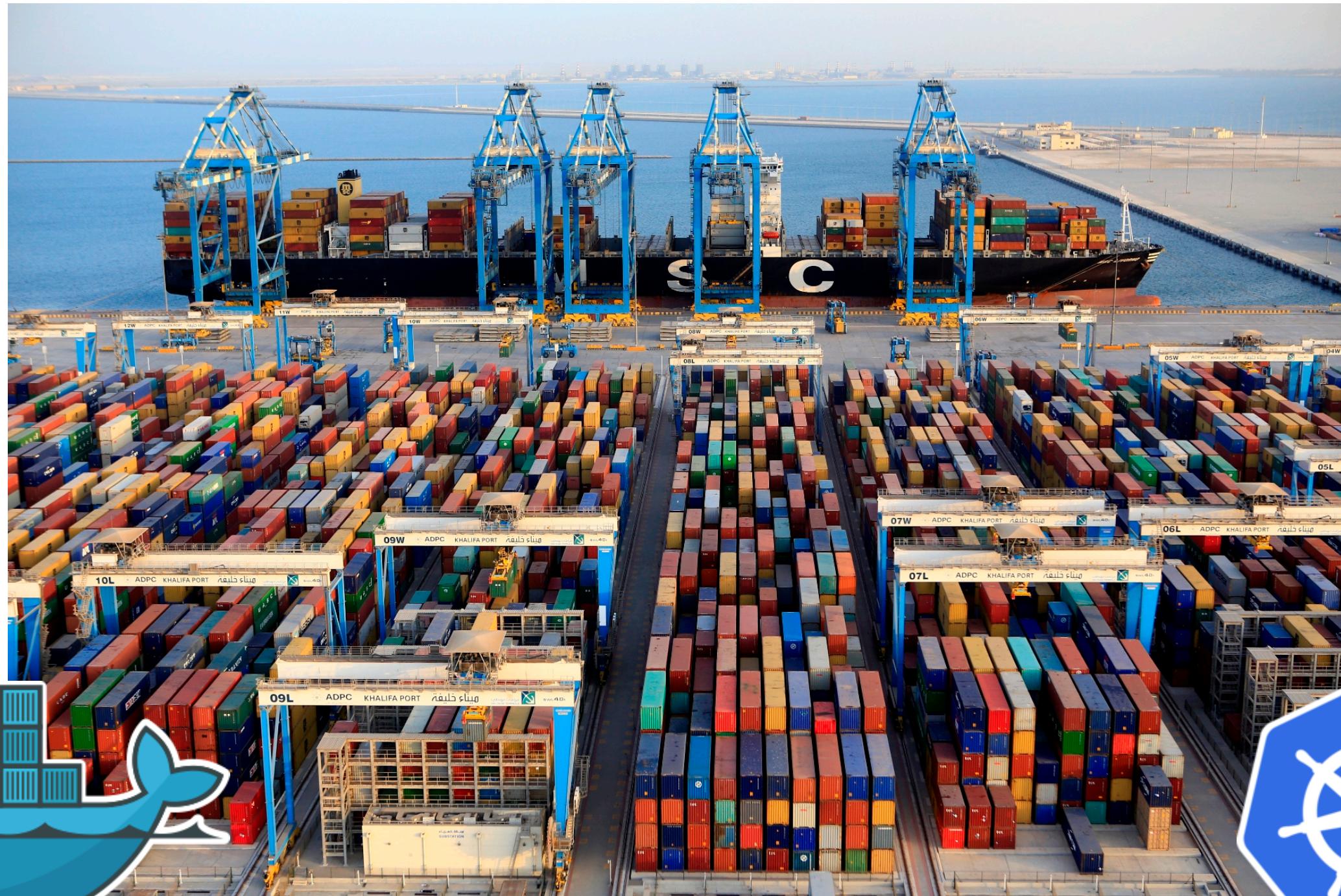
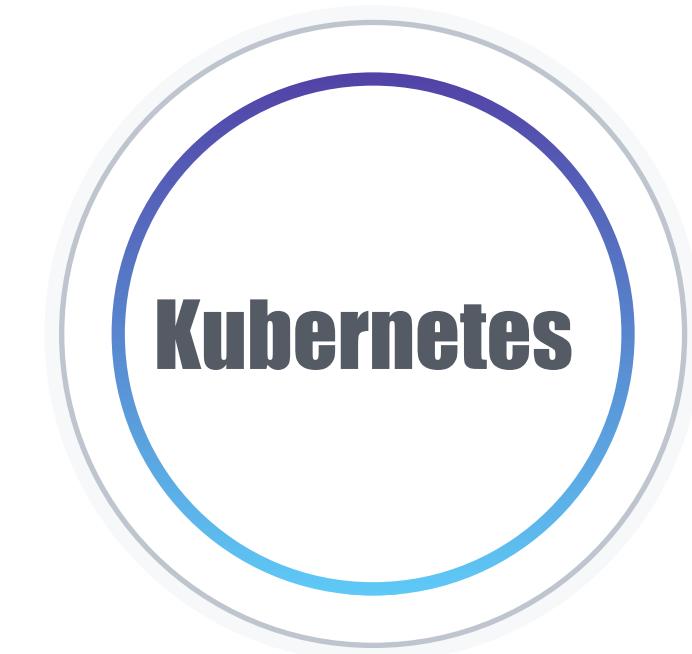


