




# Golang Developer. Professional

[otus.ru](https://otus.ru)

 Проверить, идет ли запись

# Меня хорошо видно && слышно?

 Ставим “+”, если все хорошо  
“-”, если есть проблемы

Тема вебинара

# Kubernetes

**Феоктистов Илья**

Senior Software Engineer at Agoda



# Правила вебинара



Активно  
участвуем



Off-topic обсуждаем  
в учебной группе



Задаем вопрос  
в чат или голосом



Вопросы вижу в чате,  
могу ответить не сразу

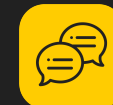
## Условные обозначения



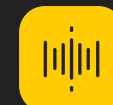
Индивидуально



Время, необходимое  
на активность



Пишем в чат



Говорим голосом



Документ



Ответьте себе или  
задайте вопрос

# О чем будем говорить:

- Что такое Кубернетес
- Архитектура
- Объекты Кубернетеса: pod, replicaSet, deployment

# Kubernetes

- Kubernetes (k8s) - инструмент контейнерной оркестрации со множеством встроенных сервисов.
- Разрабатывается Google и передан на поддержку в фонд CNCF, обладает большим комьюнити. Полностью бесплатен.

Де-факто стандарт индустрии по контейнерной оркестрации.

# kubectl

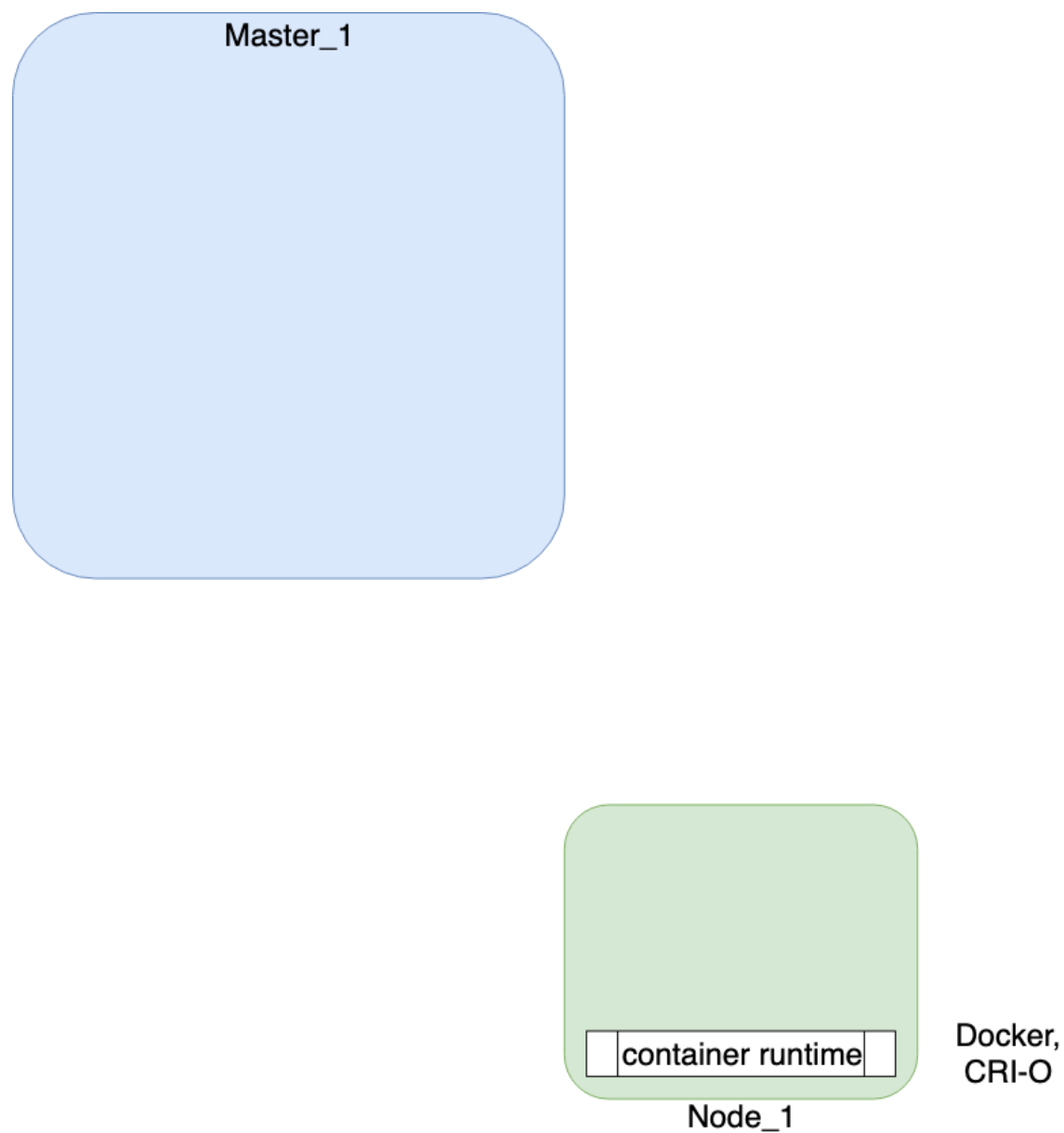
CLI утилита, распространяемая в виде бинарного файла

