Запуск приложений в K8s

Deployment, StatefulSet, DaemonSet

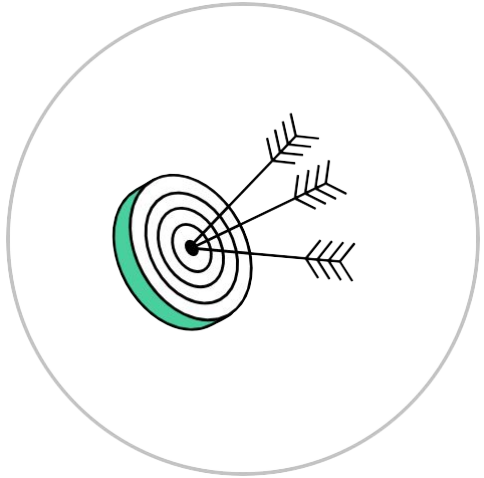
Кирилл Касаткин
DevOps-инженер, Renue

Кирилл Касаткин

DevOps-инженер, Renue

Цели занятия

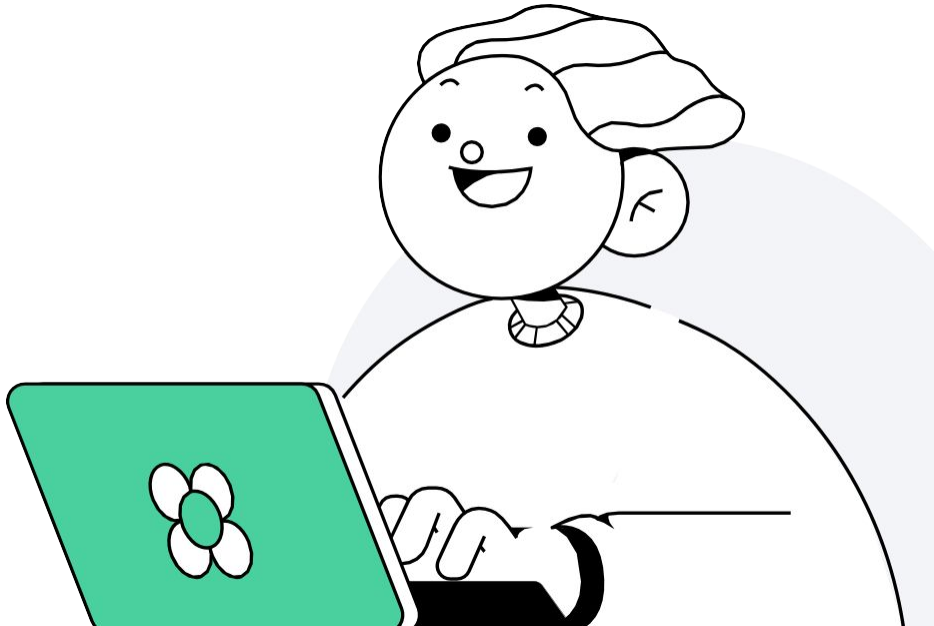
- Подробнее поговорить про Pod:
  - контейнеры внутри Pod
  - сеть внутри Pod
  - мониторинг контейнеров
- Познакомиться с объектами Kubernetes:
  - Deployment
  - StatefulSet
  - DaemonSet



План занятия

1. Namespace
2. Pod
3. Deployment
4. DaemonSet
5. Итоги
6. Домашнее задание

*Нажми на нужный раздел для перехода



Вспоминаем прошлое занятие

Вопрос: что такое Pod?

Вспоминаем прошлое занятие

Вопрос: что такое Pod?

Ответ: минимальная единица развёртывания

Вспоминаем прошлое занятие

Вопрос: что такое Service?

Вспоминаем прошлое занятие

Вопрос: что такое Service?

Ответ: средство маршрутизации или постоянный IP-адрес

Namespace

Namespace (пространство имён) обеспечивает изоляцию группы ресурсов

Использование Namespace

Namespace — это способ разделить ресурсы кластера между несколькими пользователями (с помощью квоты ресурсов).

Использование Namespace

Namespace — это способ разделить ресурсы кластера между несколькими пользователями (с помощью квоты ресурсов).

- Имена ресурсов должны быть уникальными в пределах одного и того же namespace
- Namespace не могут быть вложенными, а каждый ресурс Kubernetes может находиться только в одном Namespace

Использование Namespace

Namespace — это способ разделить ресурсы кластера между несколькими пользователями (с помощью квоты ресурсов).

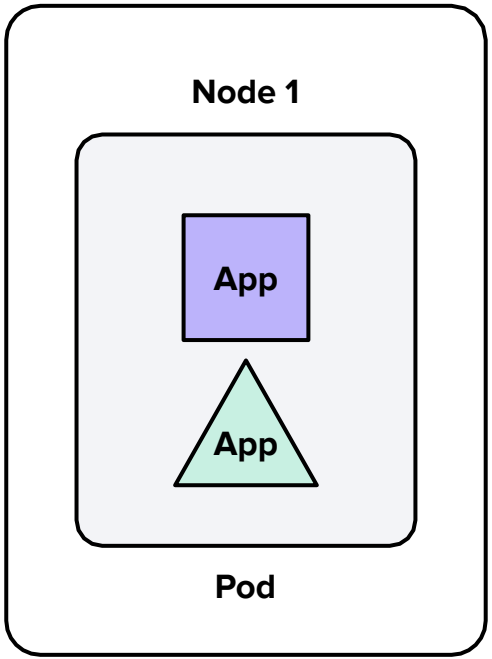
- Имена ресурсов должны быть уникальными в пределах одного и того же namespace
- Namespace не могут быть вложенными, а каждый ресурс Kubernetes может находиться только в одном Namespace

```
kubectl get namespace
```

Pod

Типы контейнеров в Pod

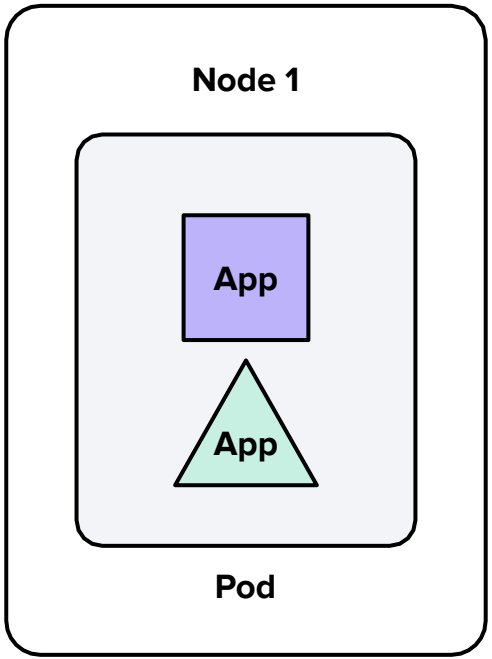
Есть два типа контейнеров в Pod:



- Init-контейнеры

Типы контейнеров в Pod

Есть два типа контейнеров в Pod:

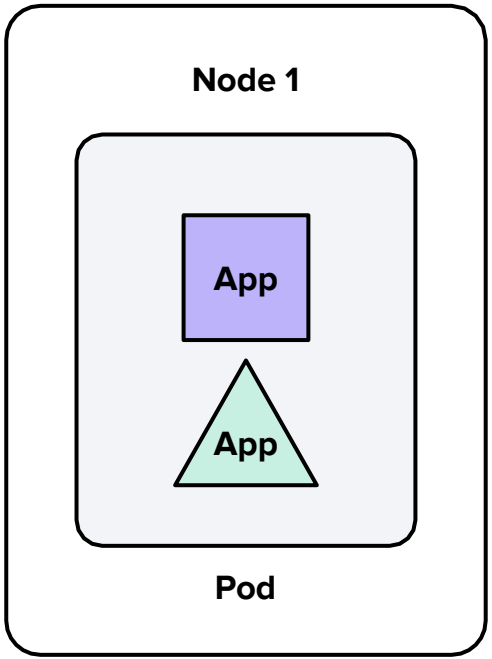


Init-контейнеры

- запускаются последовательно
- запускаются перед основными контейнерами
- могут шарить общие тома с контейнерами
- требуются для начальной настройки

Типы контейнеров в Pod

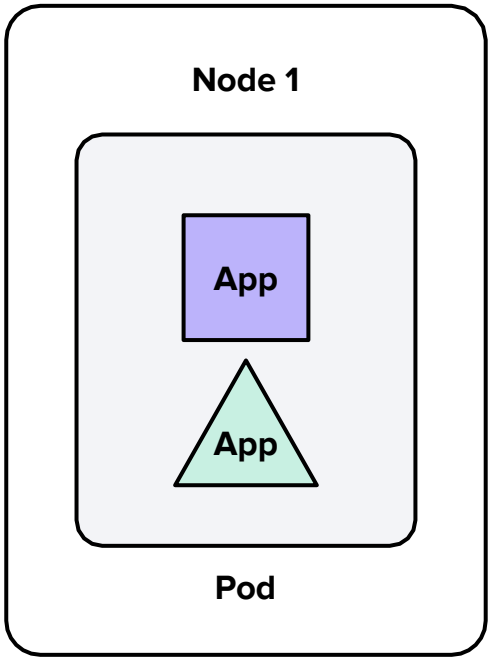
Есть два типа контейнеров в Pod:



- Init-контейнеры
  - запускаются последовательно
  - запускаются перед основными контейнерами
  - могут шарить общие тома с контейнерами
  - требуются для начальной настройки
- Runtime-контейнеры

Типы контейнеров в Pod

Есть два типа контейнеров в Pod:



Init-контейнеры

- запускаются последовательно
- запускаются перед основными контейнерами
- могут шарить общие тома с контейнерами
- требуются для начальной настройки