Знакомство с Kubernetes

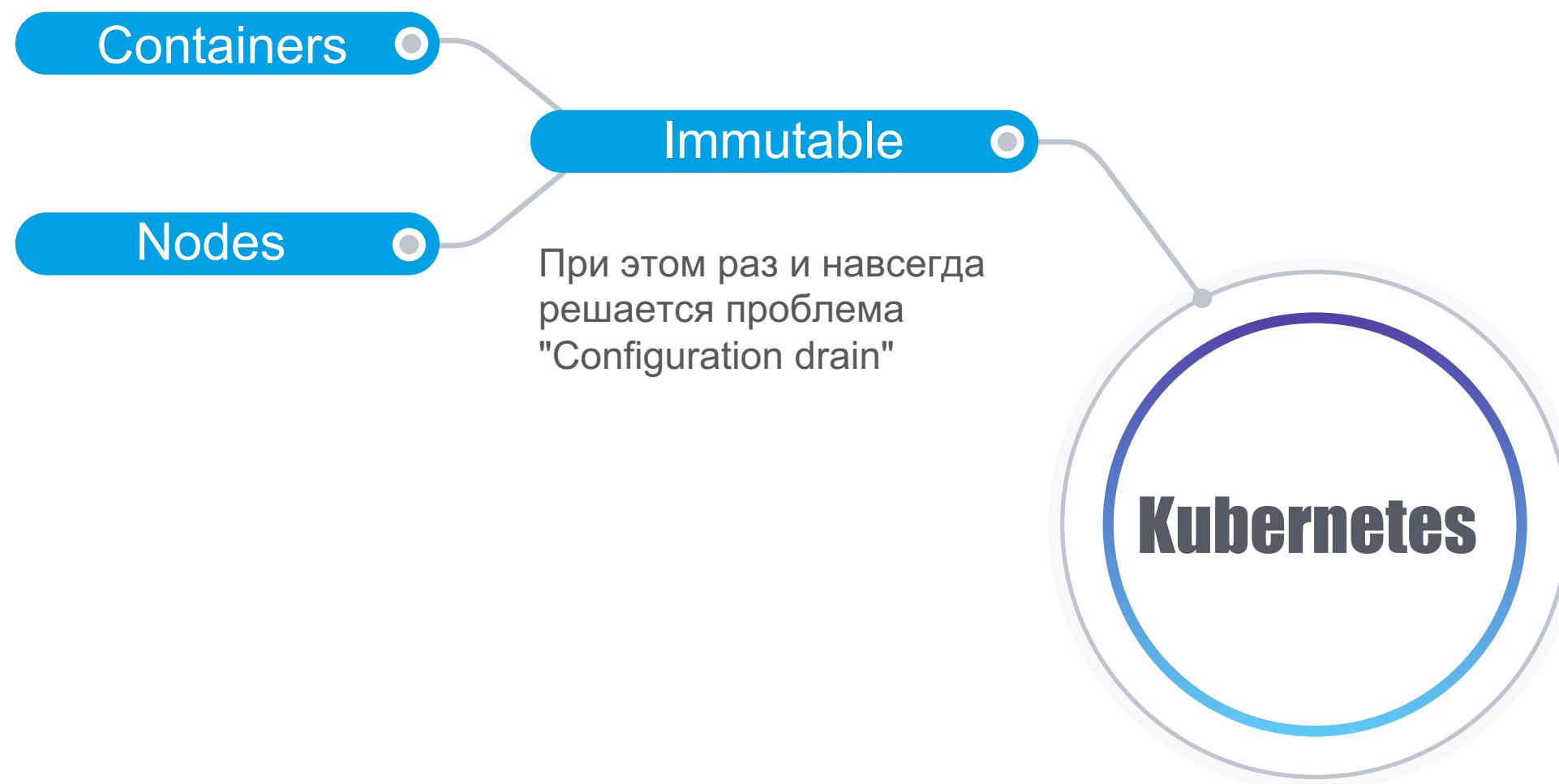
Kubernetes



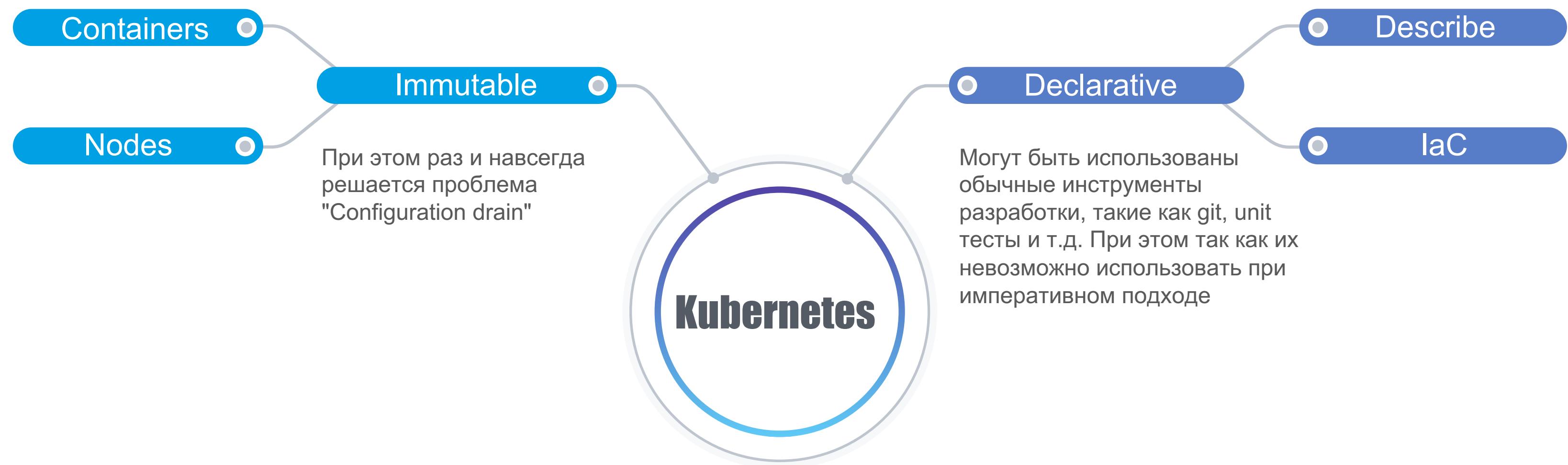
Преимущества Kubernetes



Преимущества Kubernetes



Преимущества Kubernetes



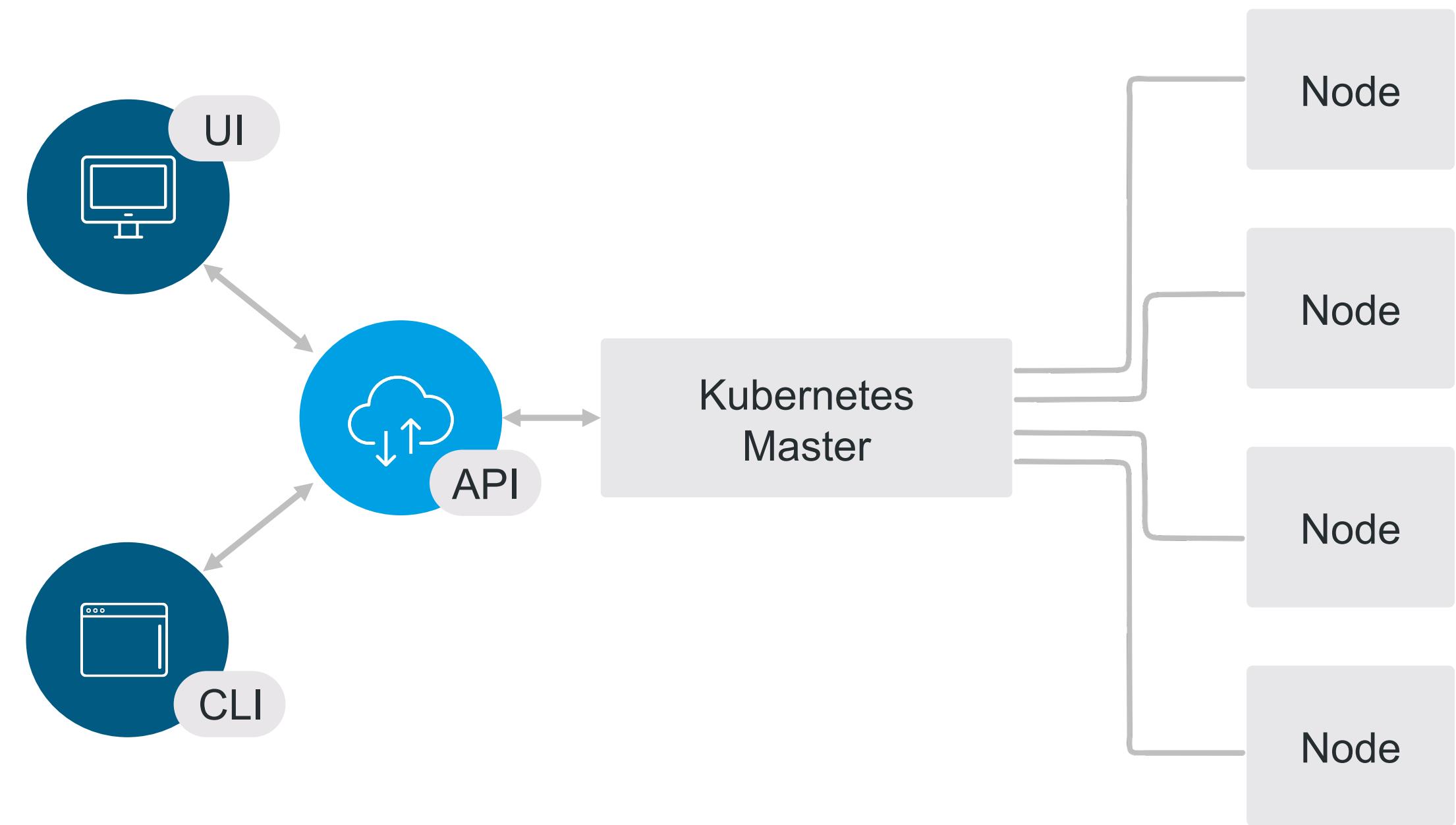
Преимущества Kubernetes



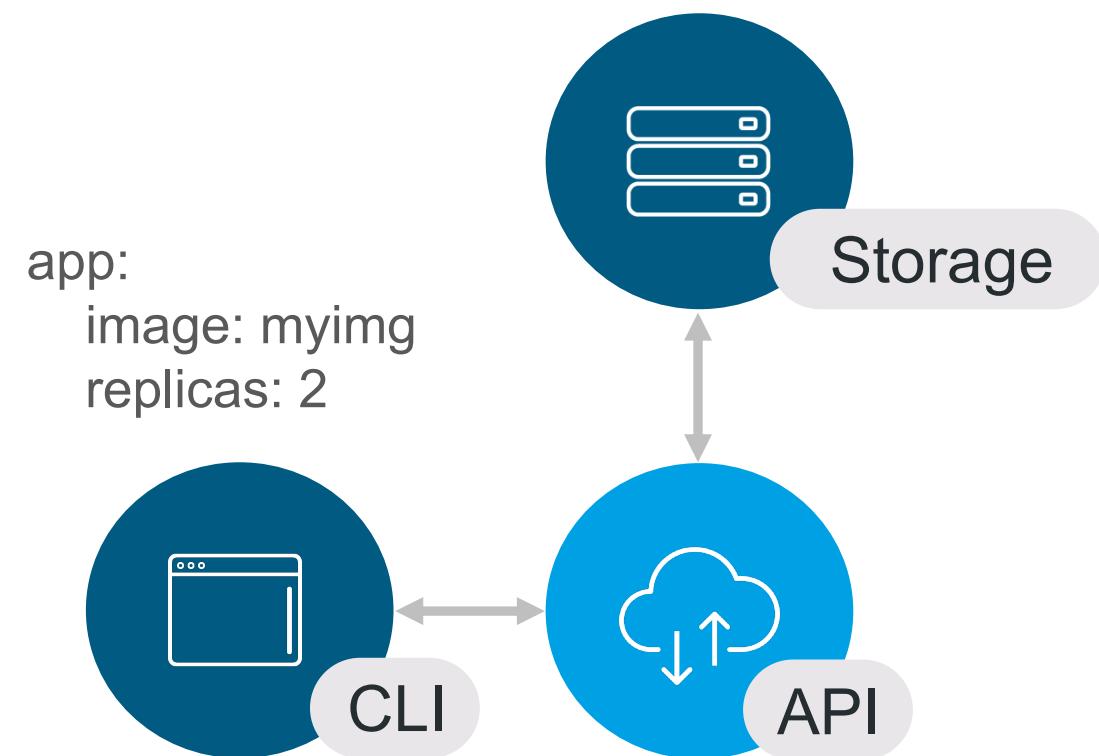
Преимущества Kubernetes



