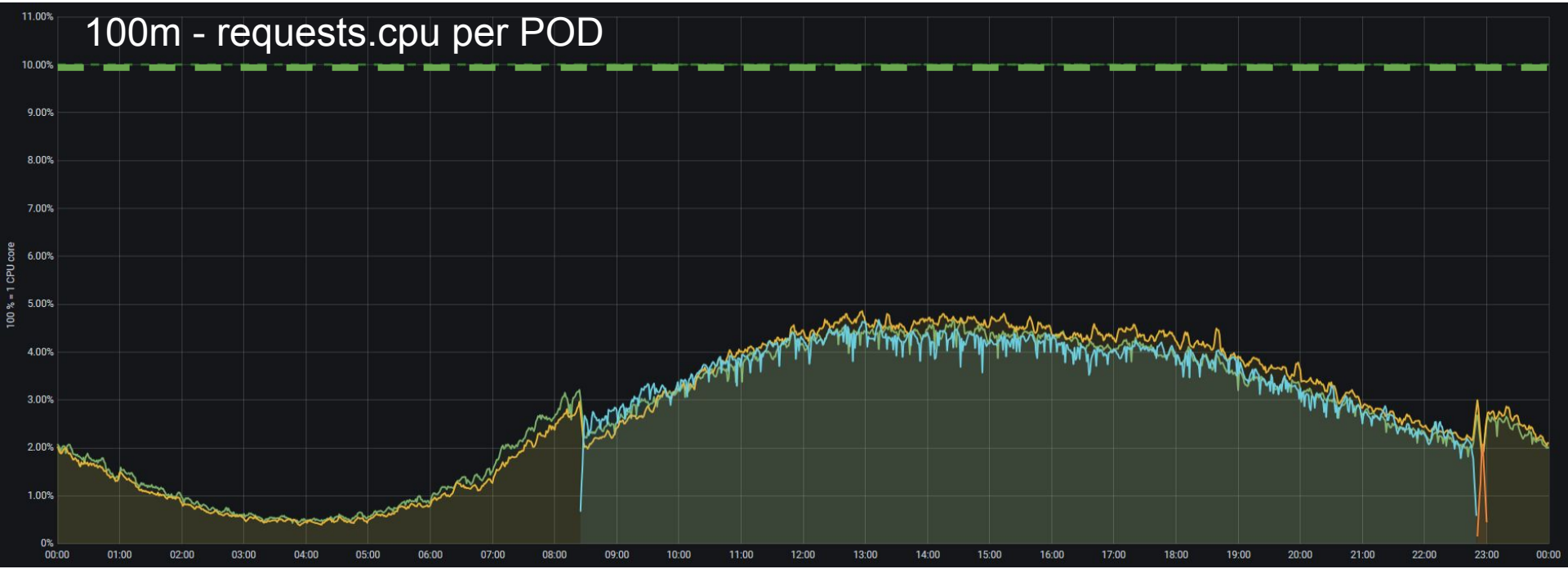
Позволяет освобождать ресурсы ночью

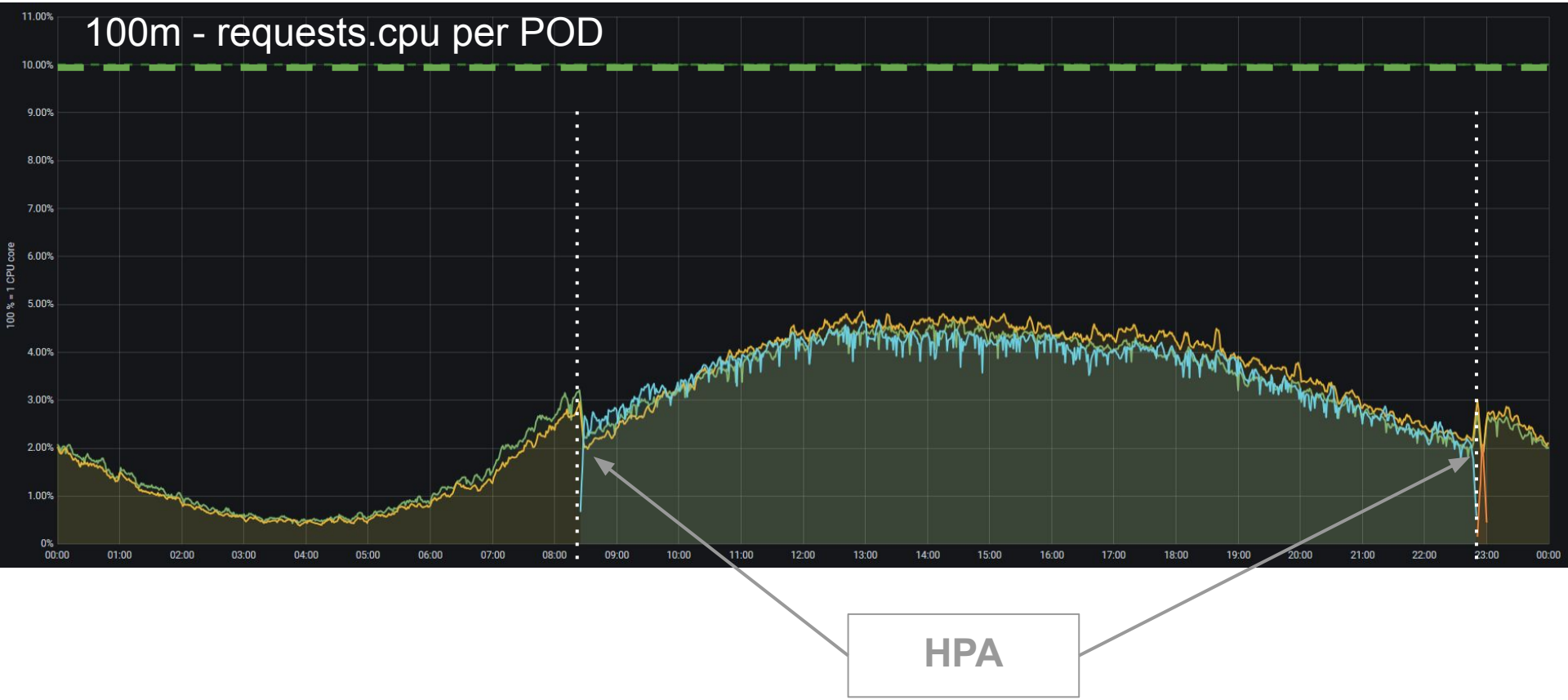
Днём не позволяет проседать SLA наших продуктов

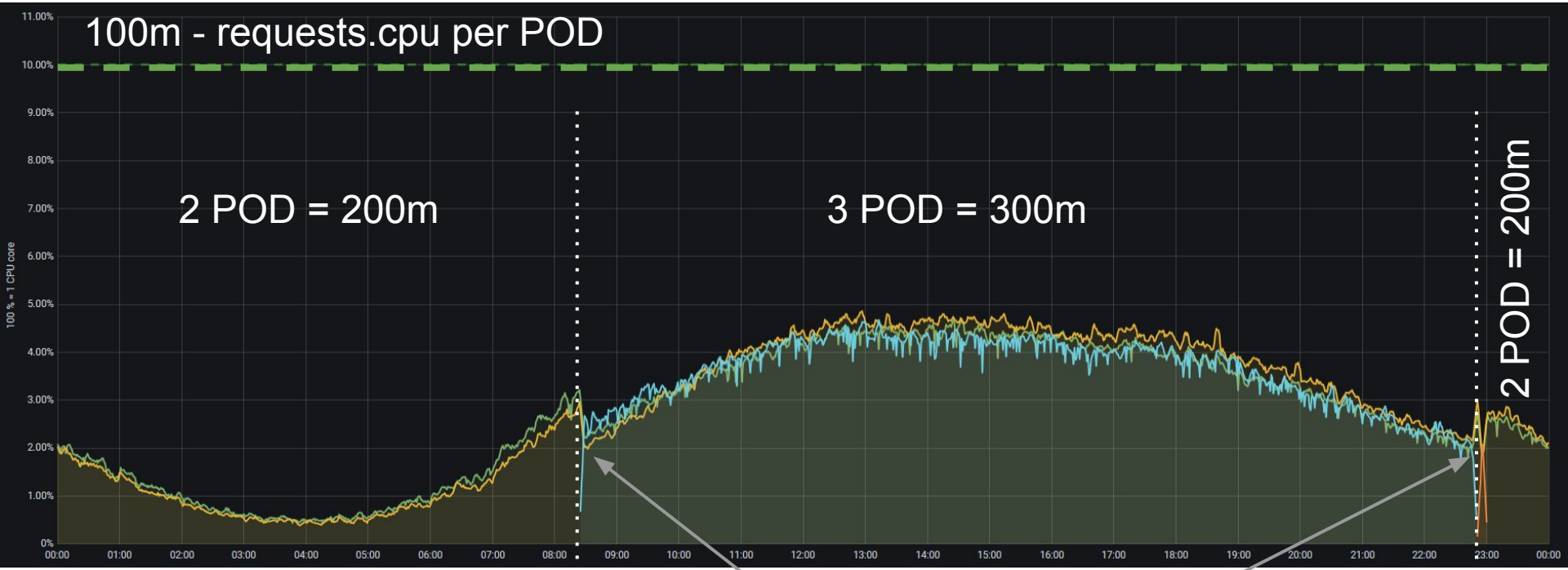
Используется только на больших продуктах

Не может использоваться вместе с вертикальным автоскейлингом*

* *но если очень хочется, то можно*


100m - requests.cpu per POD
11.00%
10.00%
9.00%
8.00%
7.00%
6.00%
5.00%
4.00%
3.00%
2.00%
1.00%
0%
100 % = 1 CPU core
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00


100m - requests.cpu per POD
11.00%
10.00%
9.00%
8.00%
7.00%
6.00%
5.00%
4.00%
3.00%
2.00%
1.00%
0%
100 % = 1 CPU core
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00
HPA


100m - requests.cpu per POD
2 POD = 200m
3 POD = 300m
2 POD = 200m
100 % = 1 CPU core
11.00%
10.00%
9.00%
8.00%
7.00%
6.00%
5.00%
4.00%
3.00%
2.00%
1.00%
0%
00:00
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
00:00

HPA

Пожар потушен
Пора планировать!

Capacity planning. Что дальше?

Что за зверь?

У кого он есть ?

Как обычно делается?

Самый действенный способ Capacity planning

Наша ситуация

Квоты выдаются на основе текущего использования ресурсов в кластере

Наша ситуация

Квоты выдаются на основе текущего использования ресурсов в кластере

Нет информации, сколько реально выдано

Наша ситуация

Квоты выдаются на основе текущего использования ресурсов в кластере

Нет информации, сколько реально выдано

Изоляция “жирных” сервисов не учитывается в квотах

Наша ситуация

Квоты выдаются на основе текущего использования ресурсов в кластере

Нет информации, сколько реально выдано

Изоляция “жирных” сервисов не учитывается в квотах

Непонятно когда заказывать железо и сколько

Наша ситуация

Квоты выдаются на основе текущего использования ресурсов в кластере

Нет информации, сколько реально выдано

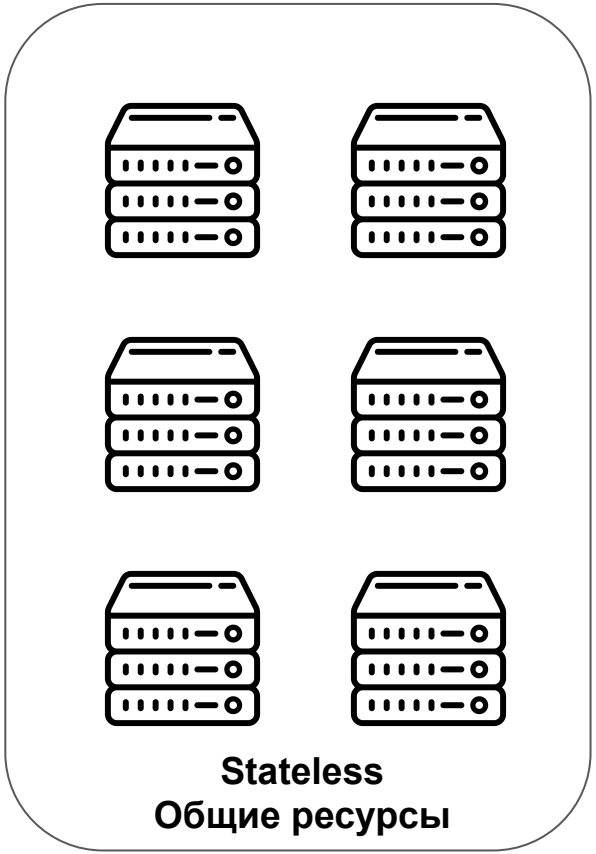
Изоляция “жирных” сервисов не учитывается в квотах

Непонятно когда заказывать железо и сколько

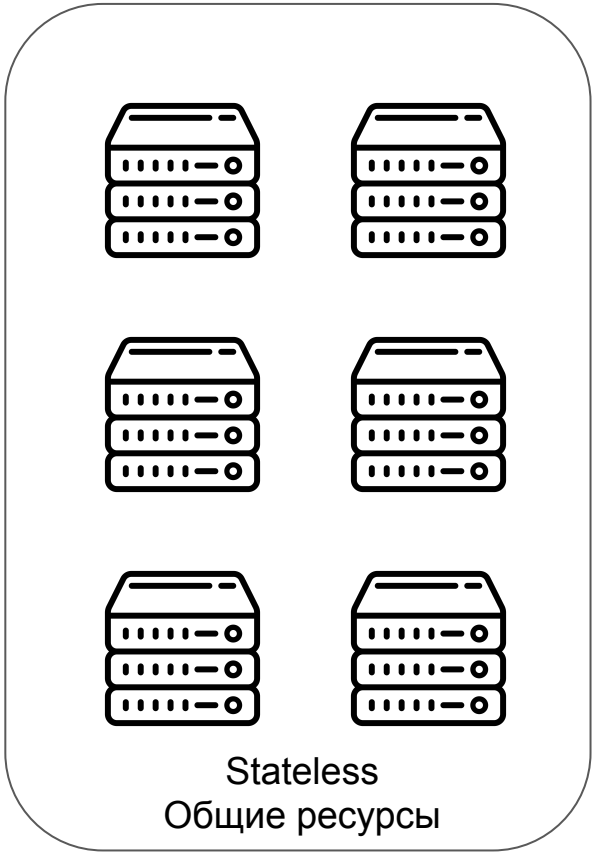
Новый сервис -> новые ресурсы -> ожидание