Runtime-контейнеры

- запускаются после init-контейнеров
- запускаются одновременно

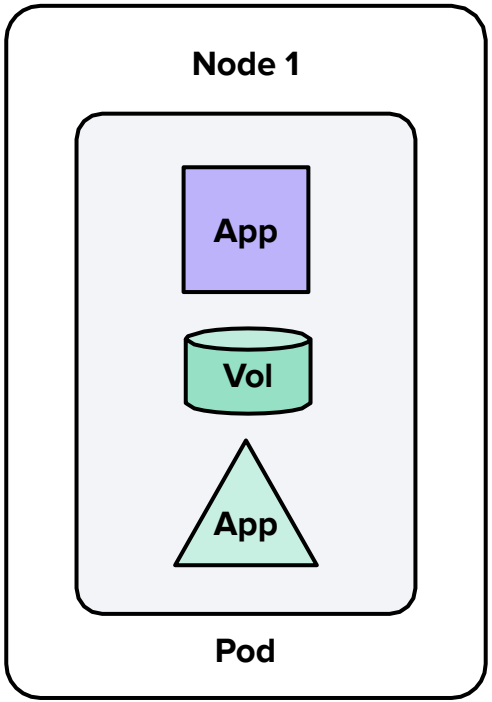
Пример конфигурации

Пример конфигураций Pod с init-контейнером:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: init-pod
spec:
  containers:
  - name: nginx
    image: nginx
  initContainers:
  - name: delay
    image: busybox
    command: ['sleep', '30']
```

Файловая система внутри Pod

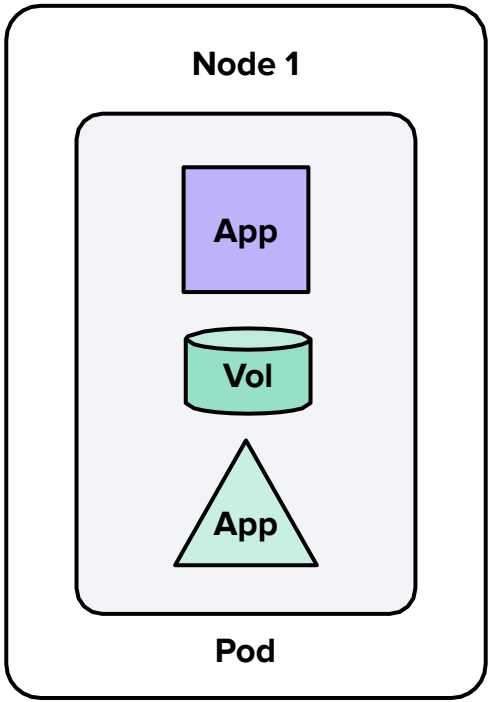
У контейнеров внутри Pod разная файловая система.



- Внутри можно создать общую папку или другой том

Файловая система внутри Pod

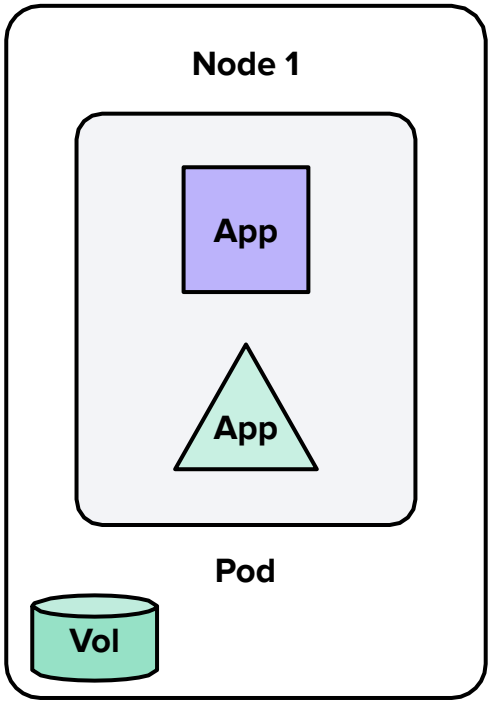
У контейнеров внутри Pod разная файловая система.



- Внутри можно создать общую папку или другой том
- Тома могут быть временными или постоянными
  - временные тома при перезапуске пода очищаются

Файловая система внутри Pod

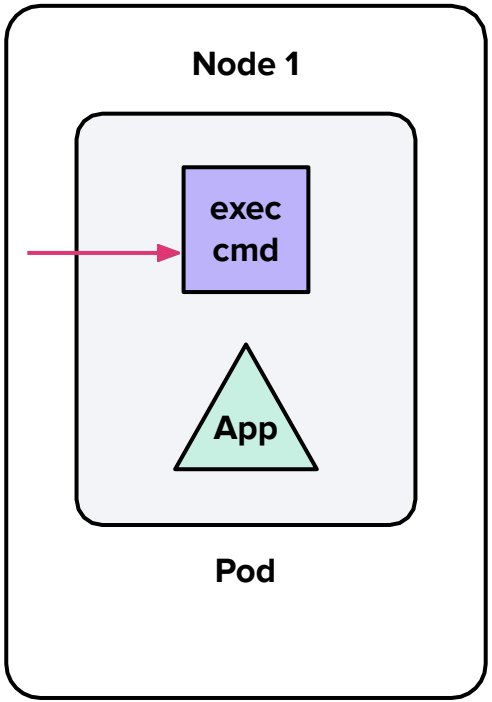
У контейнеров внутри Pod разная файловая система.



- Внутри можно создать общую папку или другой том
- Тома могут быть временными или постоянными
  - временные тома при перезапуске пода очищаются
  - постоянные остаются даже в случае удаления пода

Проверка на работоспособность и готовность

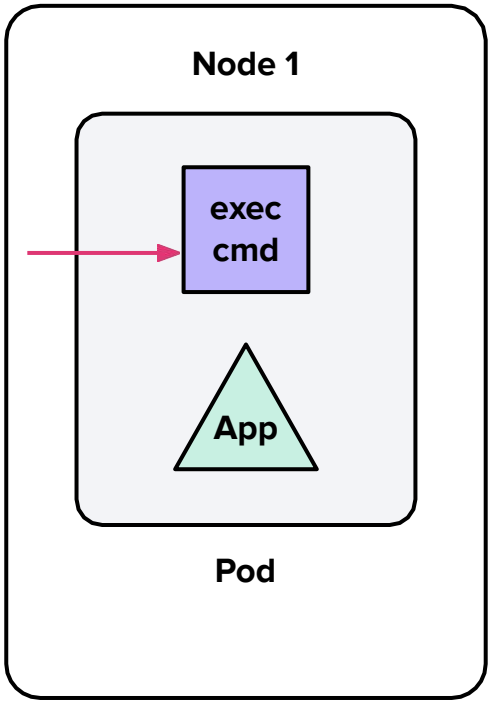
Мониторить состояние контейнеров в Pod можно с помощью Probes (зондов).



- **liveness probes** — нужна для проверки работы и перезапуска пода в случае проблем

Проверка на работоспособность и готовность

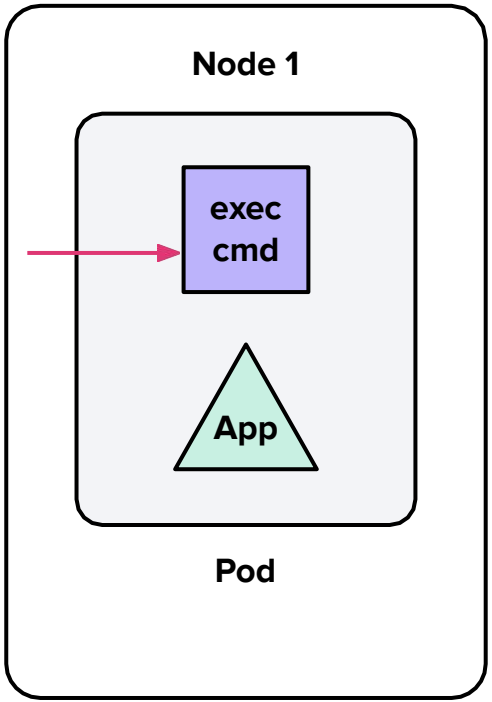
Мониторить состояние контейнеров в Pod можно с помощью Probes (зондов).



- **liveness probes** — нужна для проверки работы и перезапуска пода в случае проблем
- **startup probes** — аналогична liveness probes, но останавливается после успешного запуска пода

Проверка на работоспособность и готовность

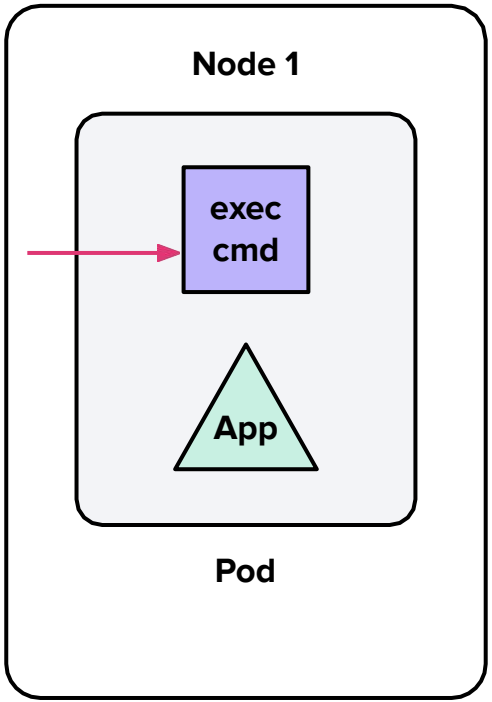
Мониторить состояние контейнеров в Pod можно с помощью Probes (зондов).



- **liveness probes** — нужна для проверки работы и перезапуска пода в случае проблем
- **startup probes** — аналогична liveness probes, но останавливается после успешного запуска пода
- **readiness** — нужны для ожидания запуска перед обслуживанием трафика

Проверка на работоспособность и готовность

Мониторить состояние контейнеров в Pod можно с помощью Probes (зондов).



- **liveness probes** — нужна для проверки работы и перезапуска пода в случае проблем
- **startup probes** — аналогична liveness probes, но останавливается после успешного запуска пода
- **readiness** — нужны для ожидания запуска перед обслуживанием трафика

Все зонды запускаются с периодичностью, которая определена

в спецификации

Пример конфигурации

Примеры конфигураций Pod с Probes:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: liveness-pod-http
spec:
  containers:
  - name: nginx
    image: nginx
    livenessProbe:
      httpGet:
        path: /
        port: 80
      initialDelaySeconds: 5
      periodSeconds: 5
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: startup-pod
spec:
  containers:
  - name: nginx
    image: nginx
    startupProbe:
      httpGet:
        path: /
        port: 80
      failureThreshold: 30
      periodSeconds: 10
```