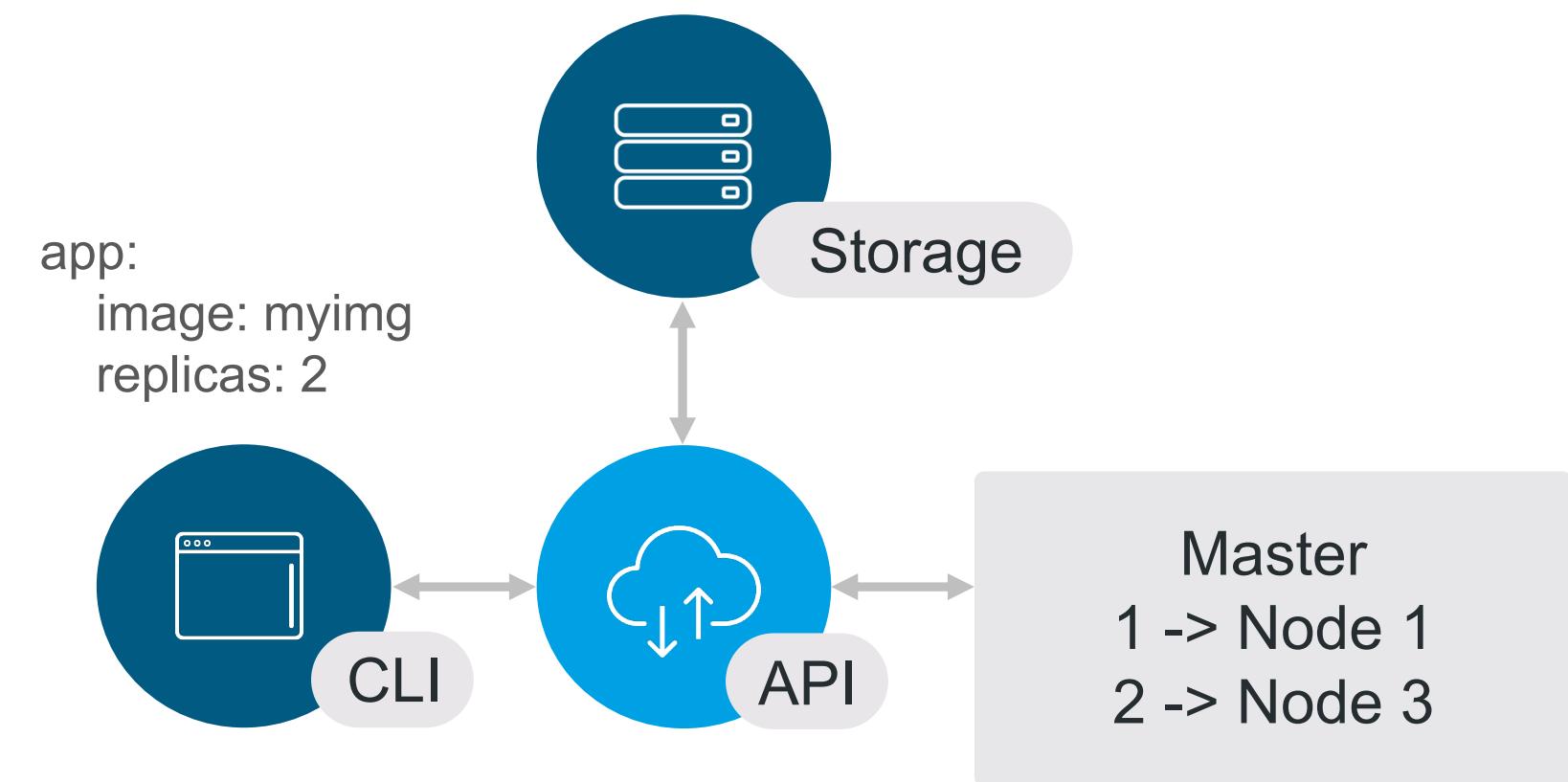
Kubernetes workflow



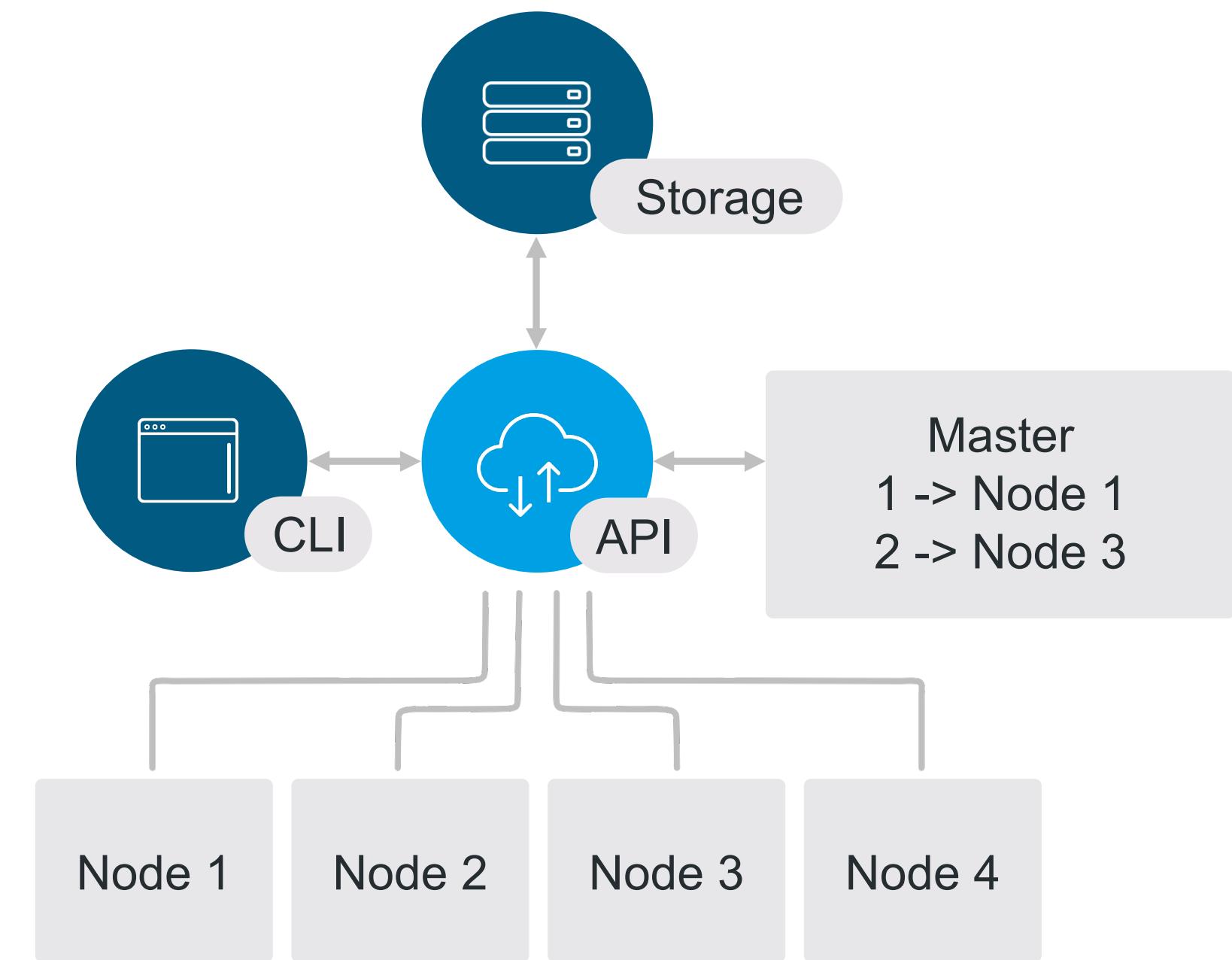
Kubernetes workflow



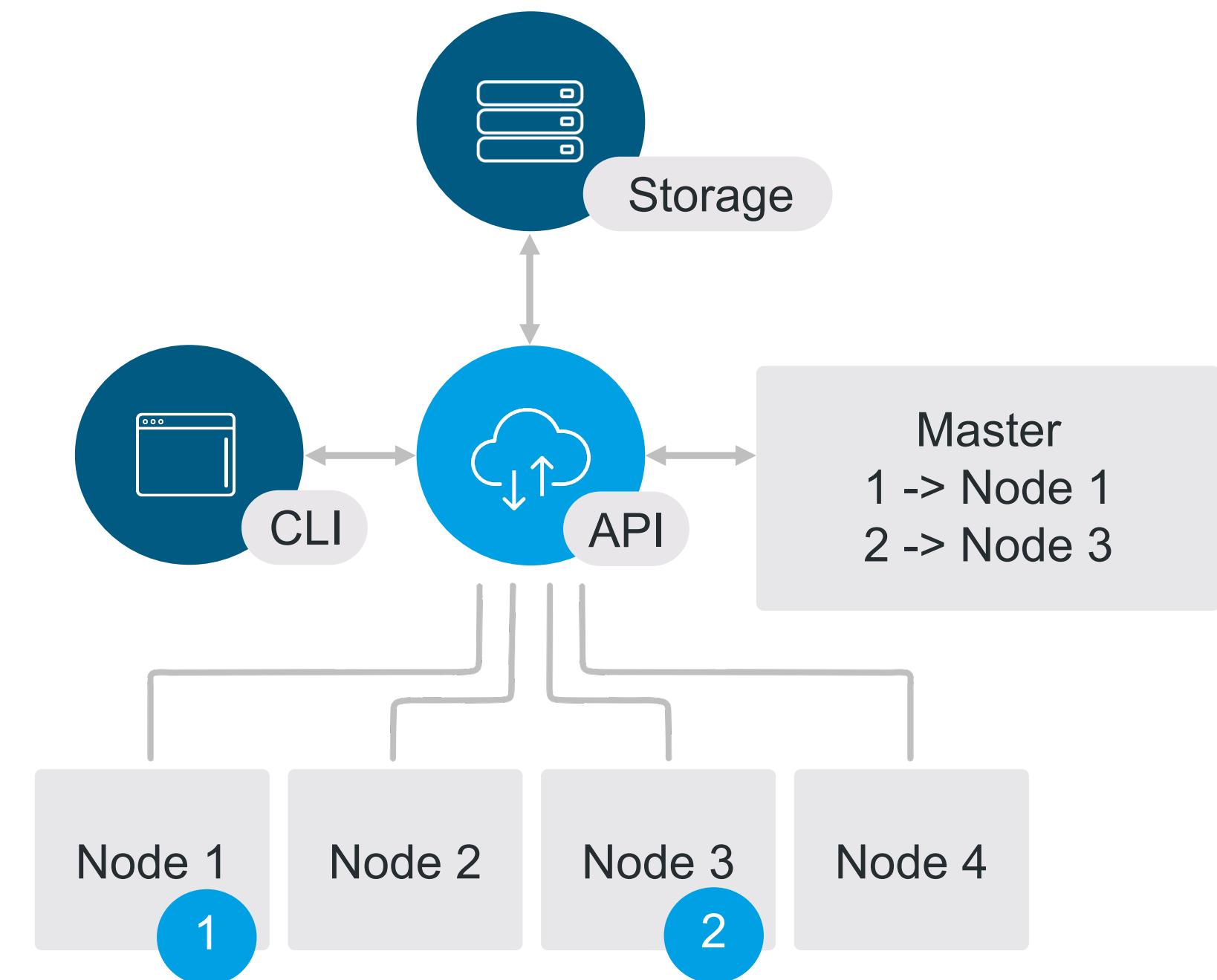
Kubernetes workflow



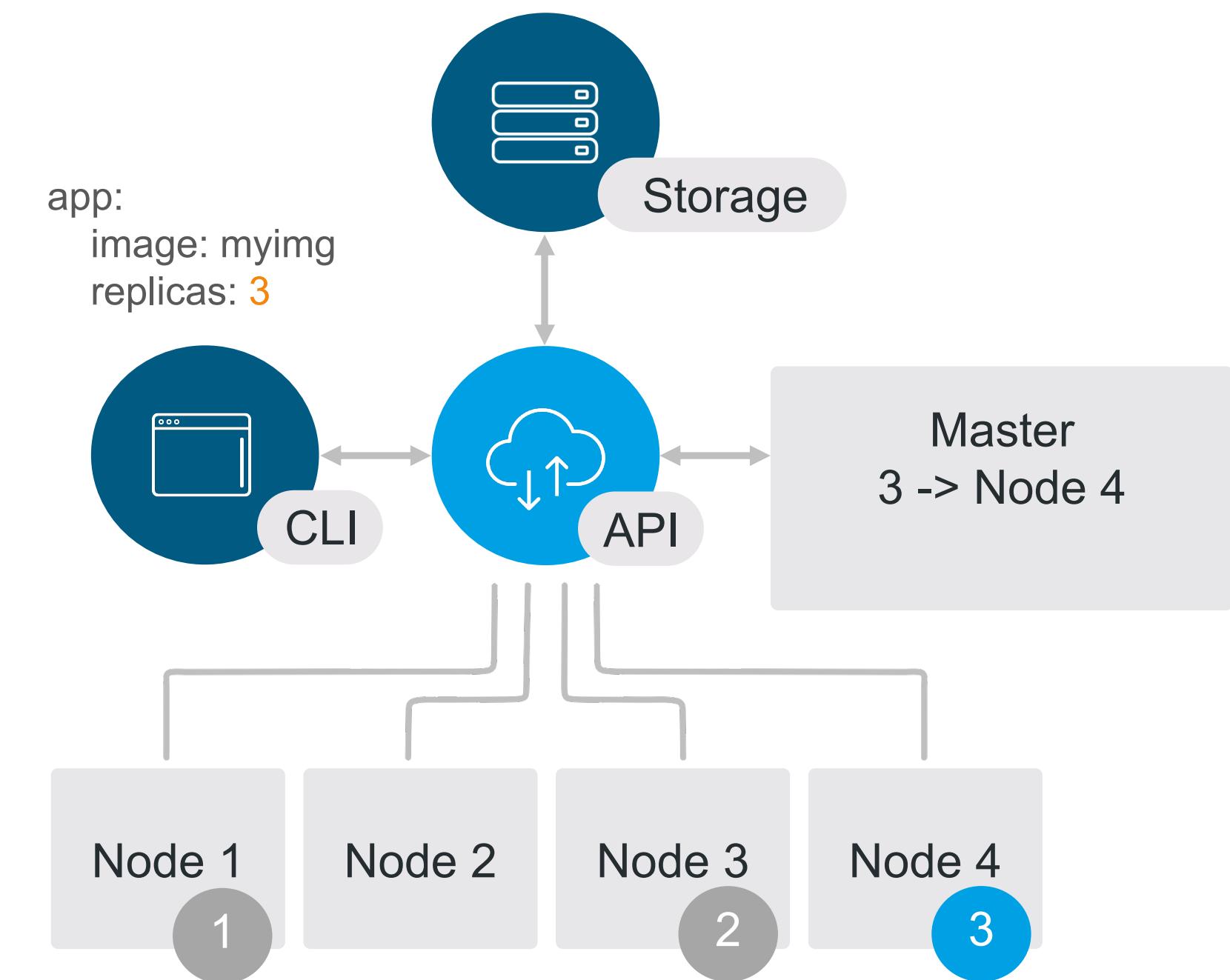
Kubernetes workflow



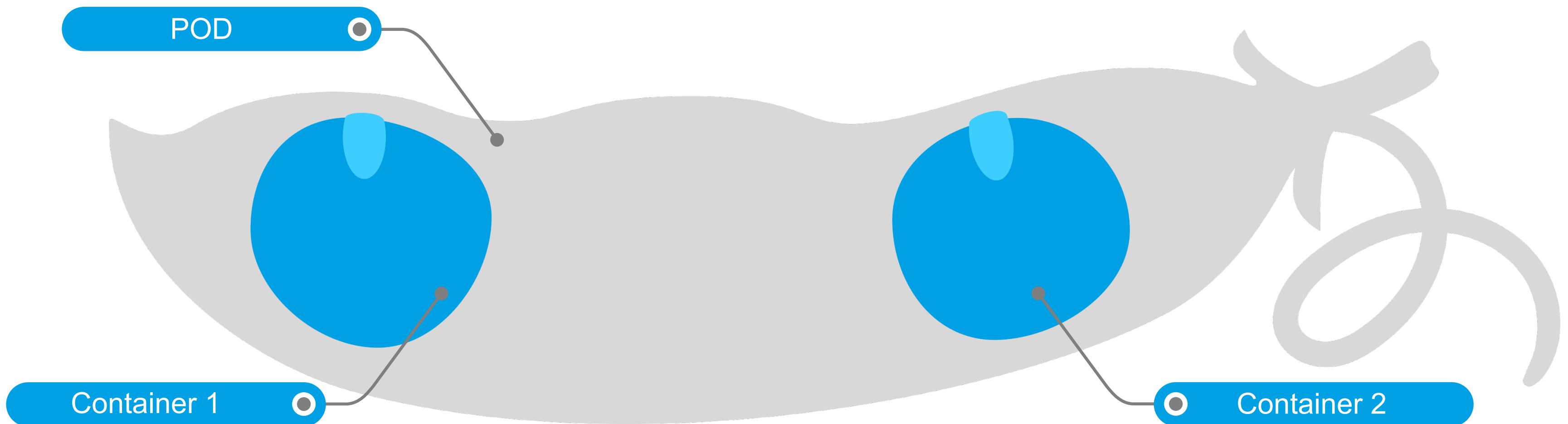
Kubernetes workflow