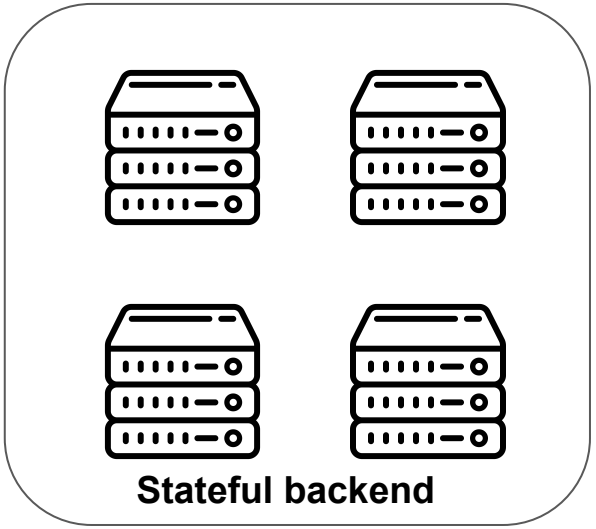
Проблема планирования

K8s как обычные VM

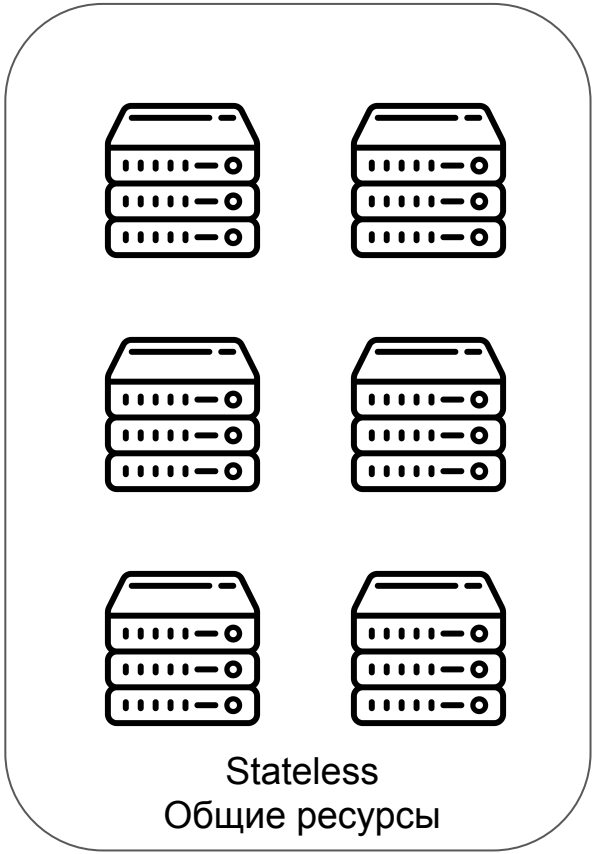


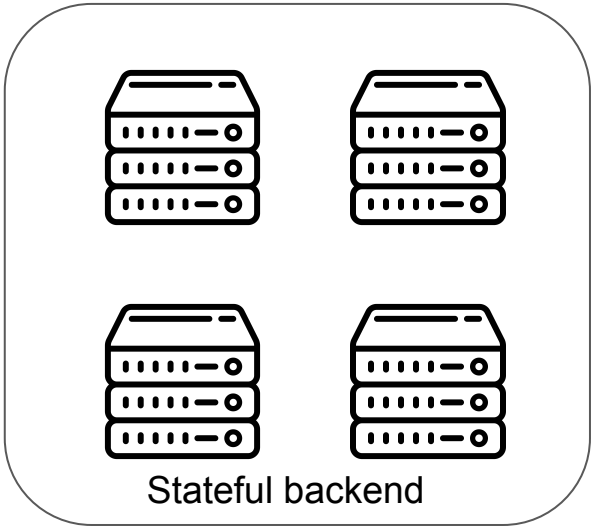
K8s как обычные VM

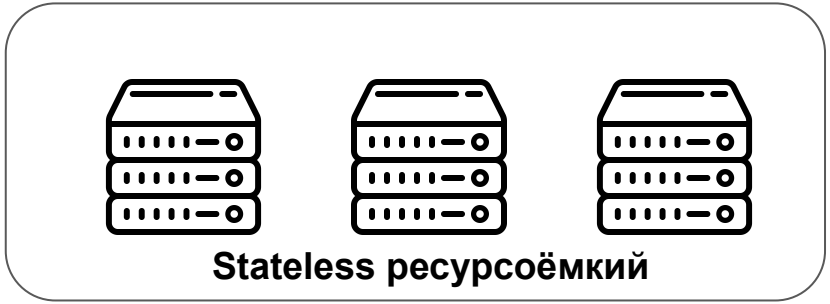




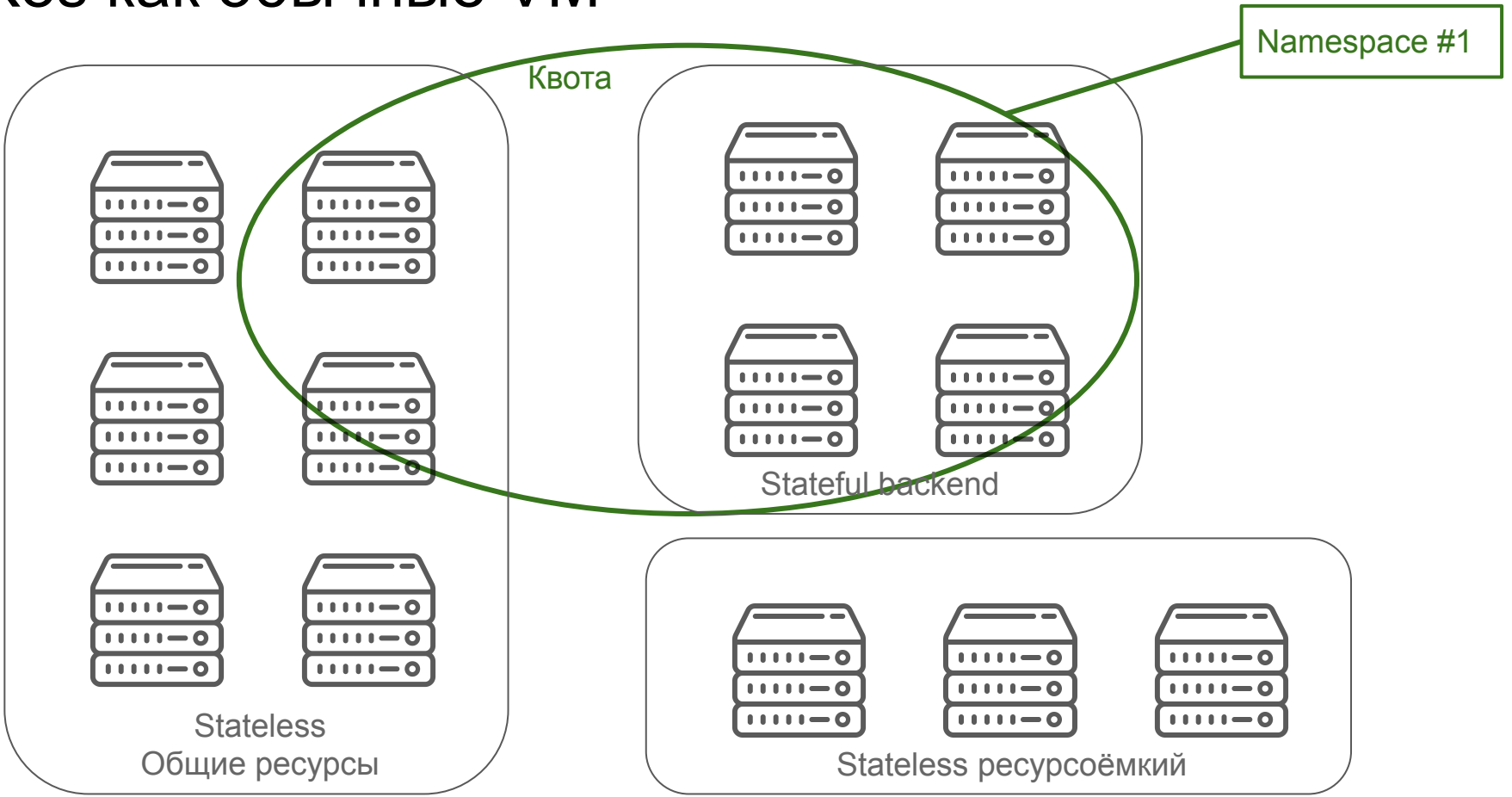
K8s как обычные VM



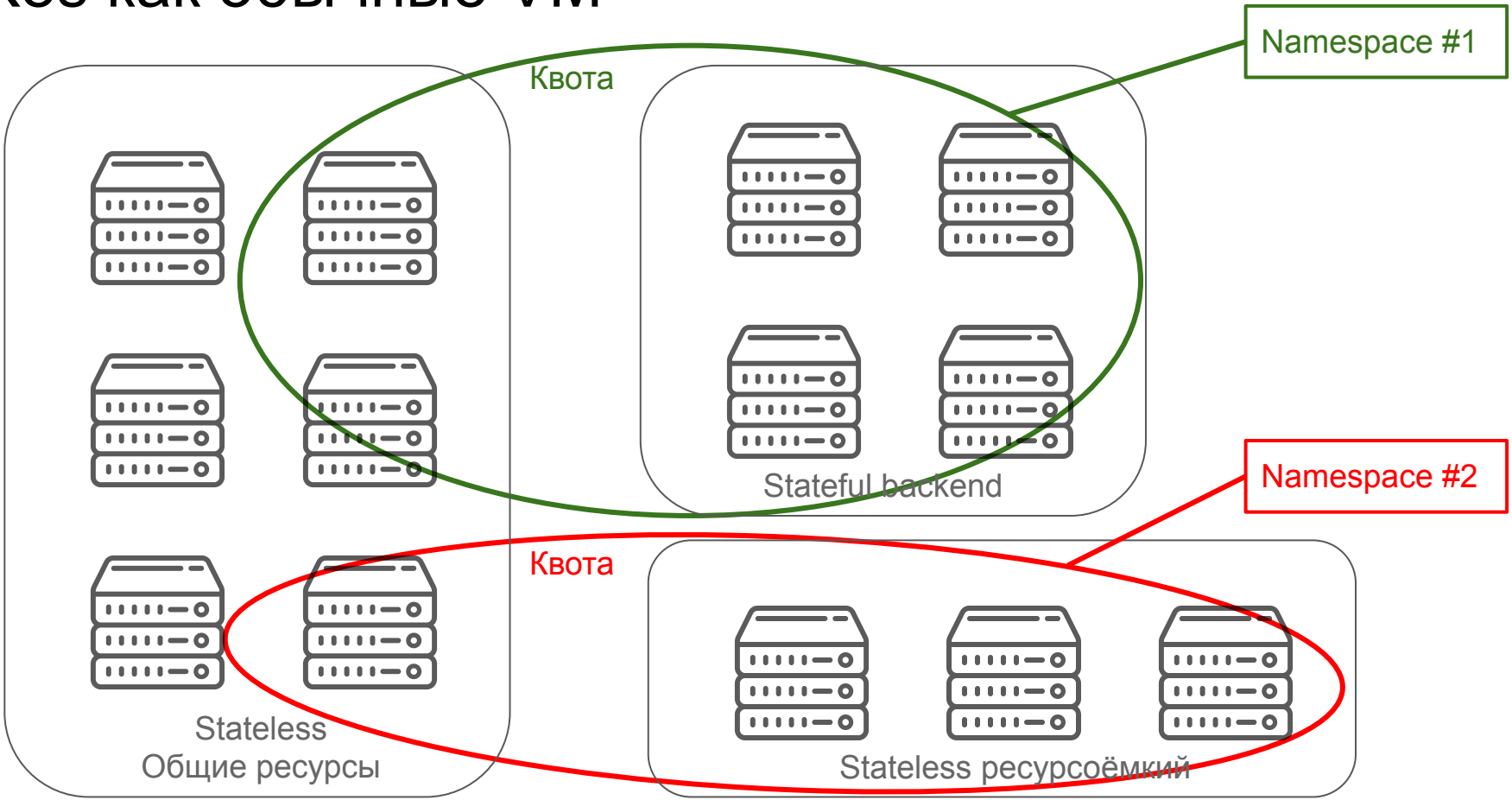




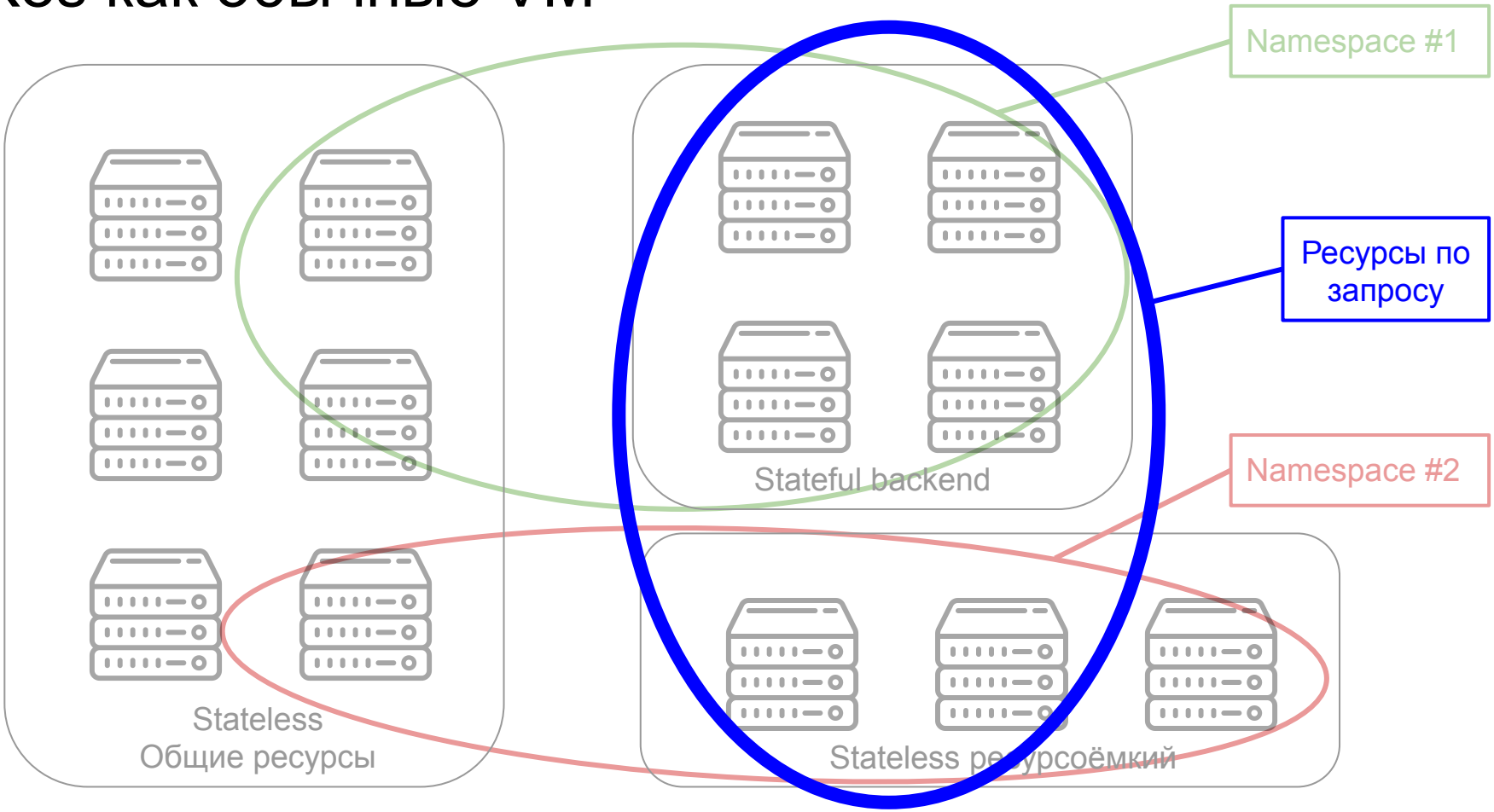
K8s как обычные VM



K8s как обычные VM



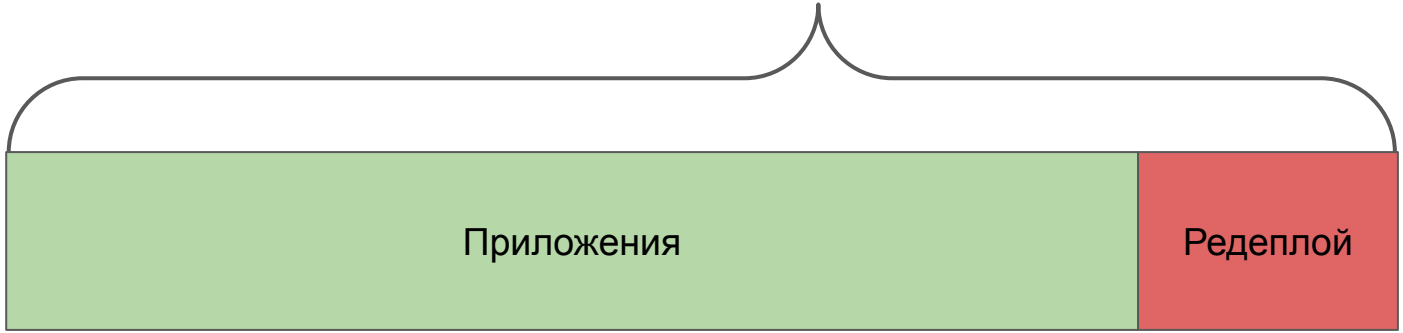