Объекты в кластере можно:

- Создать (kubectl create)
- Обновить (kubectl apply)
- Получить (kubectl get)
- Посмотреть (kubectl describe)
- Удалить (kubectl delete)

# Из каких компонентов состоит кластер k8s?

- Control Plane ноды
- Worker ноды

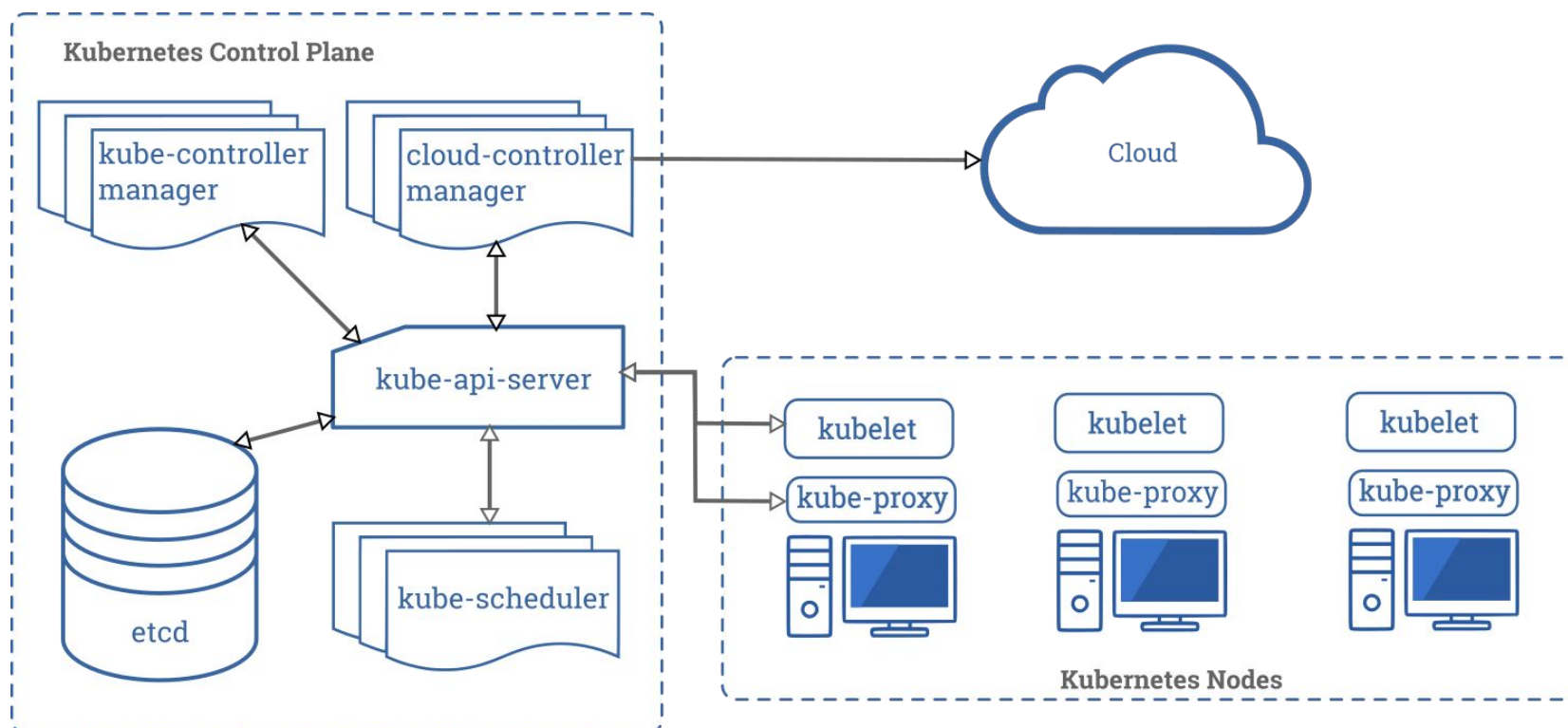




# Из каких компонентов состоит кластер k8s?

## Основные компоненты:

- etcd
- api-server
- controller-manager
- scheduler
- kubelet
- kube-proxy



# etcd

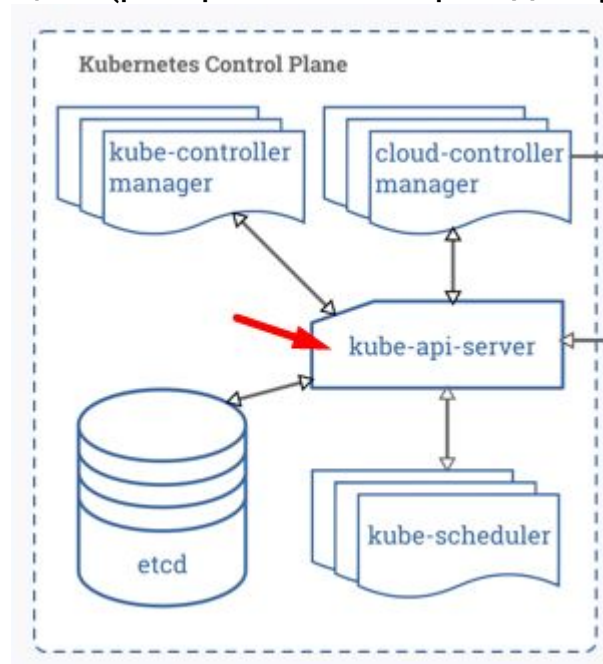
etcd – key/value база данных для хранения конфигурации кластера

- Работает по алгоритму raft (он обеспечивает надежность за счет поддержки кворума)
- Единственная база данных для хранения конфигурации, которую поддерживает k8s
- Единственный stateful-компонент
- На каждую master-ноду устанавливается по ноде etcd

# api-server

api-server – центральный, главный компонент k8s

- Stateless (в отличие от etcd)
- Взаимодействие через kubectl (но можно работать и просто curl'ом)
- Единственный компонент, который общается с etcd
- Работает по REST API
- Обеспечивает авторизацию и аутентификацию (разграничение прав доступа до содержимому кластера)



# controller-manager

controller-manager – запускает процессы набора контроллеров

В состав controller-manager'а входят следующие контроллеры:

- node-controller
- replicaset-controller
- endpoints-controller
- account-controller
- token-controller