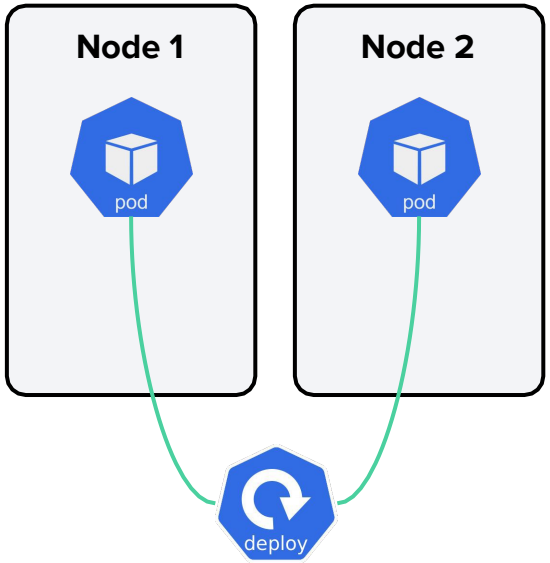
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: startup-pod
spec:
  containers:
  - name: nginx
    image: nginx
    readinessProbe:
      httpGet:
        path: /
        port: 80
      initialDelaySeconds: 5
      periodSeconds: 5
```

Deployment

Deployment — объект K8s, в котором хранится описание подов, количество реплик и алгоритм их замены в случае изменения параметров

Возможности и особенности Deployment

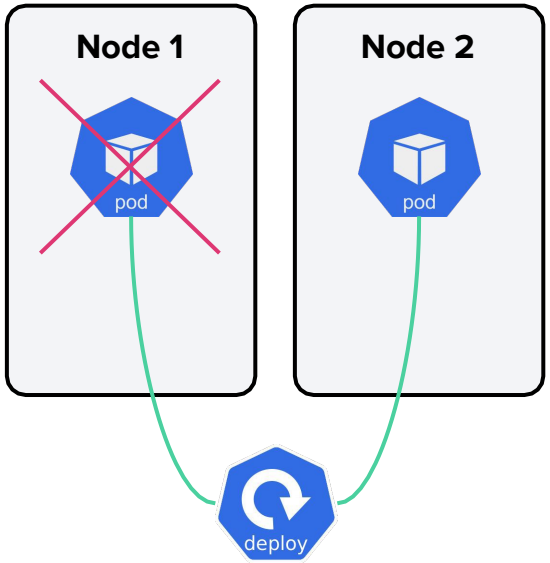
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов

Возможности и особенности Deployment

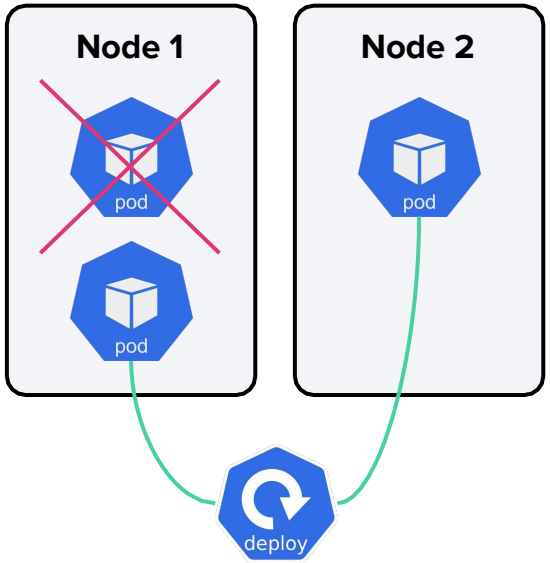
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов

Возможности и особенности Deployment

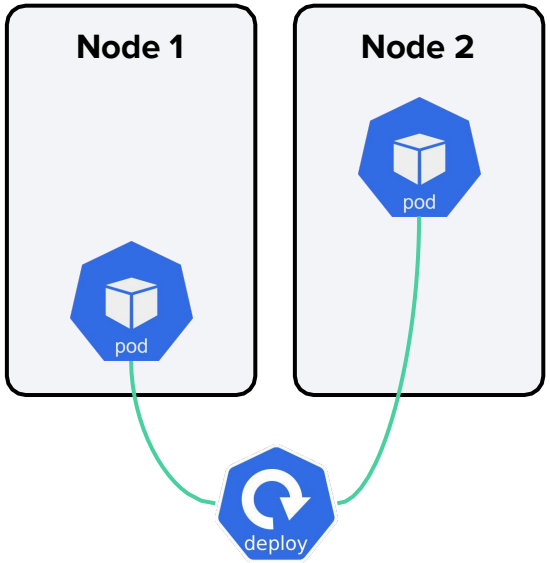
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов

Возможности и особенности Deployment

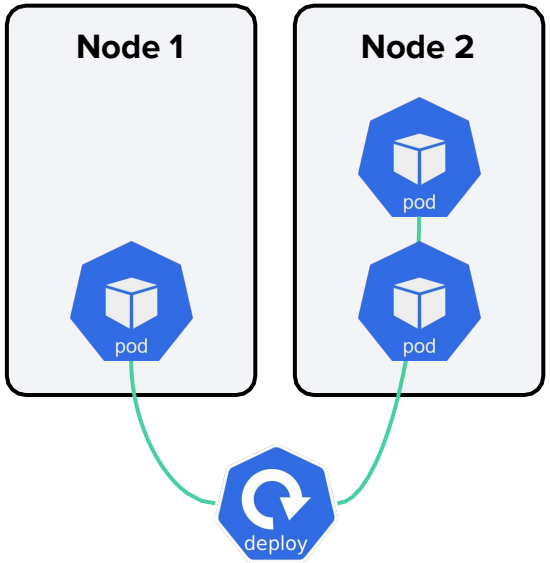
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз

Возможности и особенности Deployment

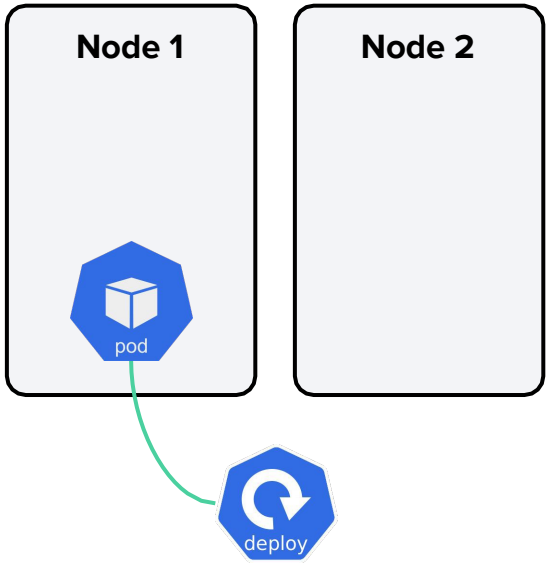
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз

Возможности и особенности Deployment

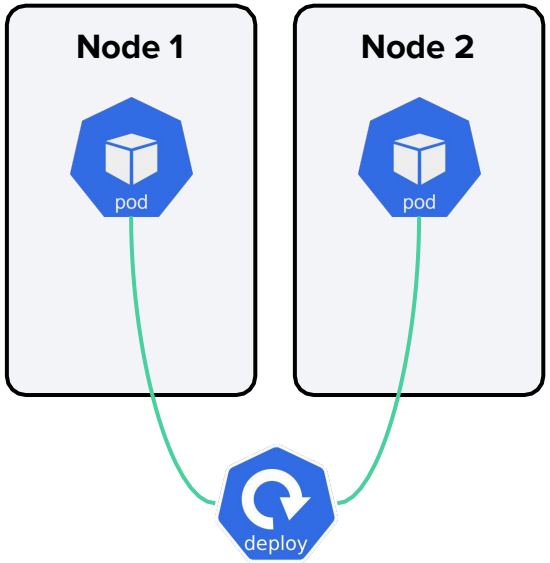
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз

Возможности и особенности Deployment

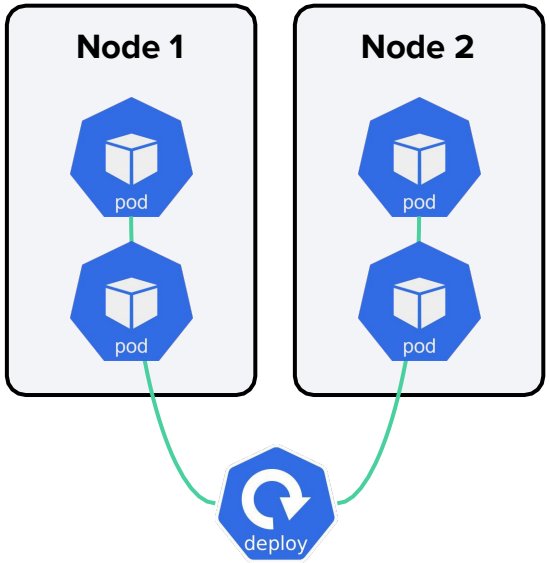