K8s как обычные VM



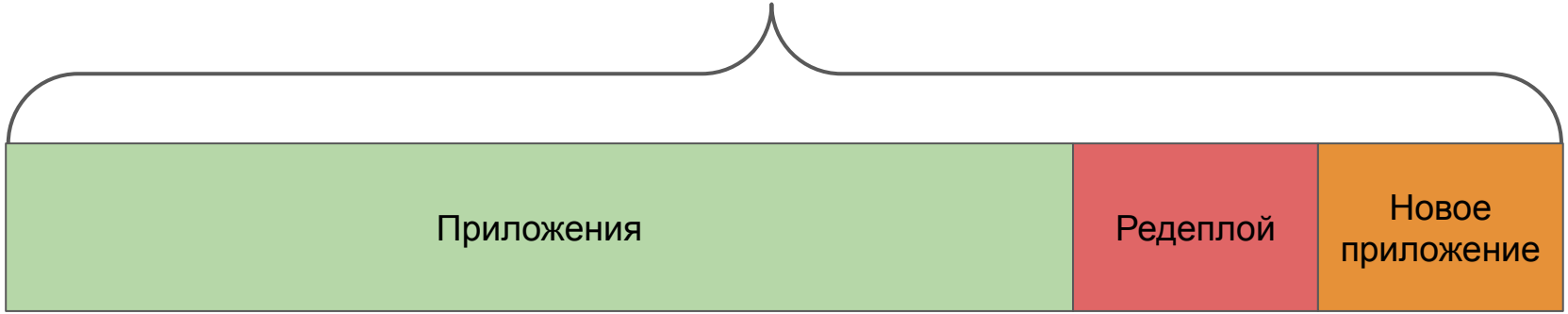
K8s как обычные VM

Проблема ждуна

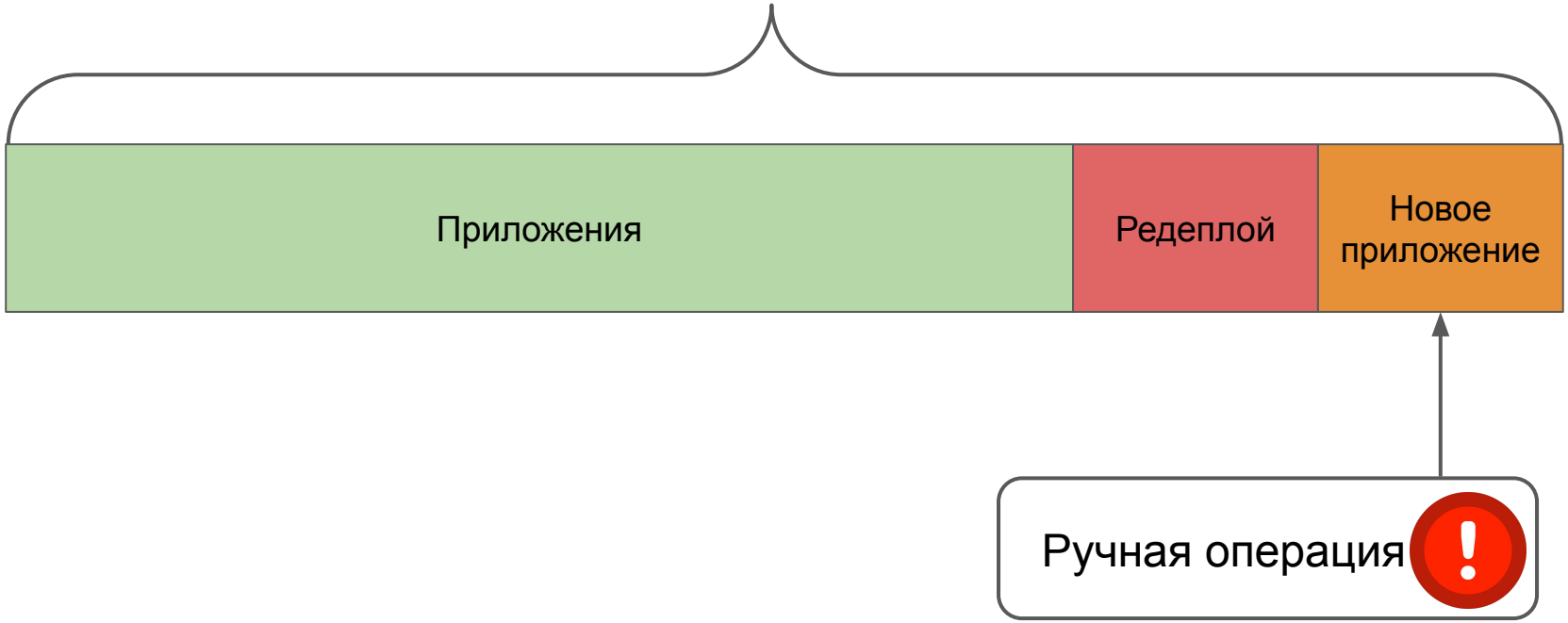
Namespace quota



Namespace quota



Namespace quota

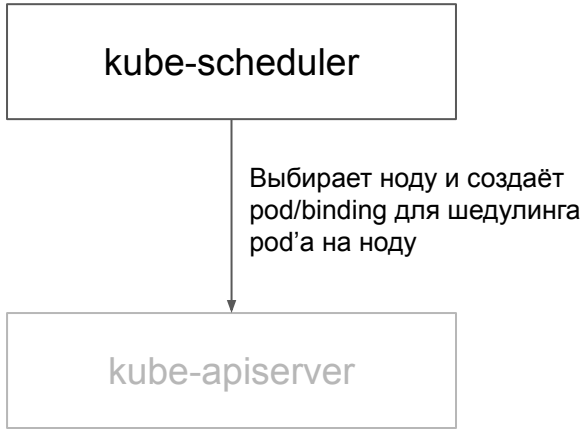


Решаем проблемы!

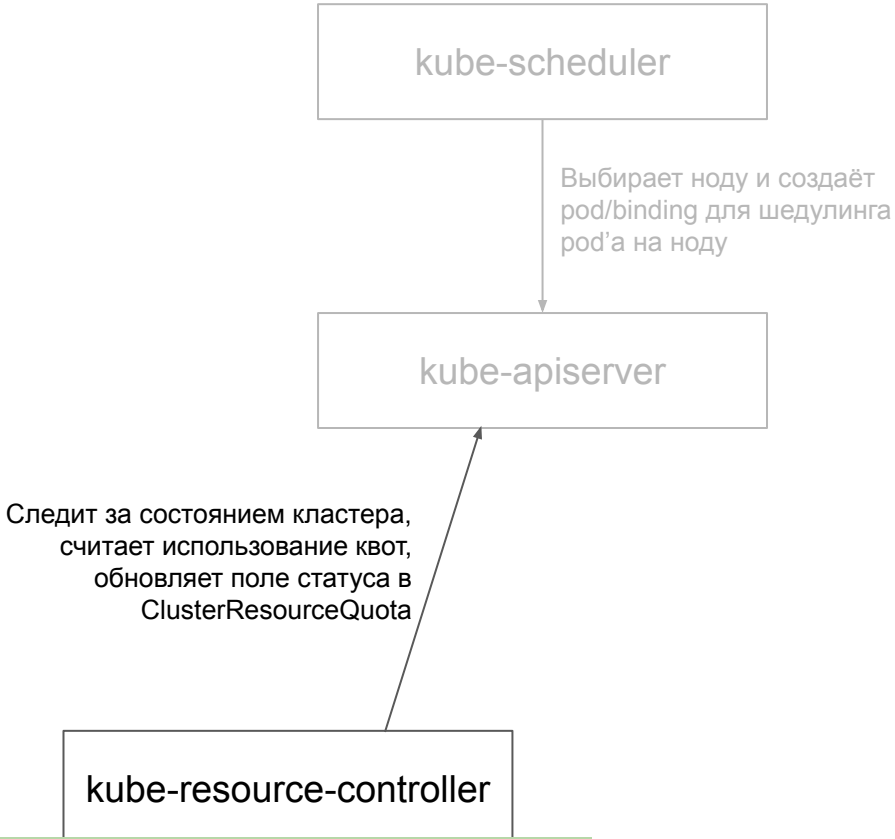
Свой контроллер квот k8s (идея)

Свой контроллер квот k8s (идея)

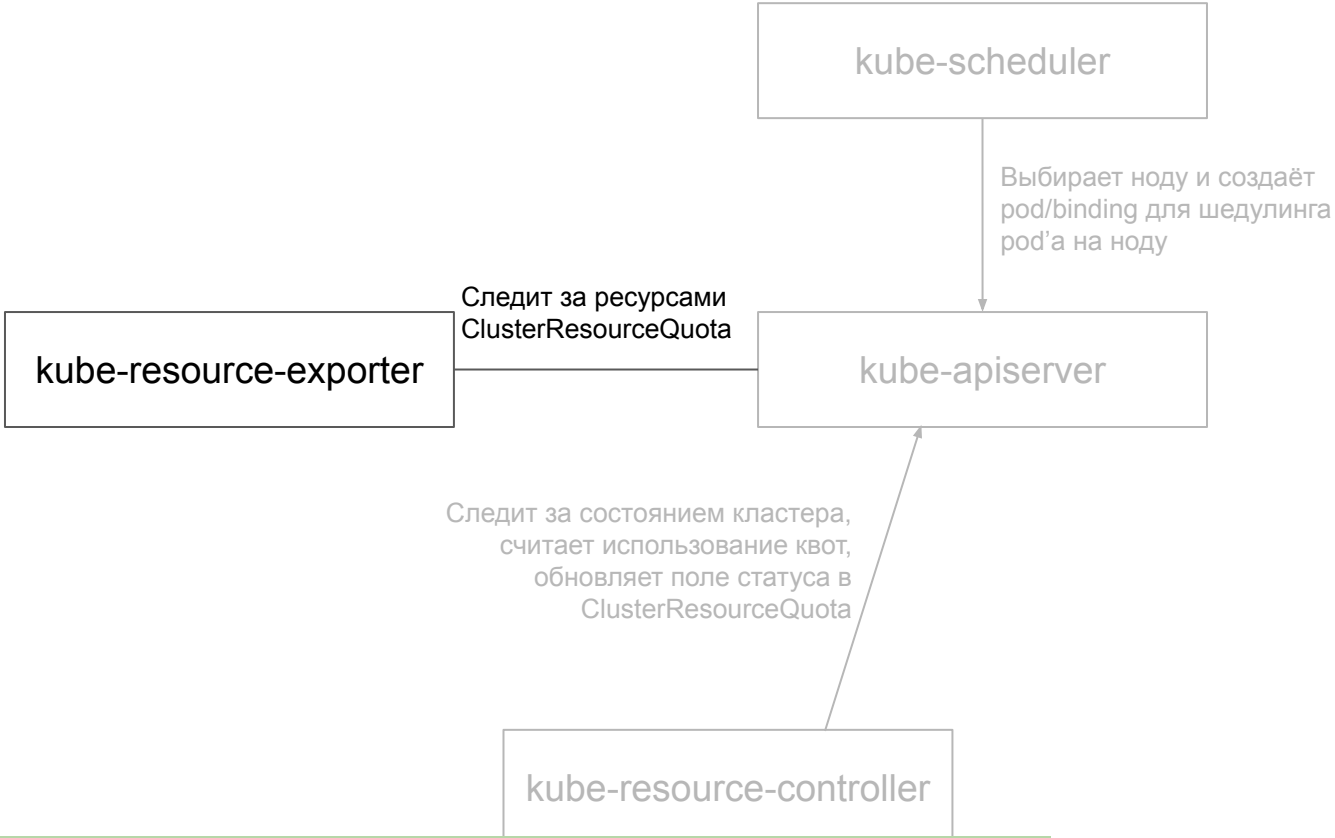
Свой контроллер квот k8s (идея)