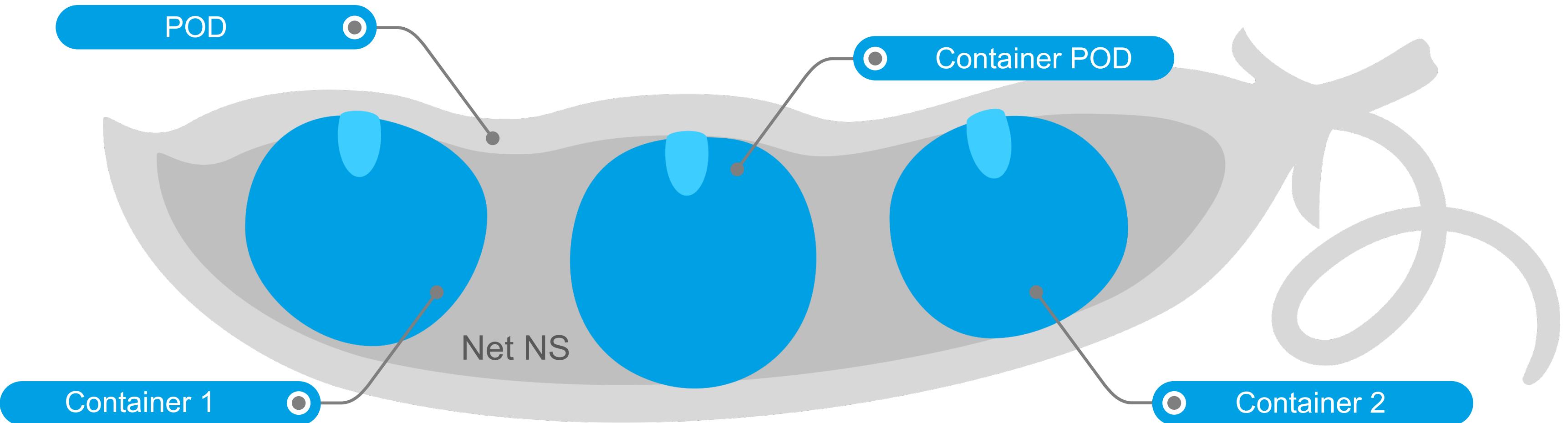
Kubernetes workflow



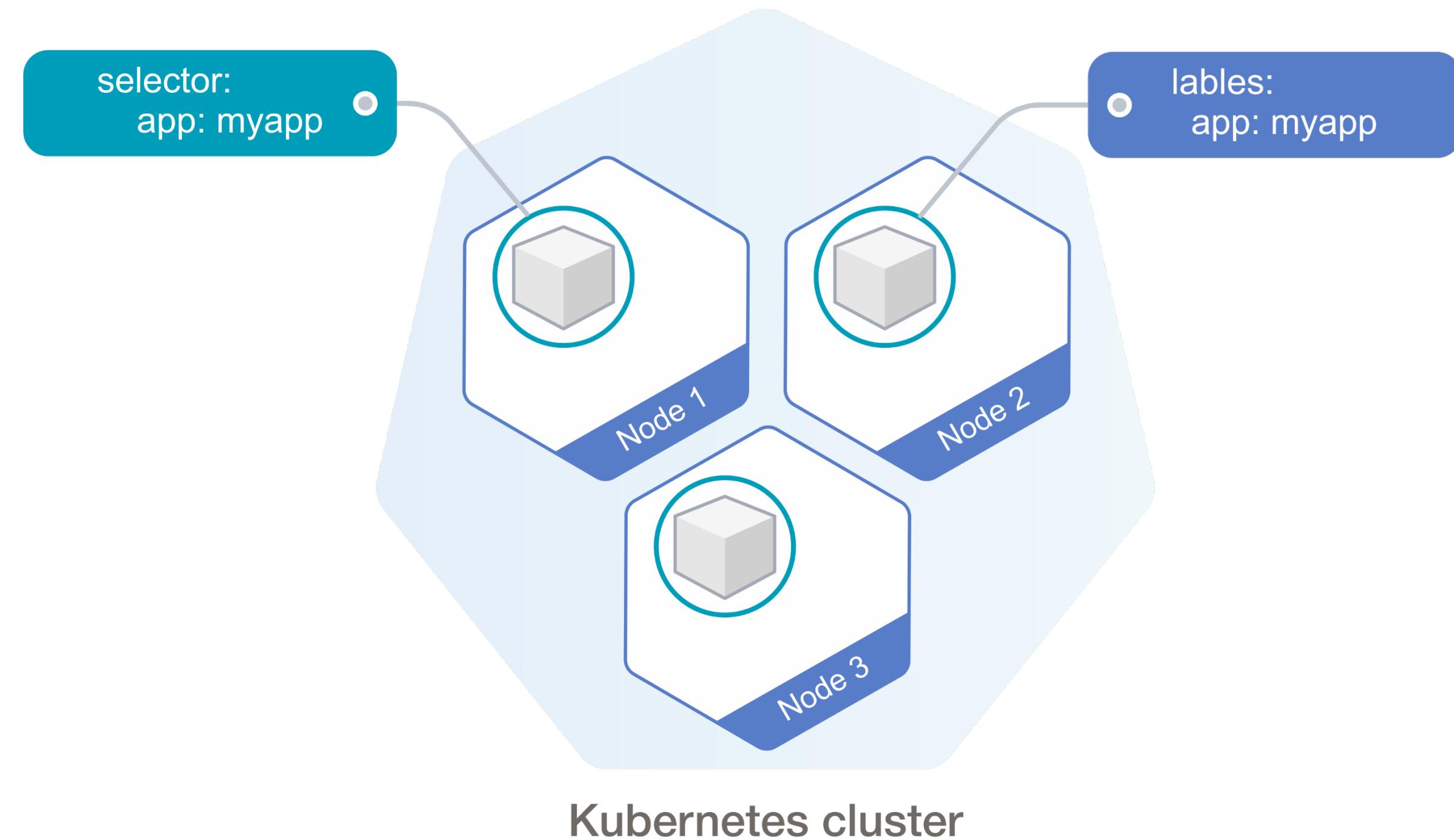
POD



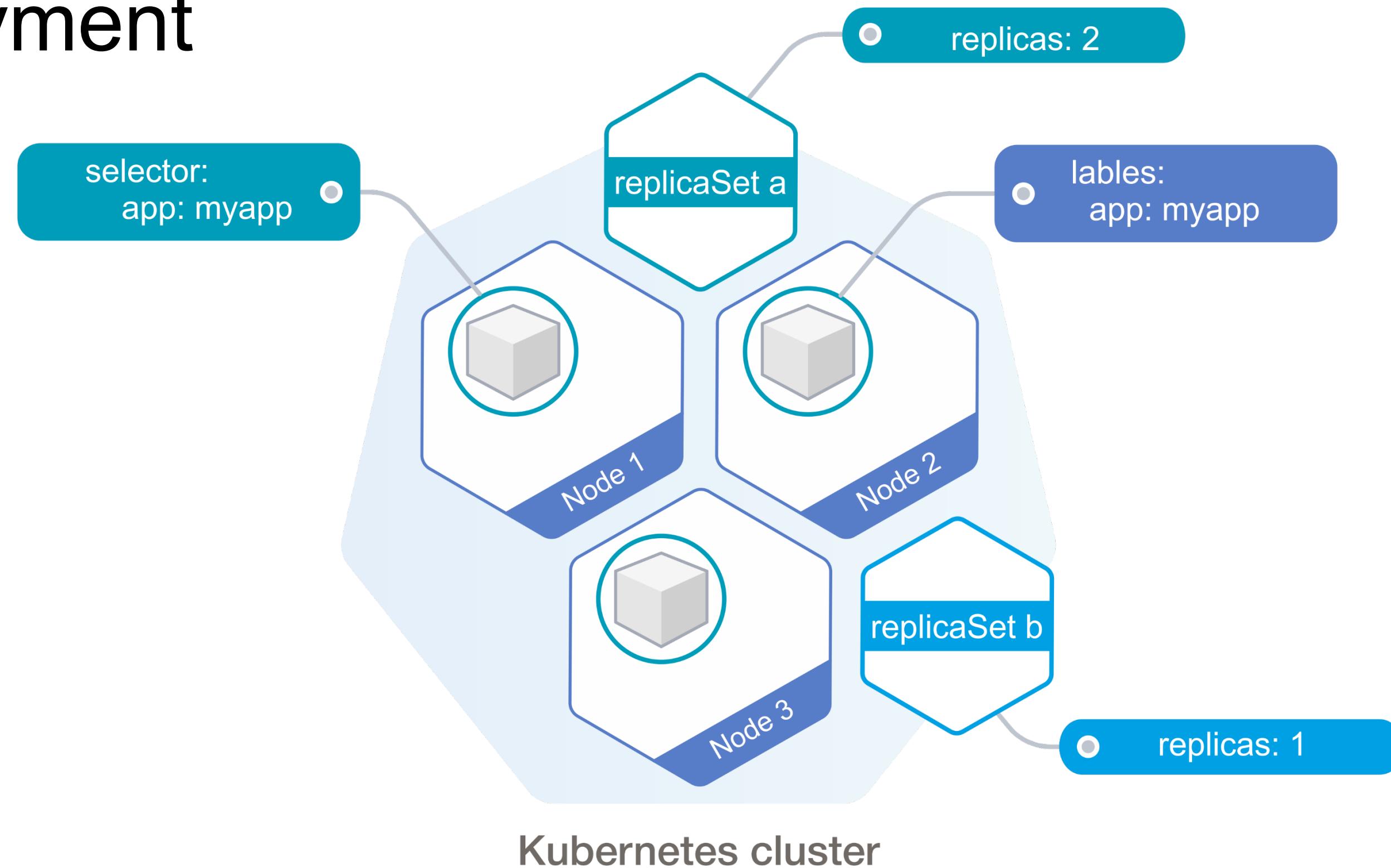
POD



ReplicaSet



Deployment



Deployment

1. Отредактируйте deployment из предыдущего задания таким образом, чтобы он запускался с **одной** репликой и rollingupdate