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз
- хранит шаблон конфигурации пода и позволяет обновлять и откатывать его

Возможности и особенности Deployment

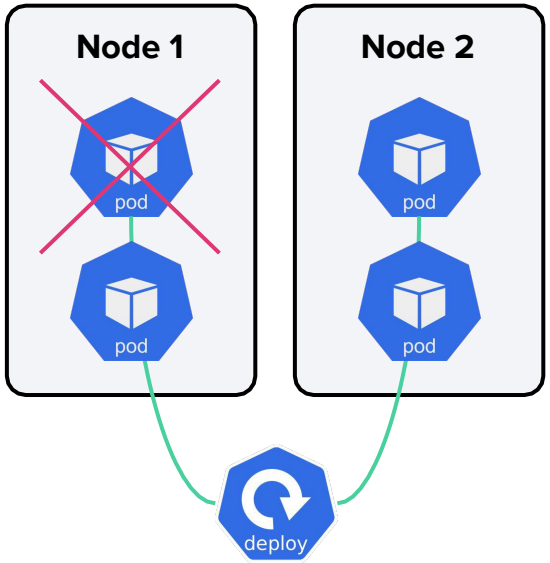
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз
- хранит шаблон конфигурации пода и позволяет обновлять и откатывать его

Возможности и особенности Deployment

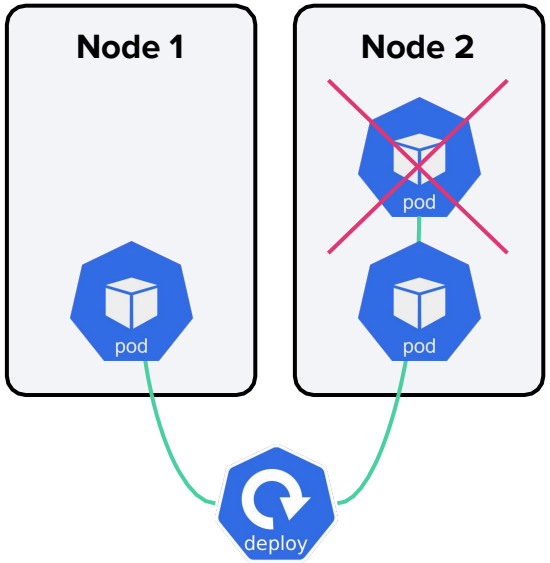
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз
- хранит шаблон конфигурации пода и позволяет обновлять и откатывать его

Возможности и особенности Deployment

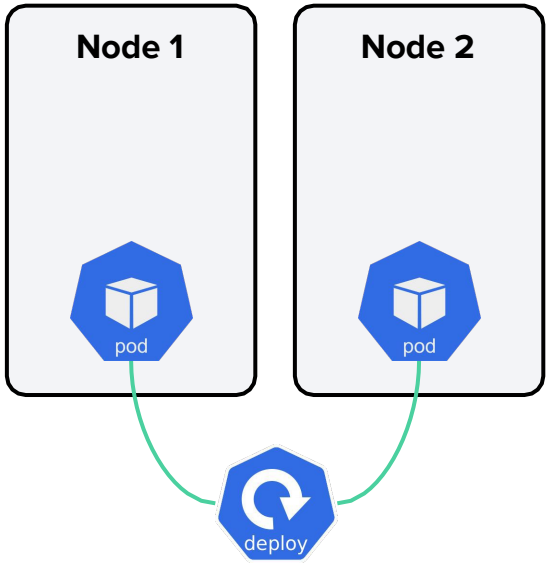
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз
- хранит шаблон конфигурации пода и позволяет обновлять и откатывать его

Возможности и особенности Deployment

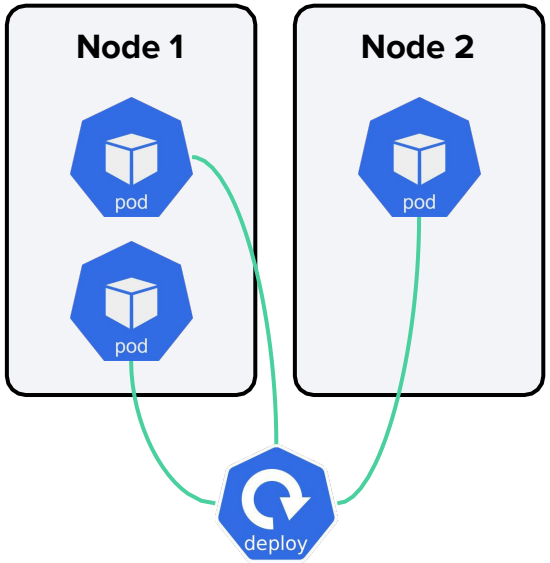
Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз
- хранит шаблон конфигурации пода и позволяет обновлять и откатывать его

Возможности и особенности Deployment

Следит за требуемым состоянием приложения, создавая, удаляя и заменяя поды с новой конфигурацией:



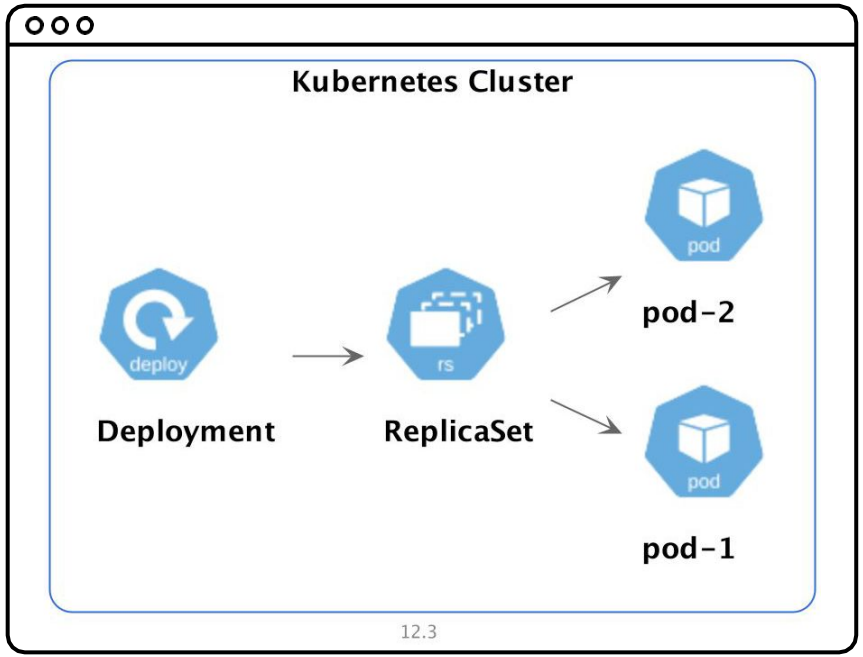
- следит за количеством и статусом запущенных подов
- масштабирует приложение как вверх, так и вниз
- хранит шаблон конфигурации пода и позволяет обновлять и откатывать его
- работает поверх **ReplicaSet**