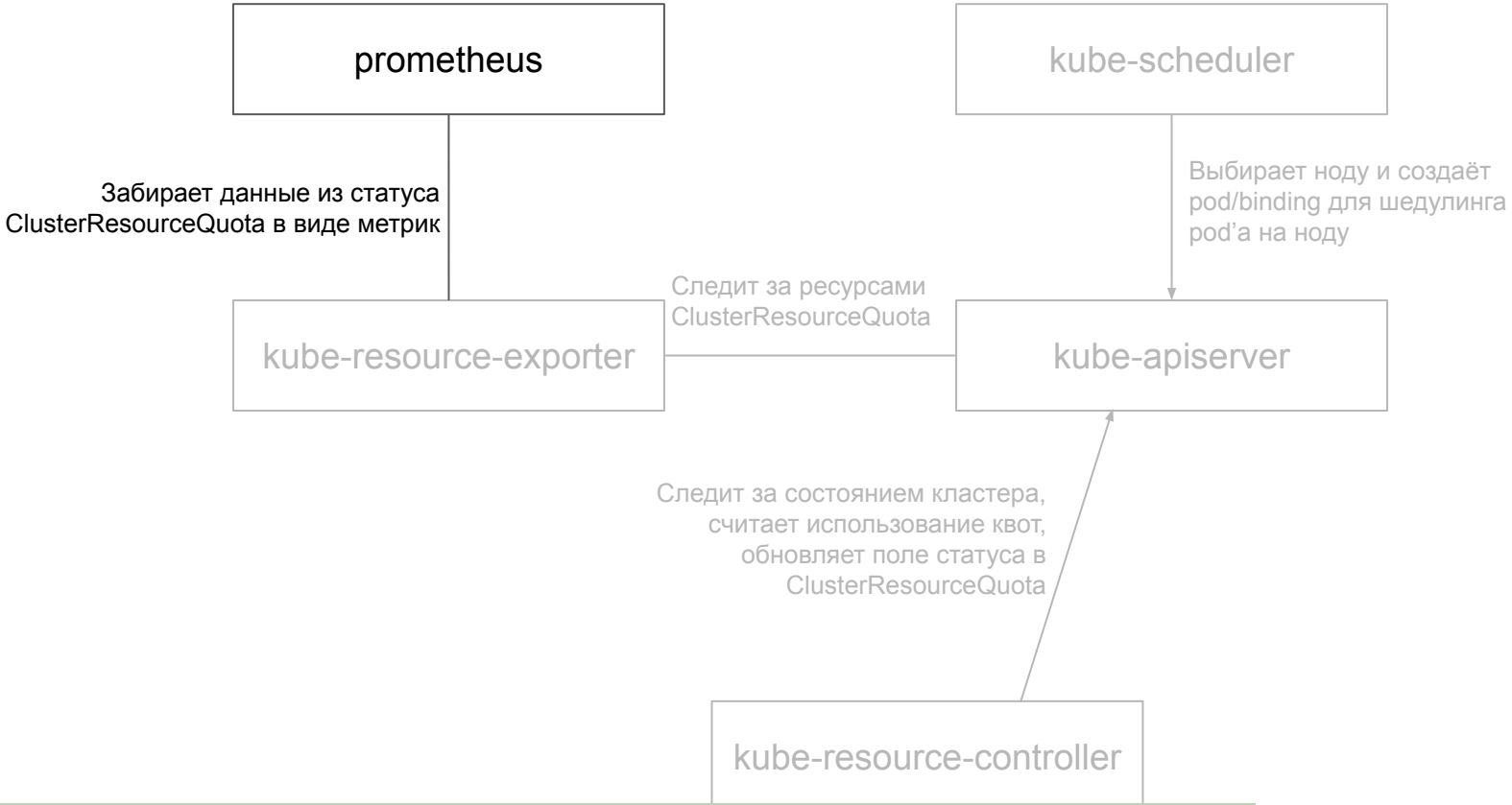
Свой контроллер квот k8s (идея)



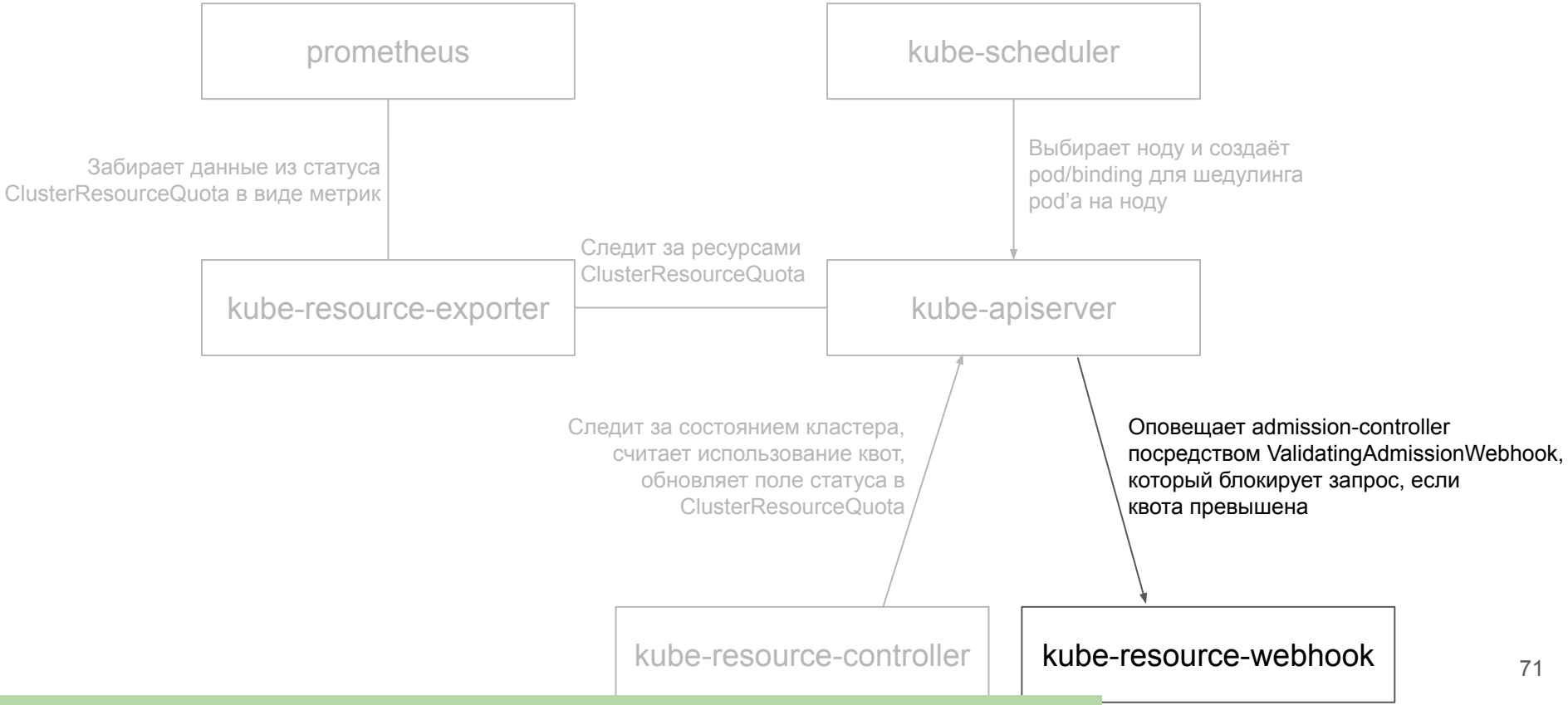
Свой контроллер квот k8s (идея)



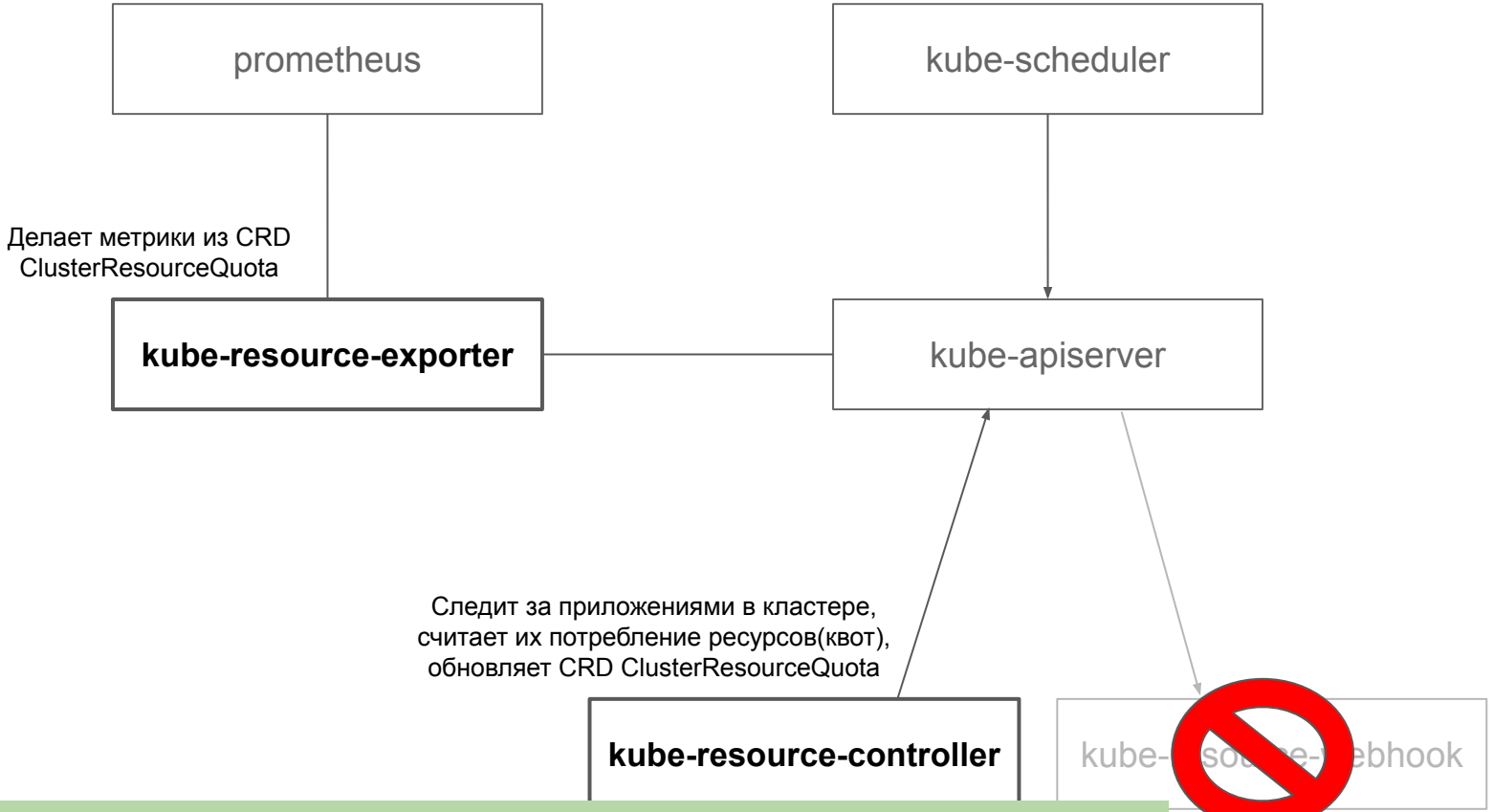
Свой контроллер квот k8s (идея)



Свой контроллер квот k8s (идея)



Свой контроллер квот k8s (факт)



Контроллер решает проблемы

Квоты привязаны к конкретным nodeSelector

Теперь легко понять, сколько планировать новых ресурсов!

Контроллер решает проблемы

Квоты привязаны к конкретным nodeSelector

Теперь легко понять, сколько планировать новых ресурсов!

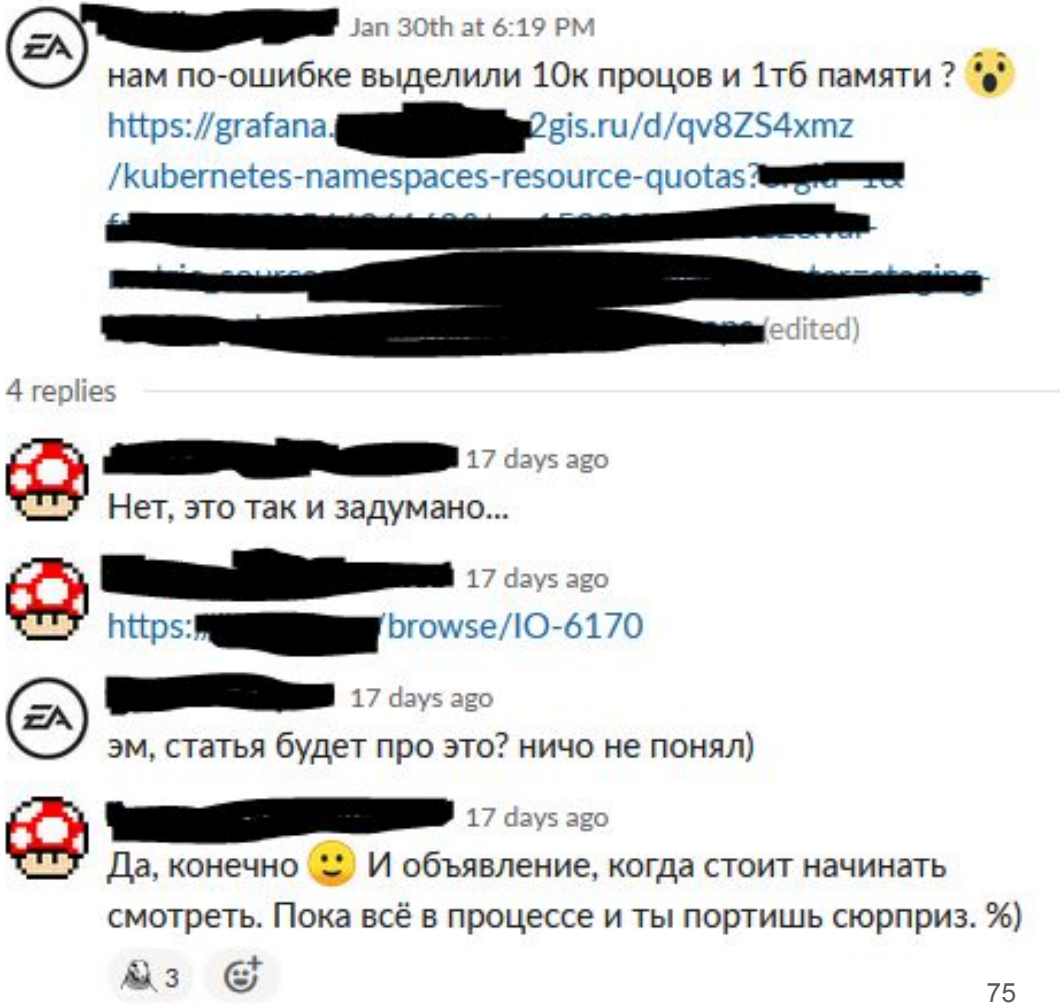
Ждун свободен! 