проходил **без даунтайма**. То есть при обновлении образа всегда должен оставаться один рабочий pod.

Используйте поля maxSurge и maxUnavailable со значениями для деплоиментов с одной репликой.

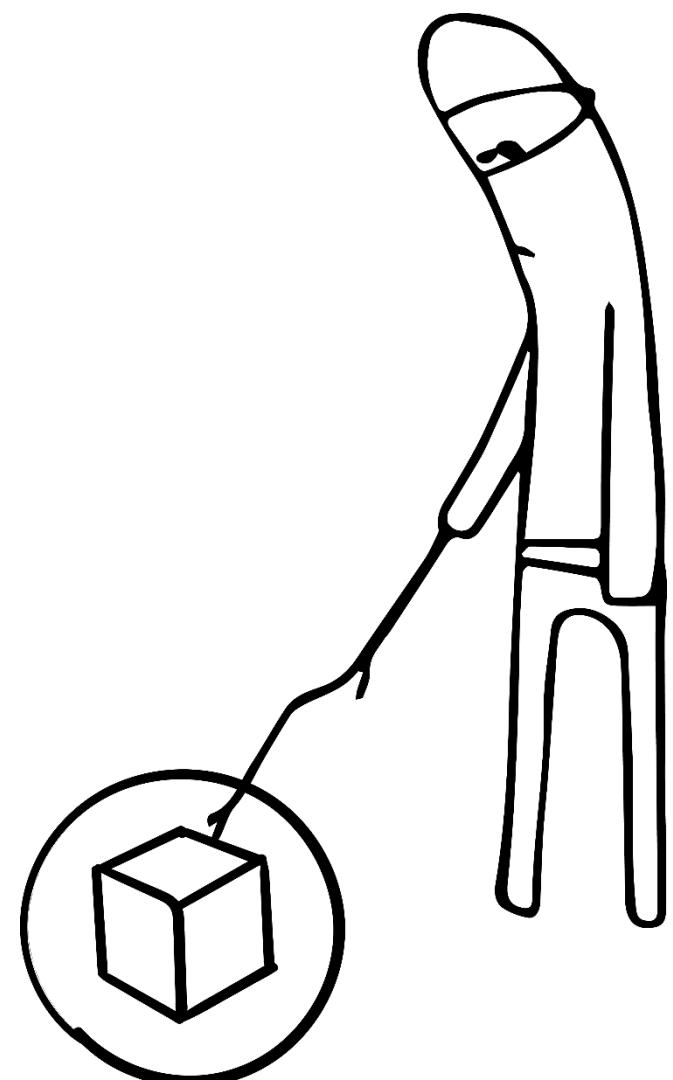
2. Обновите образ с **nginx:1.12** на **nginx:1.13**. Посмотрите в каком порядке запускаются и тушатся pods в момент обновления.