```
kubectl get deployments
```

ReplicaSet — объект K8s, отвечающий за описание и контроль за несколькими экземплярами (репликами) подов, созданных на кластере

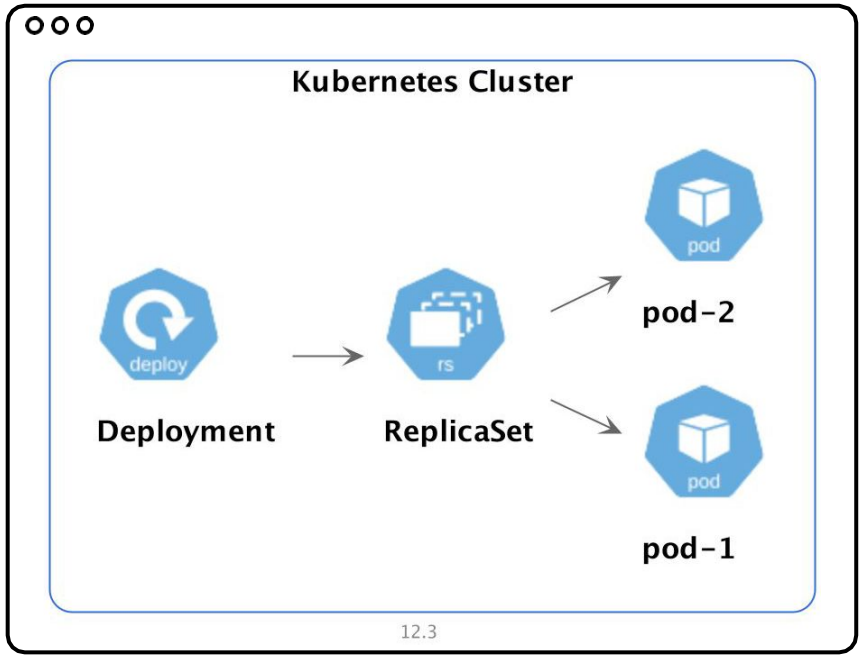
Особенности ReplicaSet

ReplicaSet гарантирует, что указанное количество реплик пода запущены в данный момент времени.



Особенности ReplicaSet

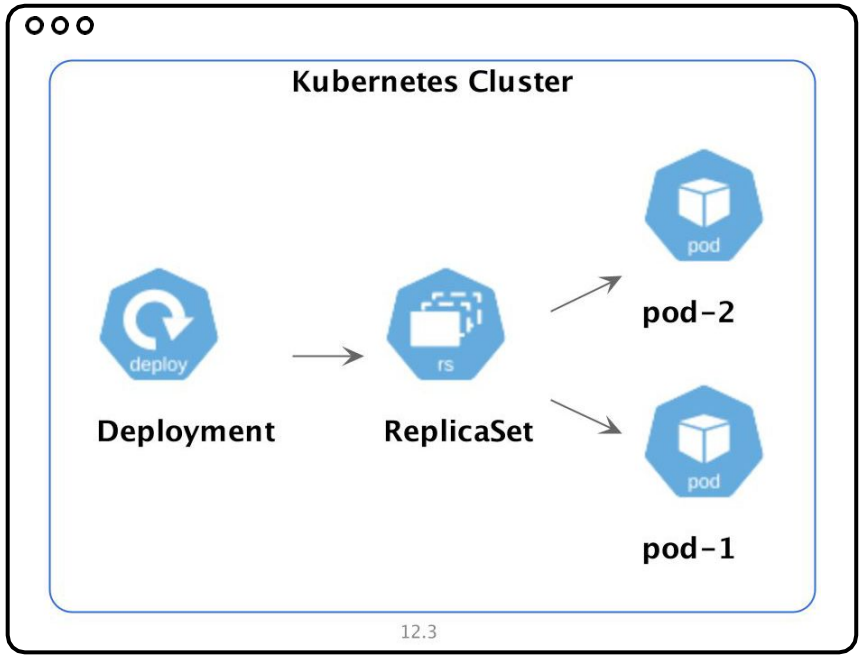
ReplicaSet гарантирует, что указанное количество реплик пода запущены в данный момент времени.



- В один и тот же момент времени могут существовать несколько ReplicaSet со своим набором подов

Особенности ReplicaSet

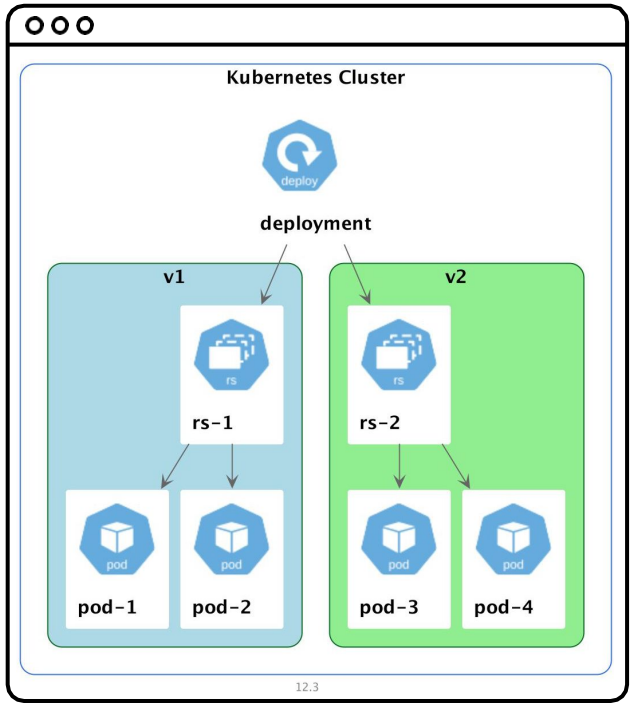
ReplicaSet гарантирует, что указанное количество реплик пода запущены в данный момент времени.



- В один и тот же момент времени могут существовать несколько ReplicaSet со своим набором подов
- На практике ReplicaSet создаётся с использованием Deployment и редко используется отдельно

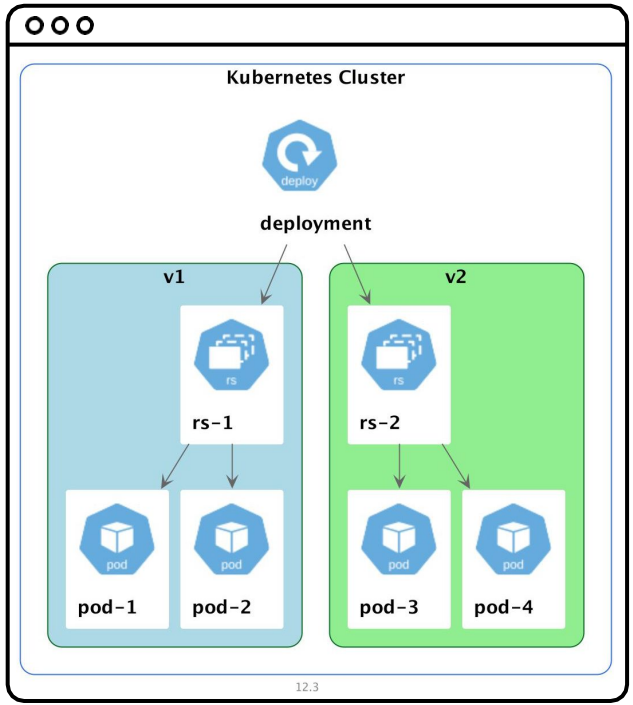
Обновление с помощью Deployment

При необходимости обновить версию ПО достаточно изменить Deployment.



Обновление с помощью Deployment

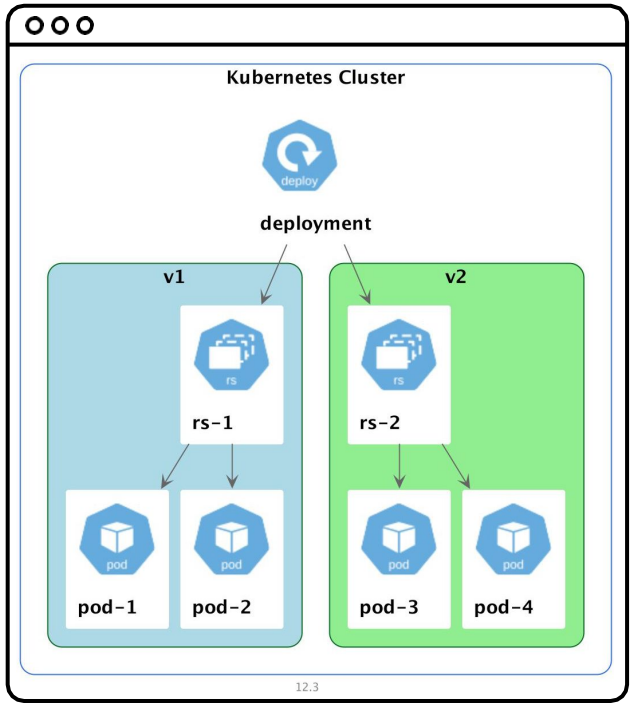
При необходимости обновить версию ПО достаточно изменить Deployment.



- Вы указываете образ новой версии в Deployment и дальнейший процесс происходит без вашего участия

Обновление с помощью Deployment

При необходимости обновить версию ПО достаточно изменить Deployment.



- Вы указываете образ новой версии в Deployment и дальнейший процесс происходит без вашего участия

- После обновления Deployment будут существовать две версии ReplicaSet с одинаковым желаемым числом реплик:
  - сначала будут созданы поды для нового ReplicaSet (rs-2)
  - после их готовности принимать трафик запустится процесс уменьшения числа реплик у старой версии ReplicaSet (rs-1)

Пример конфигурации

Примеры манифестов Deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
      app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels: app:
        nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

- **replicas** — кол-во реплик приложения (пода)

Пример конфигурации

Примеры манифестов Deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
      app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels: app:
        nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

- **replicas** — кол-во реплик приложения (пода)
- **selector** — используется для идентификации реплик подов, управляемых Deployment

Пример конфигурации

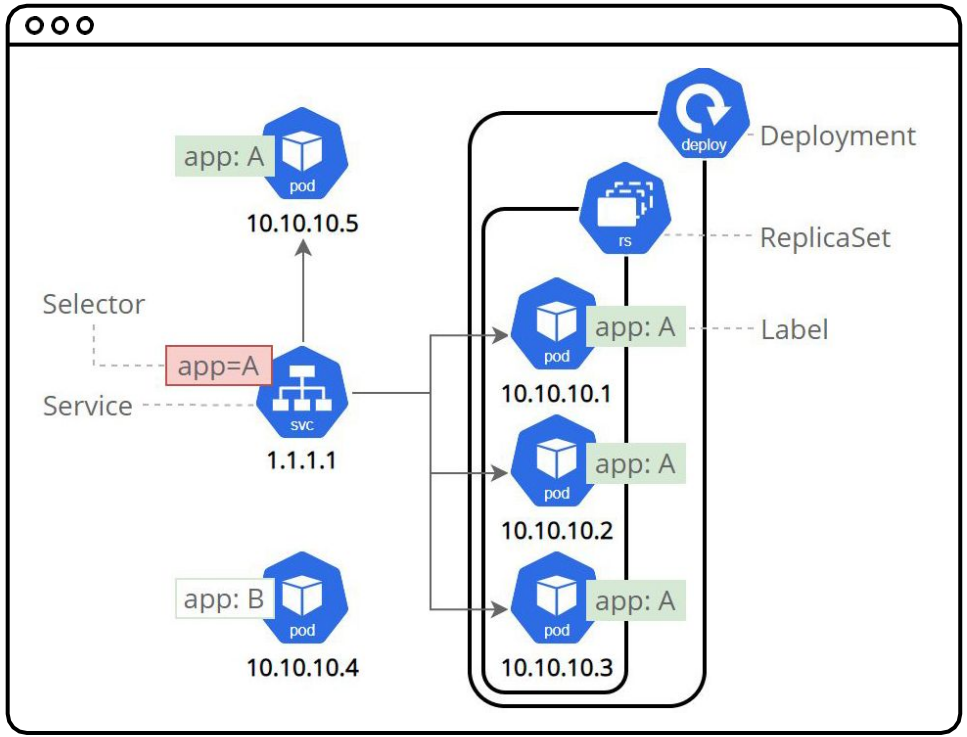
Примеры манифестов Deployment:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
      app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels: app:
        nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

- **replicas** — кол-во реплик приложения (пода)
- **selector** — используется для идентификации реплик подов, управляемых Deployment
- **template** — шаблон конфигурации пода

Схема компонентов

Взаимосвязь между объектами:

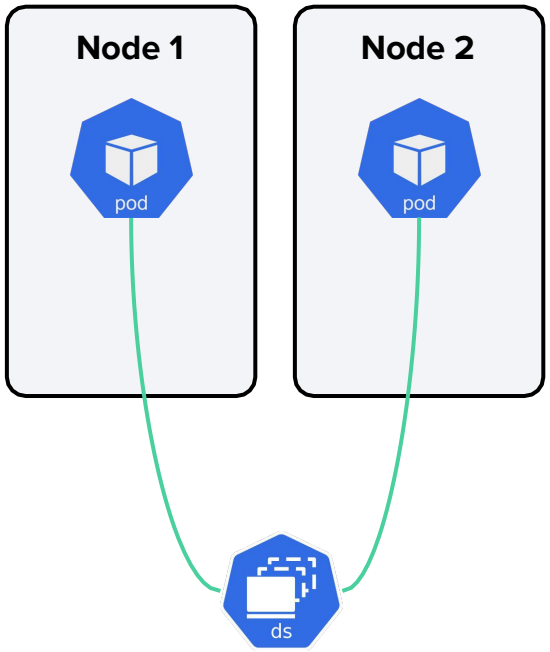


DaemonSet

DaemonSet — объект, отвечающий за то, чтобы на каждой отдельной ноде или ряде выбранных запускался один экземпляр выбранного пода

Возможности и особенности DaemonSet

Гарантирует запуск по одному экземпляру на каждой ноде:

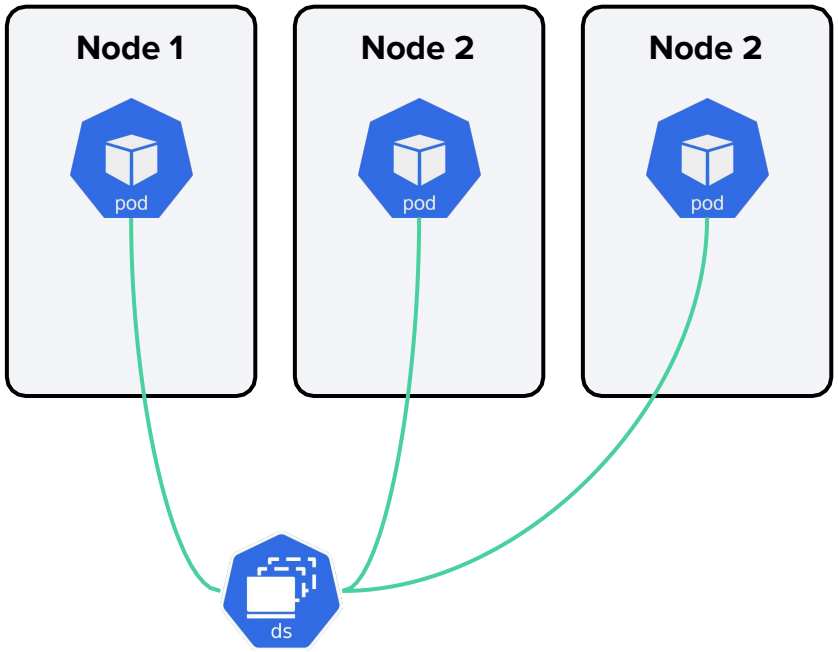


- обычно используется для системных целей