Теперь разработку никто не ограничивает

Если превышаются квоты - приходит алерт

Задачи на новые квоты решаем пачками

Перед релизом пользователи что-то заподозрили



Как выглядит наша схема с квотами

Обычная квота

```
apiVersion: v1
kind: ResourceQuota
metadata:
 name: all-quota-counts
 namespace: casino
spec:
 hard:
   limits.cpu: 10k
   limits.memory: 1Ti
   persistentvolumeclaims: "1"
   pods: 1k
   requests.cpu: 10k
   requests.memory: 1Ti
   secrets: "60"
```

Как выглядит наша схема с квотами

Обычная квота

```
apiVersion: v1
kind: ResourceQuota
metadata:
 name: all-quota-counts
 namespace: casino
spec:
 hard:
   limits.cpu: 10k
   limits.memory: 1Ti
   persistentvolumeclaims: "1"
   pods: 1k
   requests.cpu: 10k
   requests.memory: 1Ti
   secrets: "60"
```

Наш ресурс

```
apiVersion: resourcemanager.2gis.io/v1alpha1
kind: ClusterResourceQuota
metadata:
  name: casino-worker
  labels:
    namespace: casino
    team: casino
spec:
  namespaceSelector:
    matchLabels:
      namespace: casino
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: role
        operator: In
        values:
        - worker
  soft:
      limits.cpu: 146
      limits.memory: 210Gi
      pods: 112
      requests.cpu: 53
```

Как выглядит наша схема с квотами

Обычная квота

```
apiVersion: v1
kind: ResourceQuota
metadata:
 name: all-quota-counts
 namespace: casino
spec:
 hard:
   limits.cpu: 10k
   limits.memory: 1Ti
   persistentvolumeclaims: "1"
   pods: 1k
   requests.cpu: 10k
   requests.memory: 1Ti
   secrets: "60"
```

Наш ресурс

```
apiVersion: resourcemanager.2gis.io/v1alpha1
kind: ClusterResourceQuota
metadata:
  name: casino-worker
  labels:
    namespace: casino
    team: casino
spec:
  namespaceSelector:
    matchLabels:
      namespace: casino
  nodeSelector:
    nodeSelectorTerms:
    - matchExpressions:
      - key: role
        operator: In
        values:
        - worker
  soft:
      limits.cpu: 146
      limits.memory: 210Gi
      pods: 112
      requests.cpu: 53
```

Как выглядит наша схема с квотами

Обычная квота

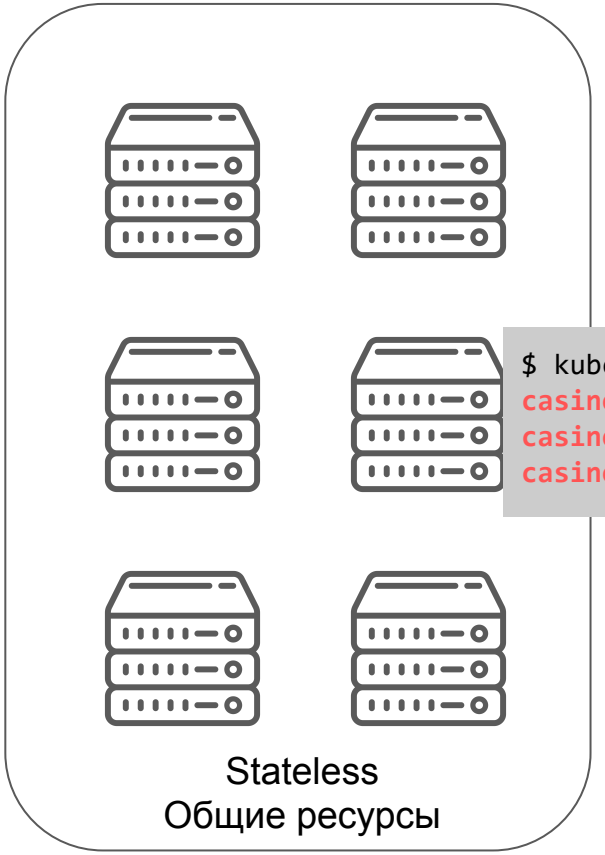
```
apiVersion: v1
kind: ResourceQuota
metadata:
 name: all-quota-counts
 namespace: casino
spec:
 hard:
   limits.cpu: 10k
   limits.memory: 1Ti
   persistentvolumeclaims: "1
   pods: 1k
   requests.cpu: 10k
   requests.memory: 1Ti
   secrets: "60"
```

```
$ kubectl get ClusterResourceQuota | grep casino
casino-other
casino-worker
casino-worker.casino.cassandra
```

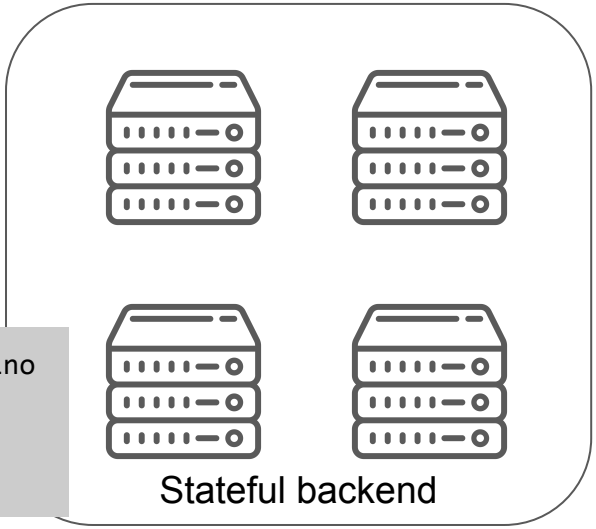
Наш ресурс

```
apiVersion: resourcemanager.2gis.io/v1alpha1
kind: ClusterResourceQuota
metadata:
  name: casino-worker
  labels:
    namespace: casino
    team: casino
                    nodeSelectorTerms:
    - matchExpressions:
      - key: role
        operator: In
        values:
        - worker
  soft:
      limits.cpu: 146
      limits.memory: 210Gi
      pods: 112
      requests.cpu: 53
```

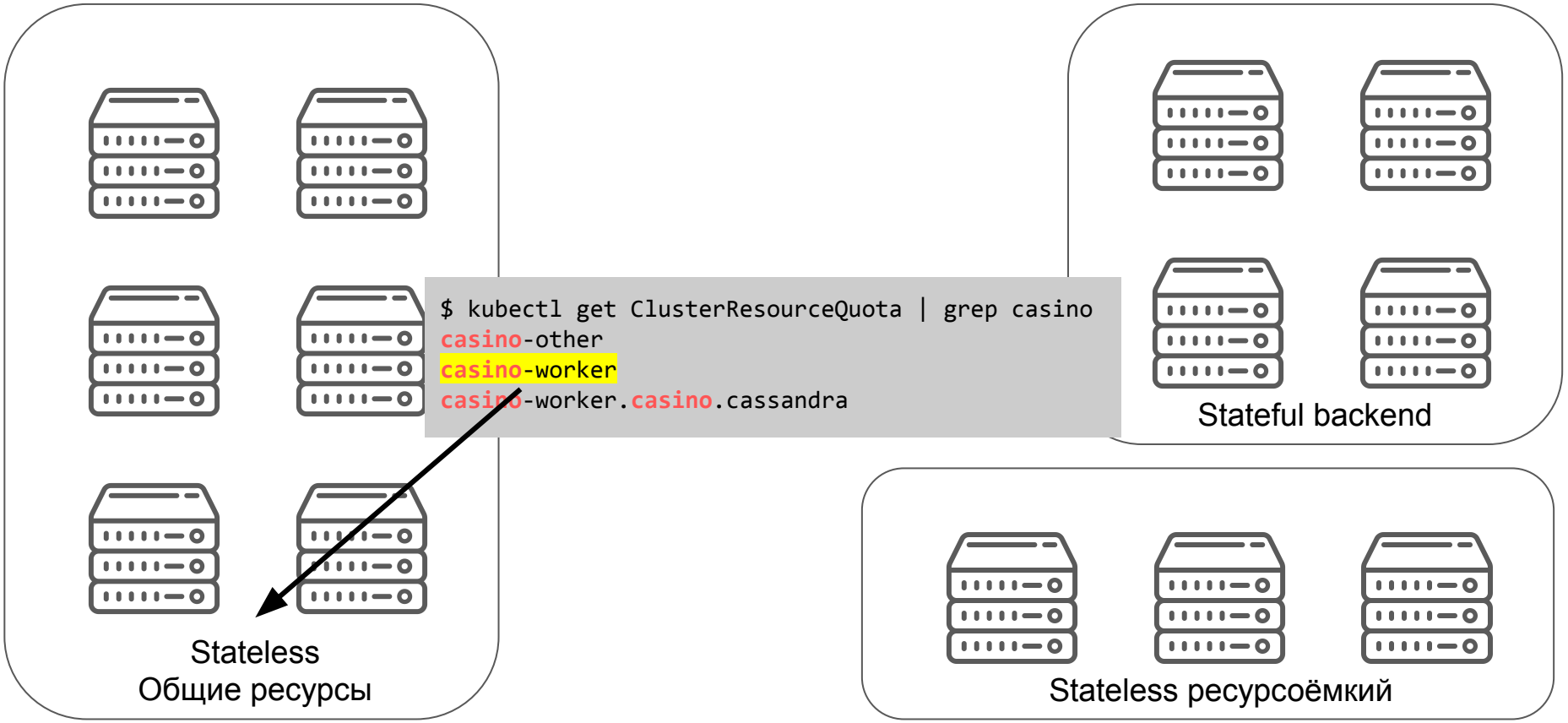
Как выглядит наша схема с квотами



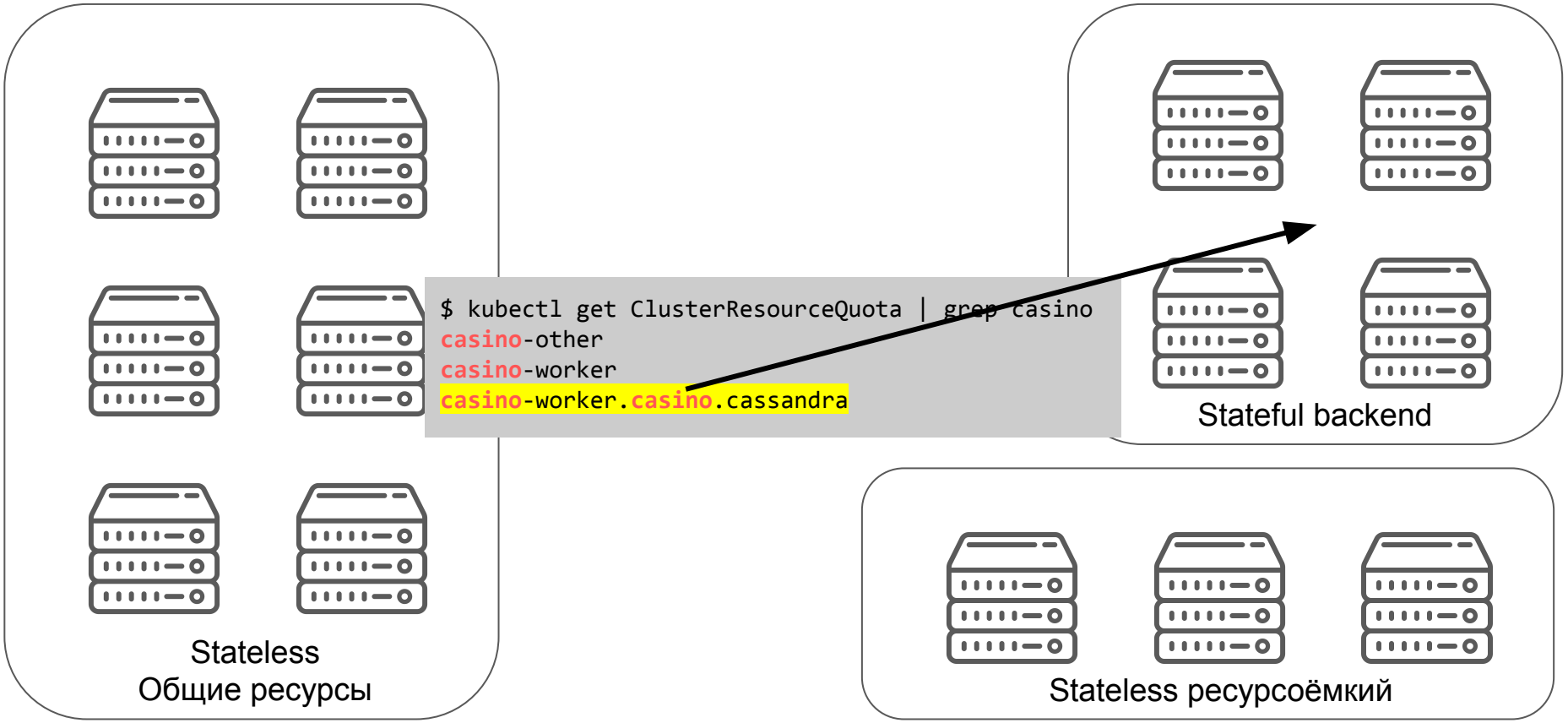

```
$ kubectl get ClusterResourceQuota | grep casino
casino-other
casino-worker
casino-worker.casino.cassandra
```