# scheduler

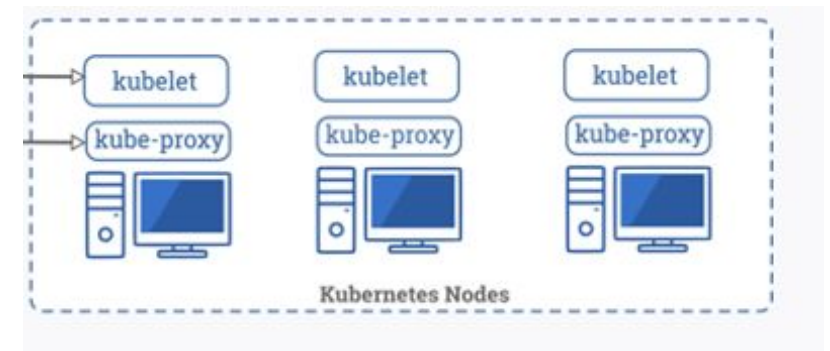
scheduler назначает поды на ноды с учетом множества факторов  
controller-manager генерирует манифесты подов, записывает данные в api-server, а scheduler назначает их на ноды, учитывая важные параметры:

- Affinity и Anti-affinity
- Requests и Limits

# Вспомогательные компоненты k8s

Помимо основных компонентов, установленных на мастер-нодах, для работы кластера необходимы дополнительные компоненты, которые устанавливаются на всех нодах (мастер и воркер):

- kubelet
- kube-proxy



# kubelet

kubelet – агент, работающий на узле кластера

- Работает на каждой ноде (и мастер и воркер)
- Не запускается в докере, работает как процесс на хосте (`systemctl status kubelet`)
- Отдает команду `docker daemon` через `docker api` (`docker run`, напр.)
- фактически реализует запуск подов на узле
- Обеспечивает проверки `liveness probe`, `readiness probe`, `startup probe`

# kubelet. Подробнее о probe

- Liveness probe – используется для определения, когда контейнер необходимо перезапустить
- Readiness probe – используется для проверки, доступен ли модуль в течение всего жизненного цикла. В отличие от liveness probe, в случае сбоя проверки останавливается только трафик к модулю, но перезапуска не происходит
- Startup probe – используется для проверки, что приложение в контейнере было запущено. Если проба настроена, то liveness и readiness проверки блокируются, до того как проба пройдет успешно

```
livenessProbe:  
  httpGet:  
    path: /healthz  
    port: liveness-port  
  failureThreshold: 1  
  periodSeconds: 10
```

```
readinessProbe:  
  exec:  
    command:  
    - cat  
    - /tmp/healthy  
  initialDelaySeconds: 5  
  periodSeconds: 5
```

```
startupProbe:  
  tcpSocket:  
    port: 8080  
  initialDelaySeconds: 15  
  periodSeconds: 10
```