```
kubectl get ds
```

Возможности и особенности DaemonSet

Гарантирует запуск по одному экземпляру на каждой ноде:

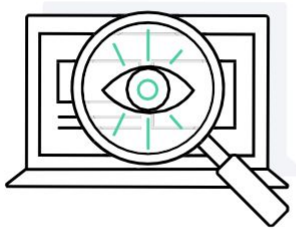


- обычно используется для системных целей
- при подключении новой ноды запустит экземпляр пода автоматически

```
kubectl get ds
```

Демонстрация работы

Работа с объектами K8s



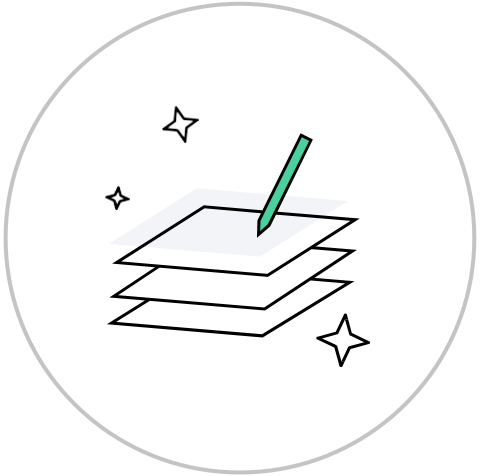
Итоги

1. Узнали, что такое Namespace, Deployment, StatefulSet, DaemonSet
2. Разобрались, какие типы контейнеров бывают в Pod
3. Поняли, как можно обновлять приложения
4. Рассмотрели примеры манифестов объектов K8s
5. Попробовали подключиться к кластеру и посмотреть в работе объекты, изученные на занятии

Домашнее задание

Давайте посмотрим ваше домашнее задание

1. Вопросы о домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



Задавайте вопросы и пишите отзыв о лекции