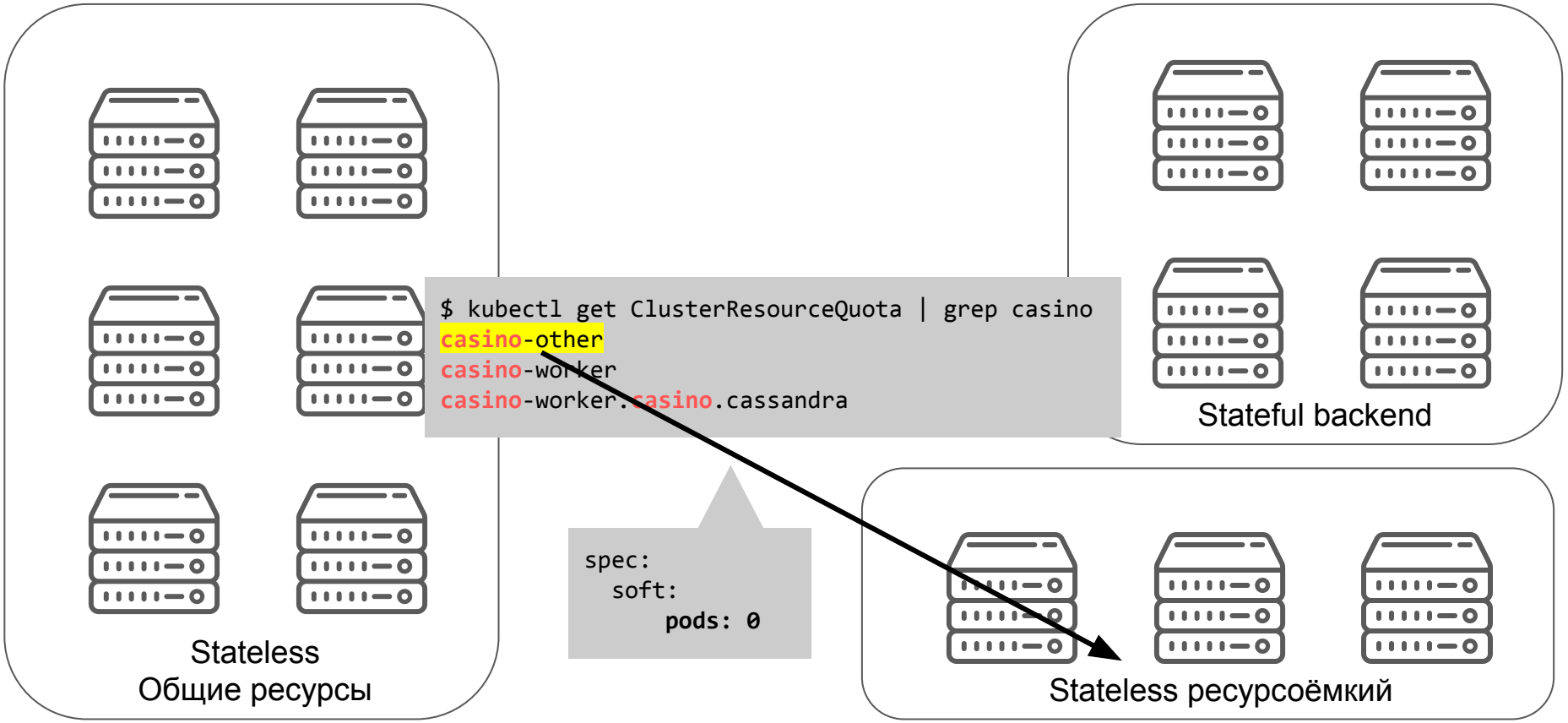
Как выглядит наша схема с квотами



Как выглядит наша схема с квотами

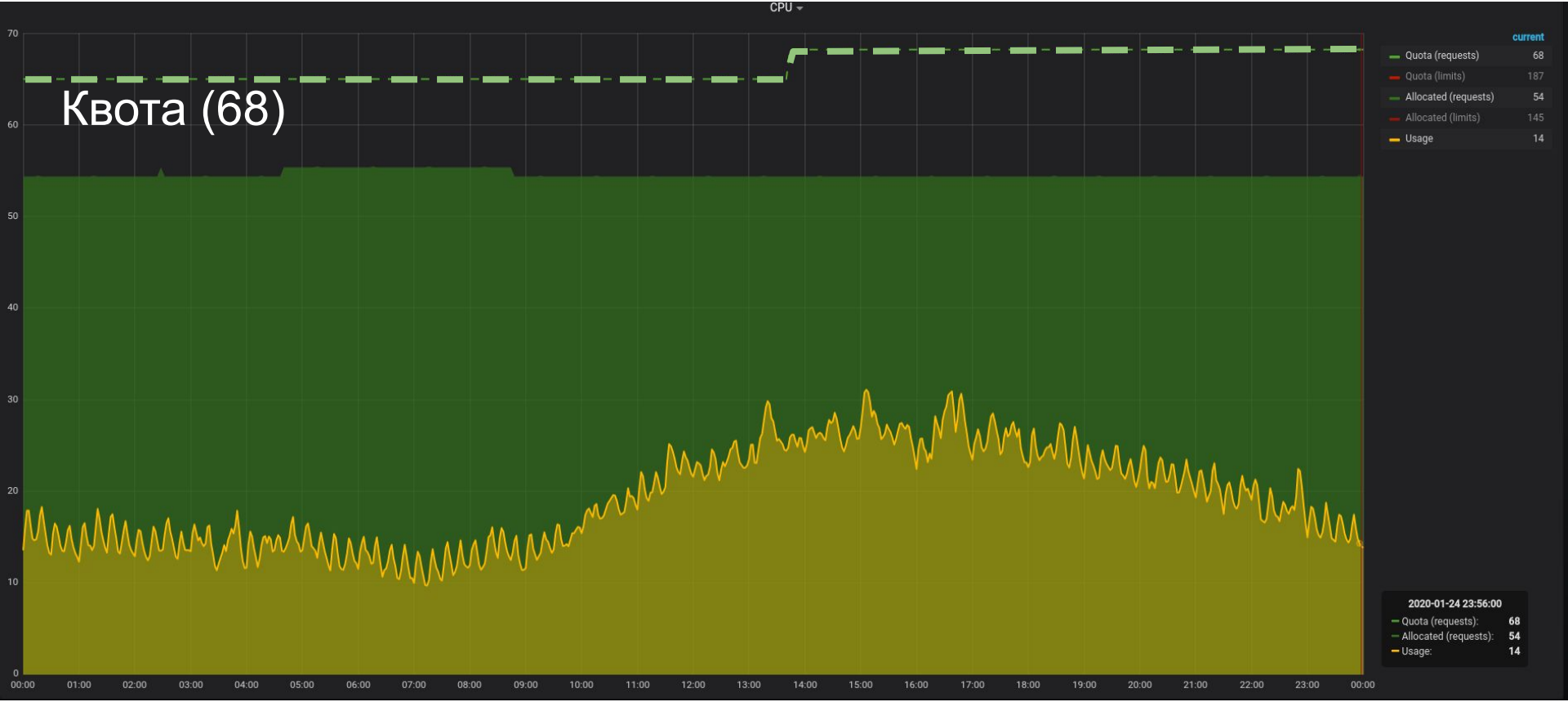


Как выглядит наша схема с квотами

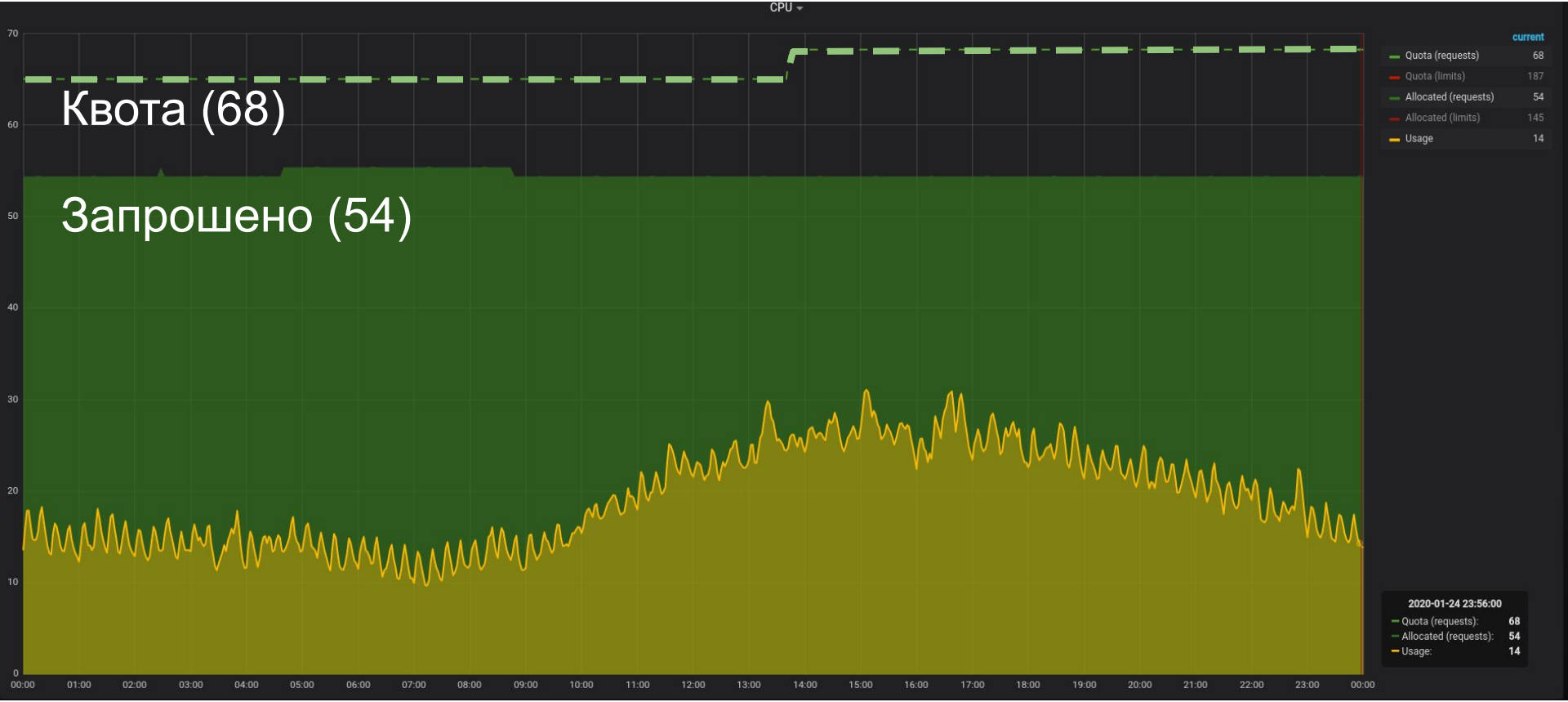


Графики

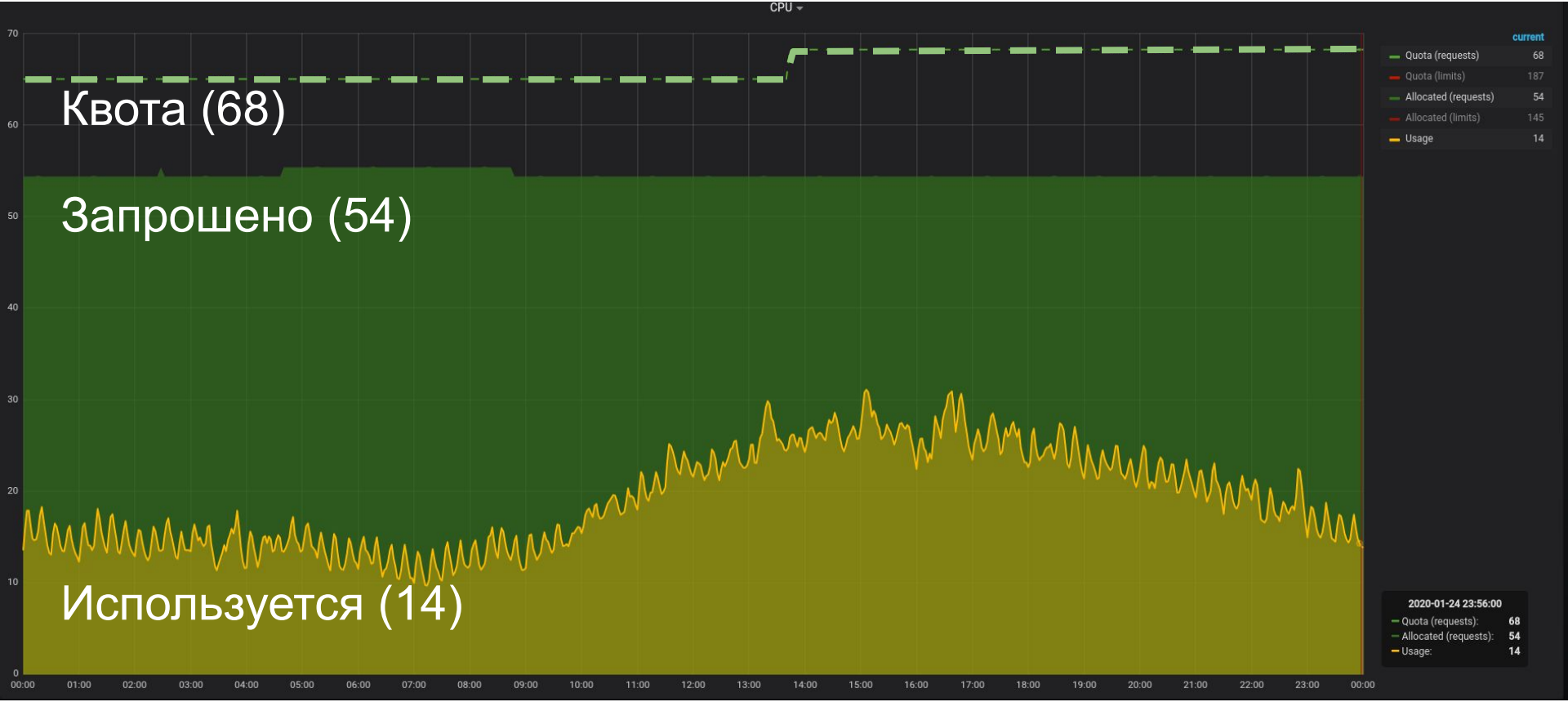
Как было - **считаем всё выданное**