# kube-proxy

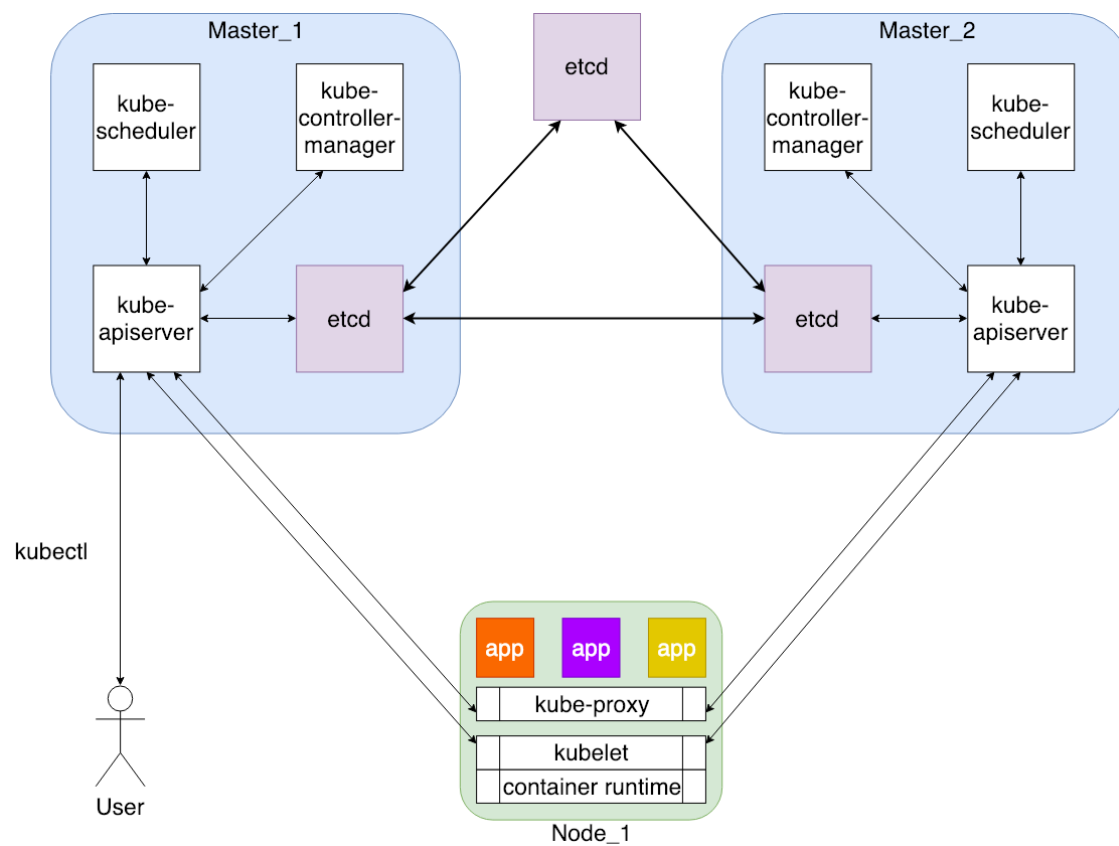
kube-proxy – сетевой прокси, работающий на каждом узле в кластере

- Взаимодействует с api-server
- Устанавливается на всех нодах