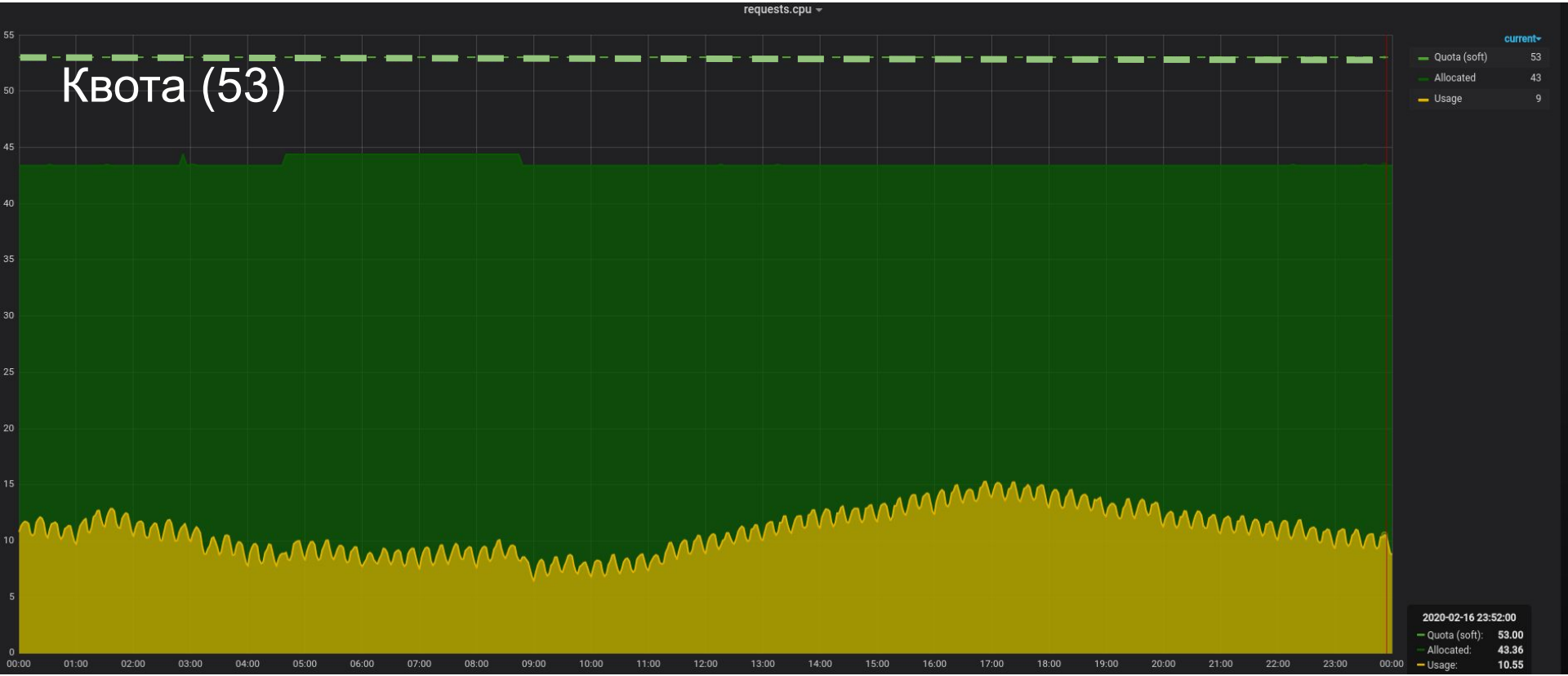
Как было - **считаем всё выданное**



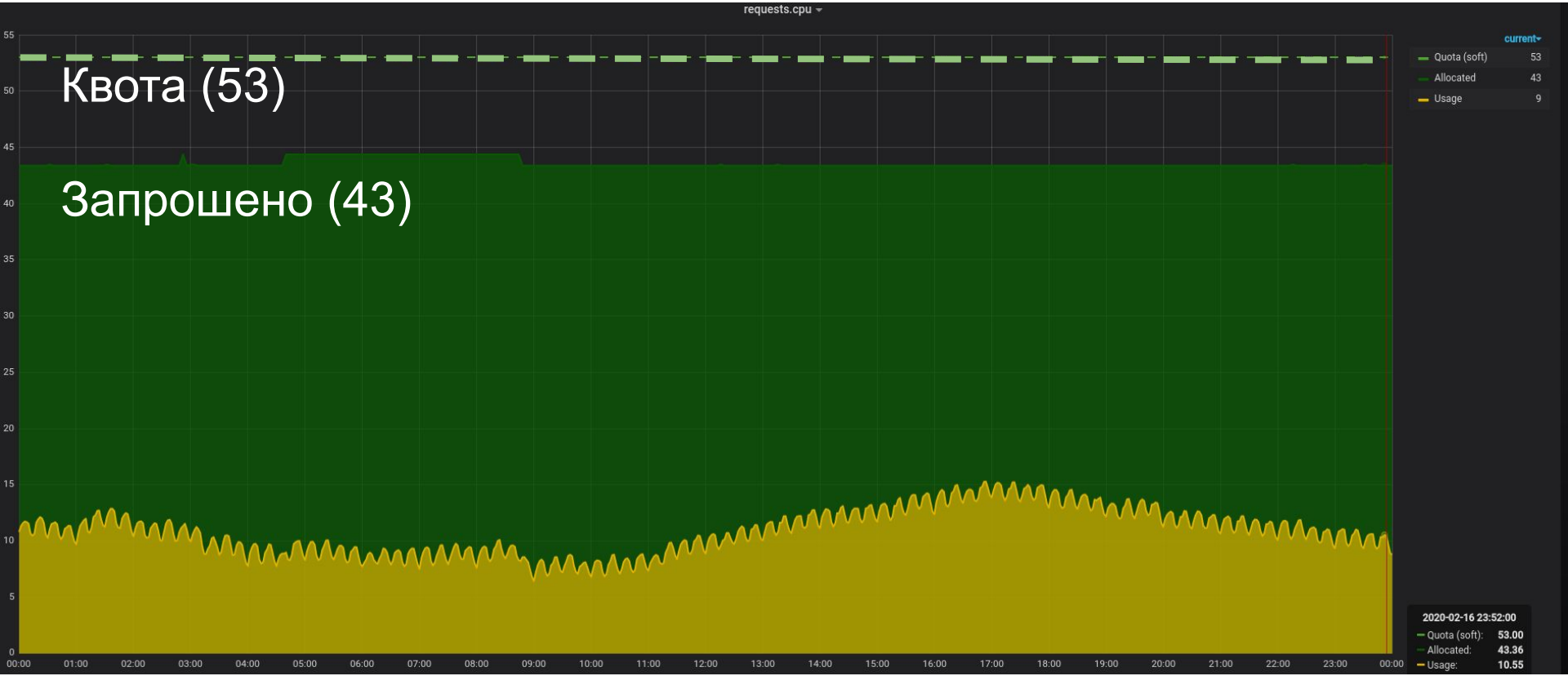
Как было - **считаем всё выданное**



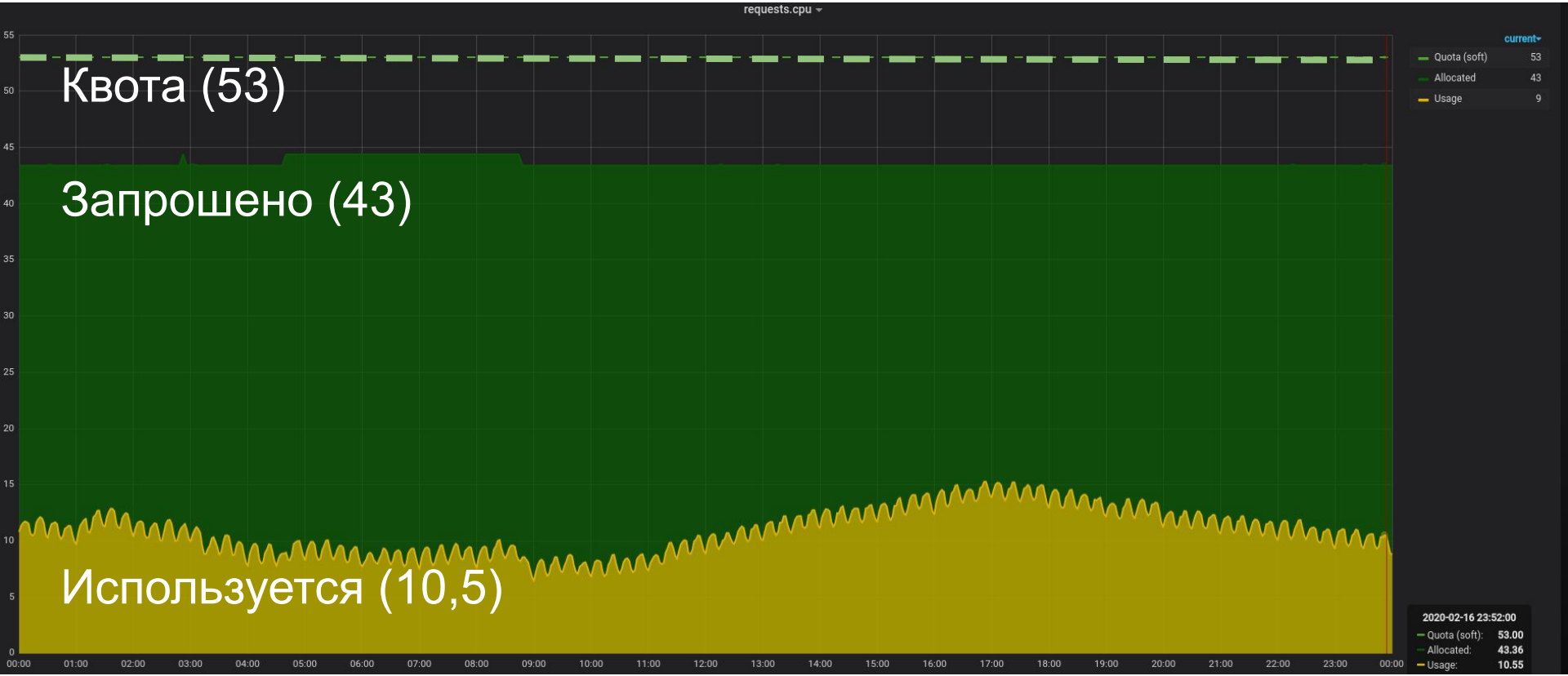
Как стало - **считаем только общие ресурсы**



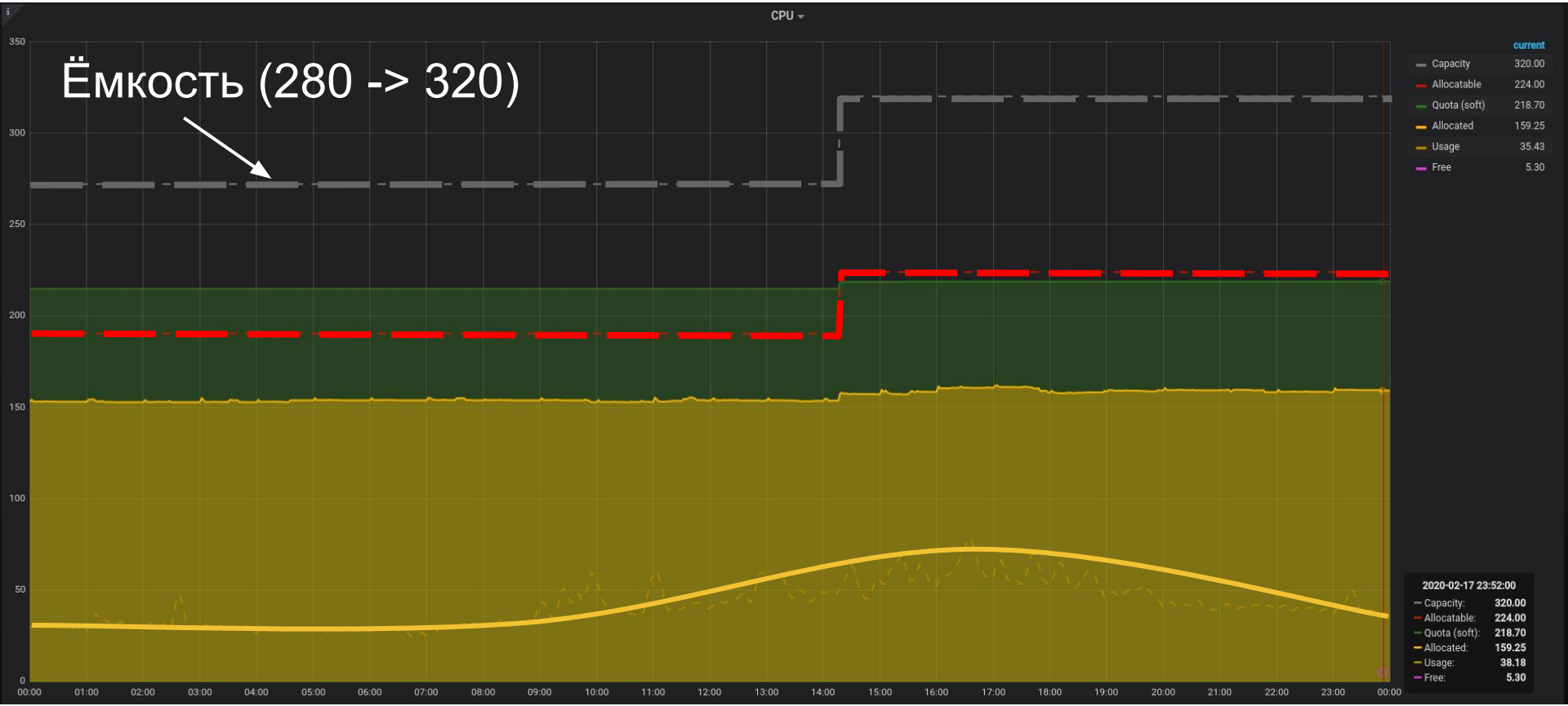
Как стало - **считаем только общие ресурсы**