- Управляет сетевыми правилами на нодах
- Запускается в контейнере

# Подведем итог

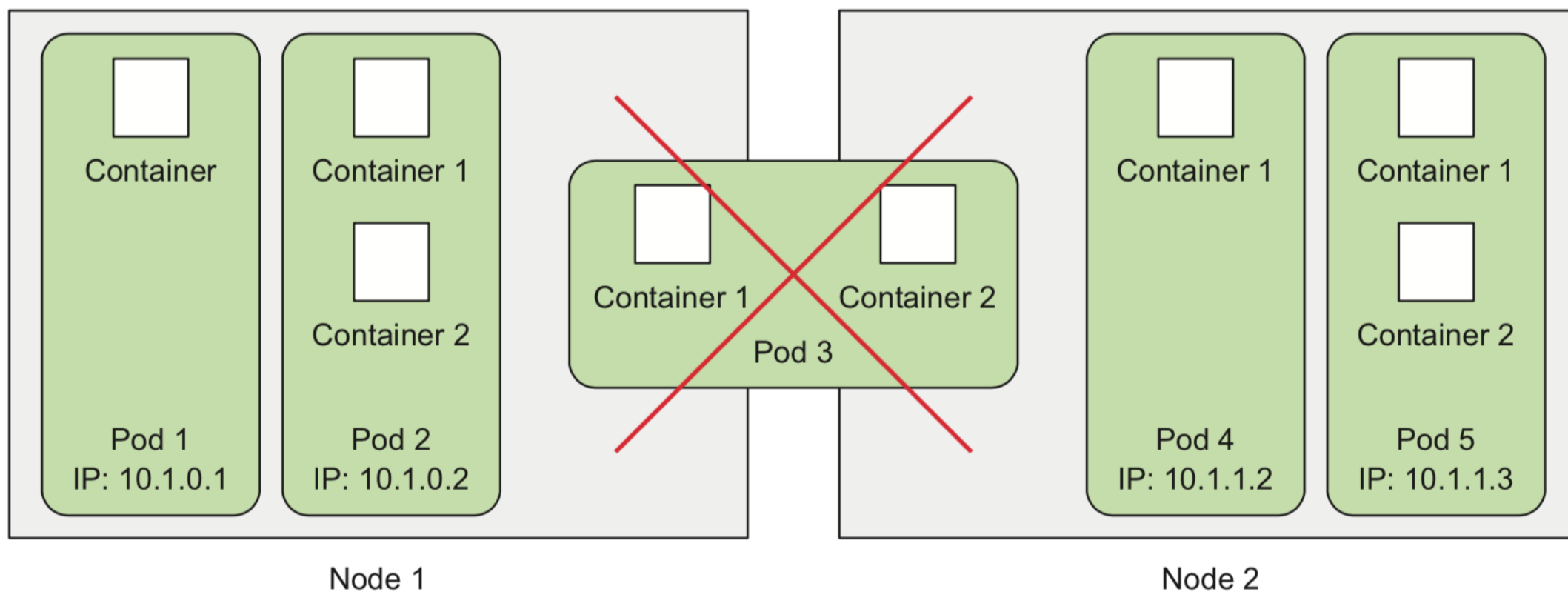
Для работоспособности кластера необходимы следующие компоненты.