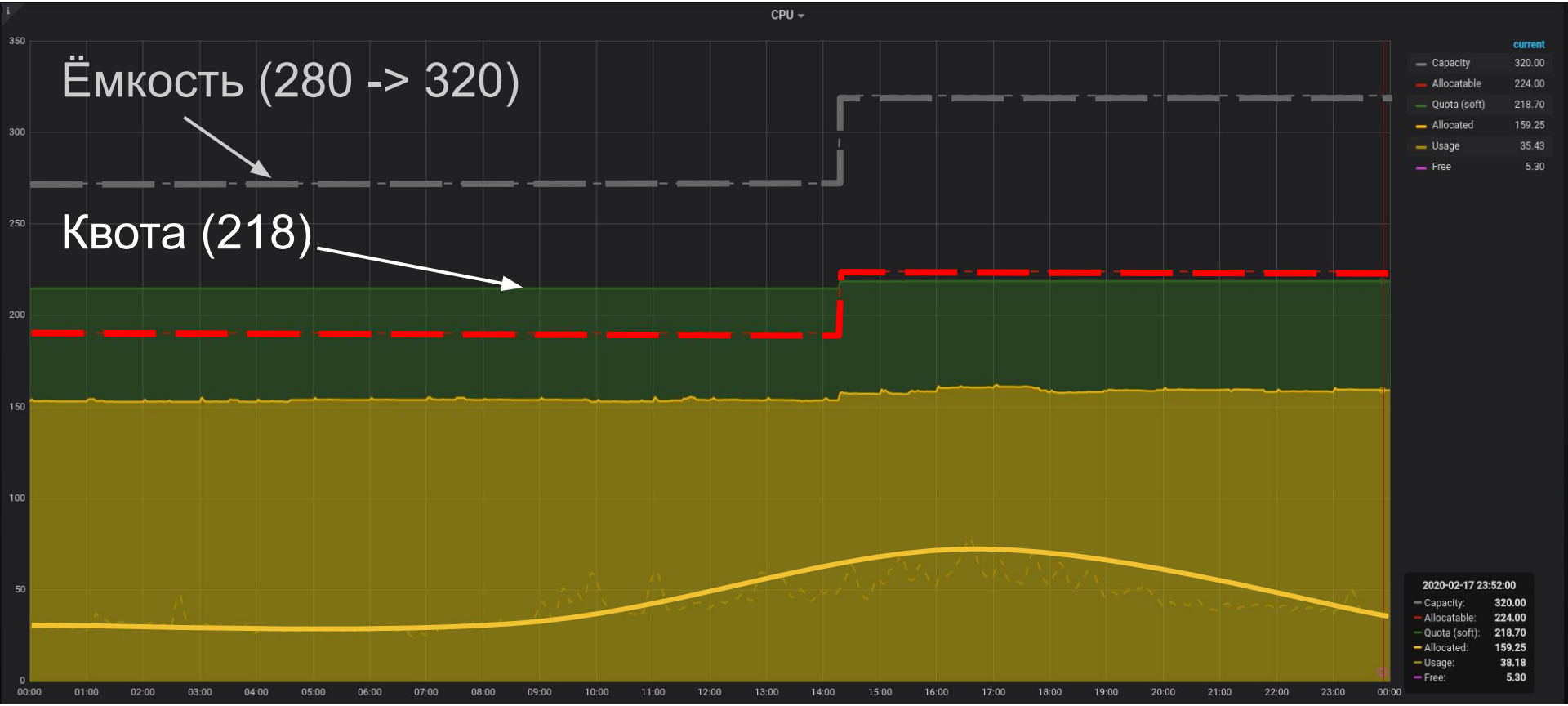
Как стало - **считаем только общие ресурсы**



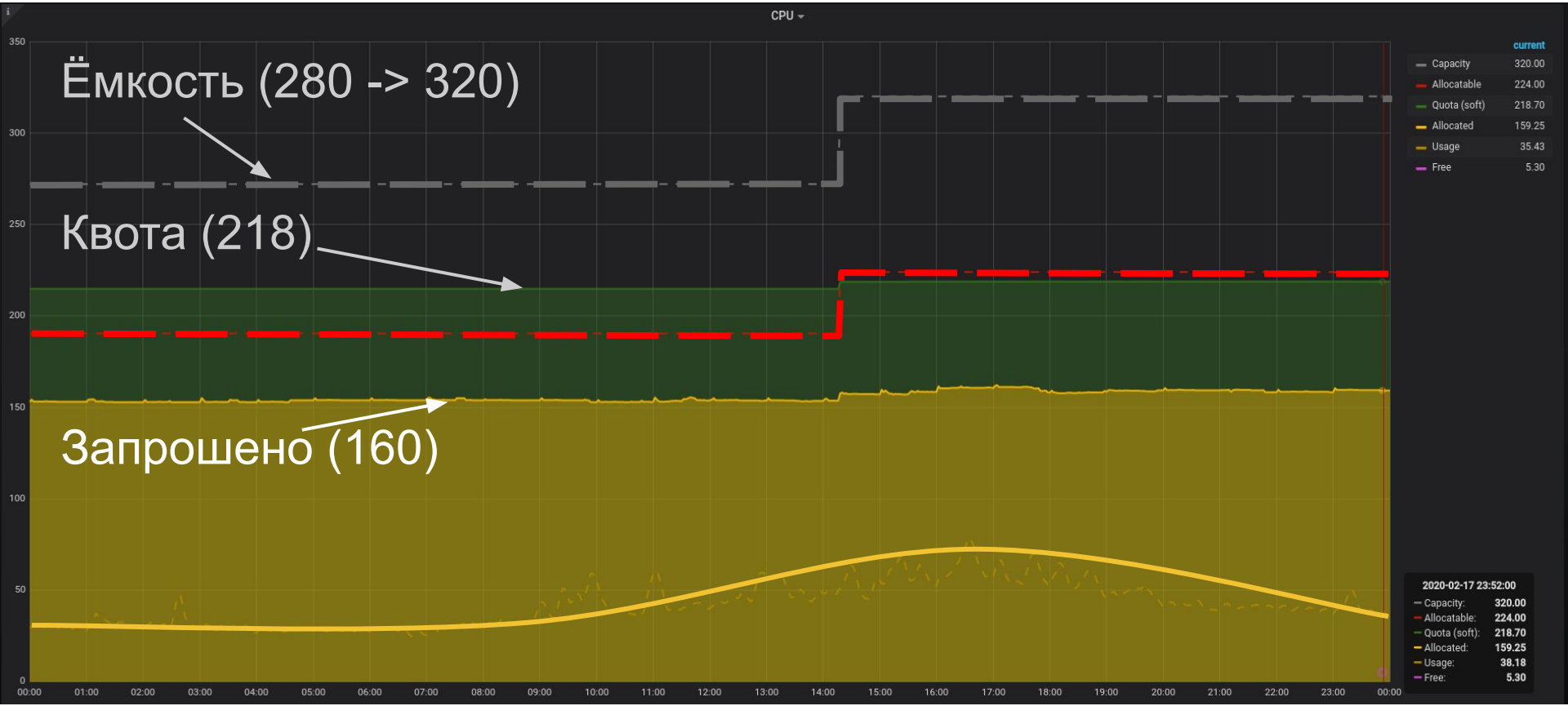
Квота кластера k8s



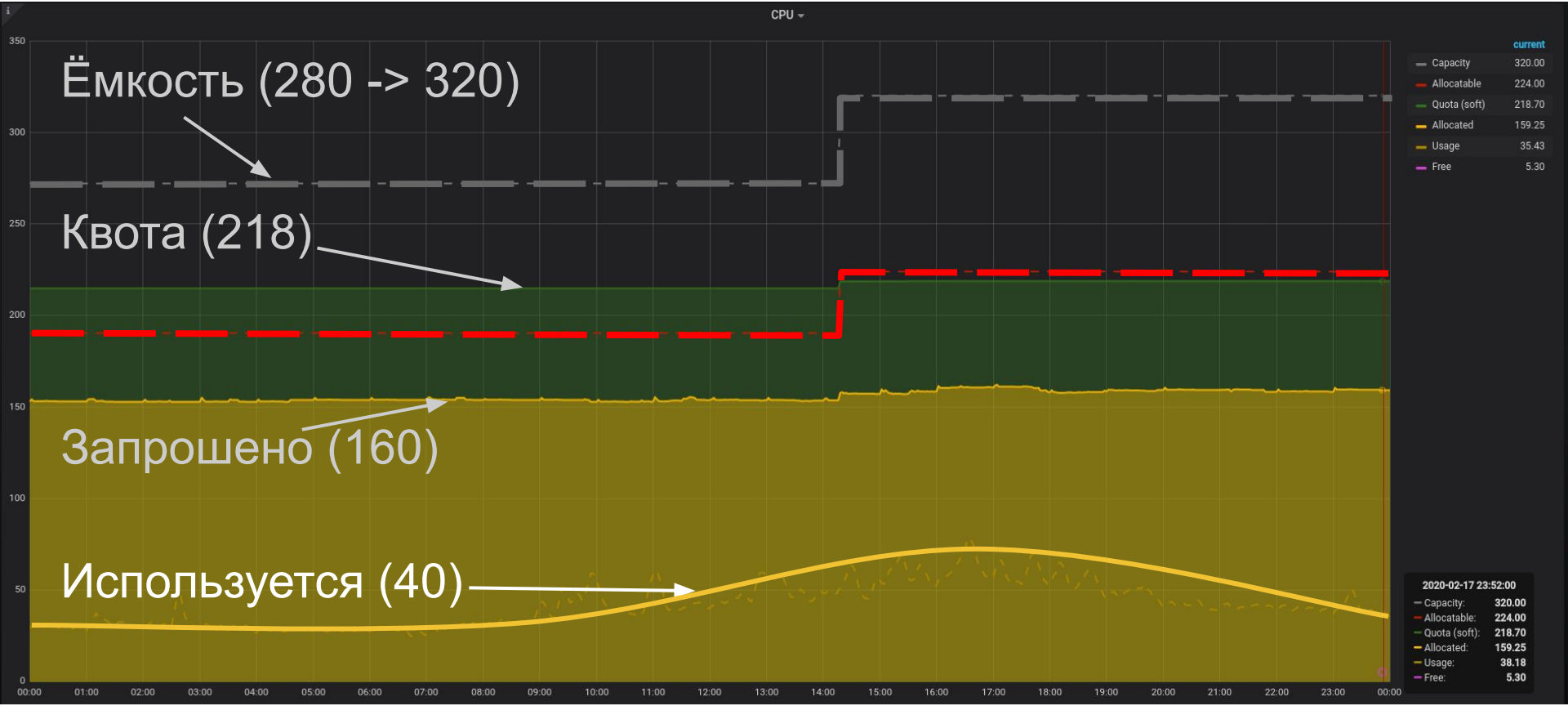
Квота кластера k8s



Квота кластера k8s

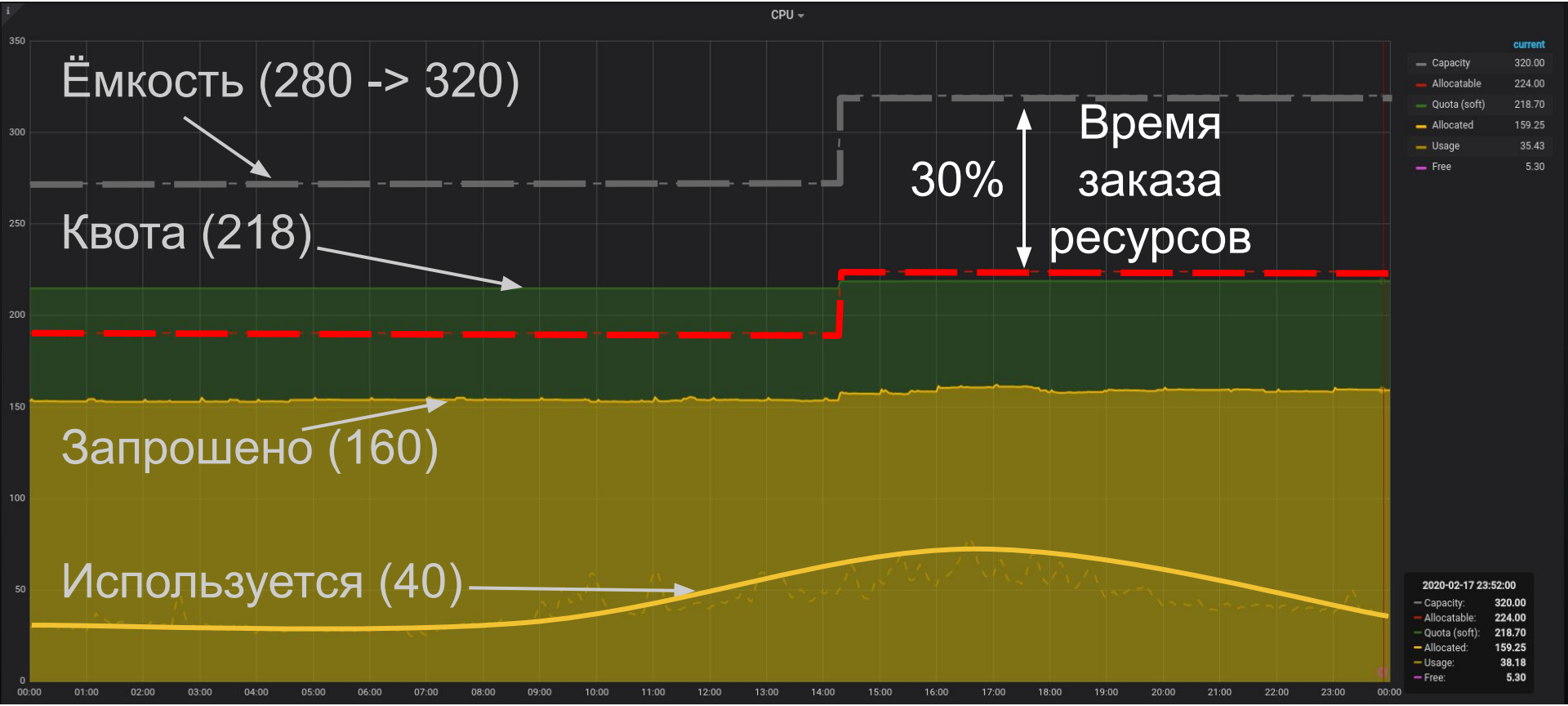


Квота кластера k8s



Квота кластера k8s

Квота кластера k8s



Теперь мы знаем

Когда выдали квот больше, чем железа

Когда осталось 30% кластера и пора покупать железо

Сколько запросили квоты с последнего заказа железа

- Новый заказ: расширение + то что запросили раньше

Что мы решили для себя

Держим запас ёмкости кластера в ~30-40%

Бережем “заначку” для временного расширения кластера

Всем сервисам продвигаем вертикальный автоскейлер

Что мы решили для себя

Держим запас ёмкости кластера в ~30-40%

Бережем “заначку” для временного расширения кластера

Всем сервисам продвигаем вертикальный автоскейлер

Следующий шаг: отказаться от квот полностью!

- Оставить обязательное проставление requests/limits
- Следить за утилизацией автоматикой, работать с командами

Выводы

Со своим железом удобно, но надо думать заранее

Некоторые решения могут стрелять по ногам

Capacity Planning - работает, когда соблюдаешь процессы

Спасибо!

Вопросы?

Дехтярёв Евгений
e.dekhtyarev@2gis.ru
github.com/dekhtyarev/devoops-readme

2GIS