- etcd
- api-server
- controller manager
- scheduler
- kubelet
- kube-proxy



# Концепция Pod

- Pod - это группа контейнеров (один или несколько)
- Минимальная сущность, управляемая Kubernetes'ом



# Контейнеры внутри одного Pod'a или разные Pod'ы?

- Сервисы должны масштабироваться вместе или по отдельности?
- Должны ли сервисы быть запущены вместе или могут быть разнесены на разные хосты?
- Это связанные сервисы или независимые компоненты?

# ReplicaSet

- Следит за тем, чтобы число подов соответствовало заданному
- Умеет пересоздавать Поды при отказе узла (обычные "голые" поды умирают вместе с нодой)
- Умеет добавлять/удалять Поды не пересоздавая всю группу
- **НЕ** проверяет соответствие запущенных Подов шаблону

# Пора деплоить...

Алгоритм:

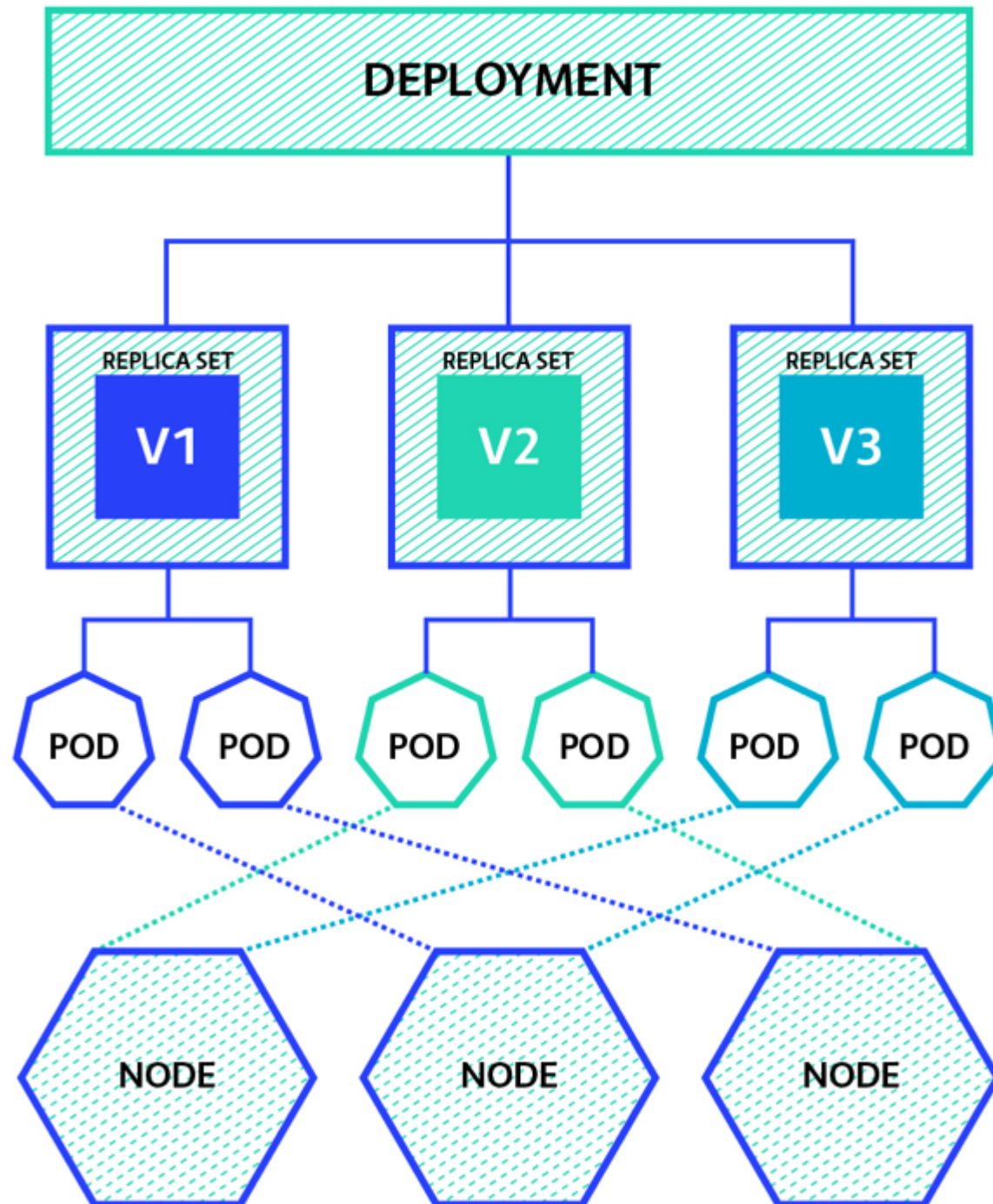
- Создать в торой ReplicaSet с новой конфигураций
- Одновременно:
  - уменьшаем replicas в старом ReplicaSet
  - увеличиваем replicas в новом ReplicaSet

Когда-то давно так работал `kubectl rolling-update` в связке с `ReplicationController`  
Главное НО - приходится это делать руками

# Deployment

- Это контроллер контроллеров, управляющий ReplicaSets
- Делает то, что описано на слайде выше, но без ручного вмешательства, внутри кластера K8s
- Соответственно, это "декларативный" способ развертывания
- И рекомендованный способ запуска Podов (даже если нужна только одна реплика)

# Связь Deployment, ReplicaSet и Pod





# Деплойменты (deployments)

За счет гарантированного поддержания определенного количества идентичных подов, Deployment позволяет выполнять следующие операции в кластере:

- Обновление (rolling update)
- Просматривать информацию о текущем обновлении (rollout status)
- Откат обновлений (rollout undo)
- История ревизий обновлений (rollout history)
- Откат (роллбэк) до конкретной ревизии обновления (rollout undo to revision)
- Скалирование рабочих нагрузок (scaling)

# Вопросы?



Ставим “+”,  
если вопросы есть



Ставим “-”,  
если вопросов нет



**Заполните, пожалуйста,  
опрос о занятии  
по ссылке в чате**

# Следующий вебинар

1 декабря

Kubernetes часть 2



Ссылка на вебинар будет в ЛК за 15 минут



Материалы к занятию в ЛК — можно изучать



Обязательный материал обозначен красной лентой

Спасибо за внимание!

# Приходите на следующие вебинары

**Илья Феокистов**

Senior Software Engineer at Agoda