3. Выполните команду:

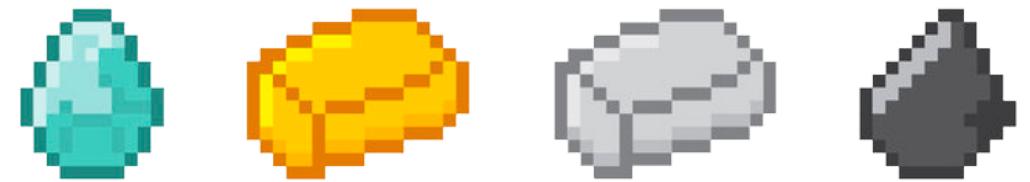
```
$ kubectl get deployment my-deployment -o custom-columns='NAME:.metadata.name,MAXSURGE:.spec.strategy.rollingUpdate.maxSurge,MAXUNAVAILABLE:.spec.strategy.rollingUpdate.maxUnavailable'
```

Probes

- Liveness Probe
 - Контроль за состоянием приложения во время его **жизни**
 - Исполняется постоянно
- Readiness Probe
 - Проверяет, **готово ли** приложение принимать трафик
 - В случае неудачного выполнения, приложение убирается из балансировки
 - Исполняется постоянно

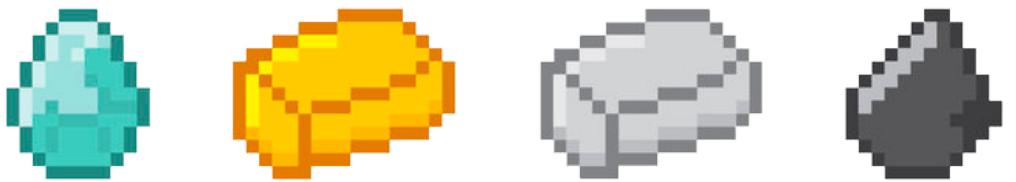


Resources



- **Limits**
 - Количество ресурсов, которые POD **может использовать**
 - Верхняя граница
- **Requests**
 - Количество ресурсов, которые **резервируются** для PODa **на ноде**
 - Не делятся с другими PODами на ноде

Resources



1. Отредактируйте deployment из предыдущего задания таким образом, чтобы он перестал влезать на ноду.

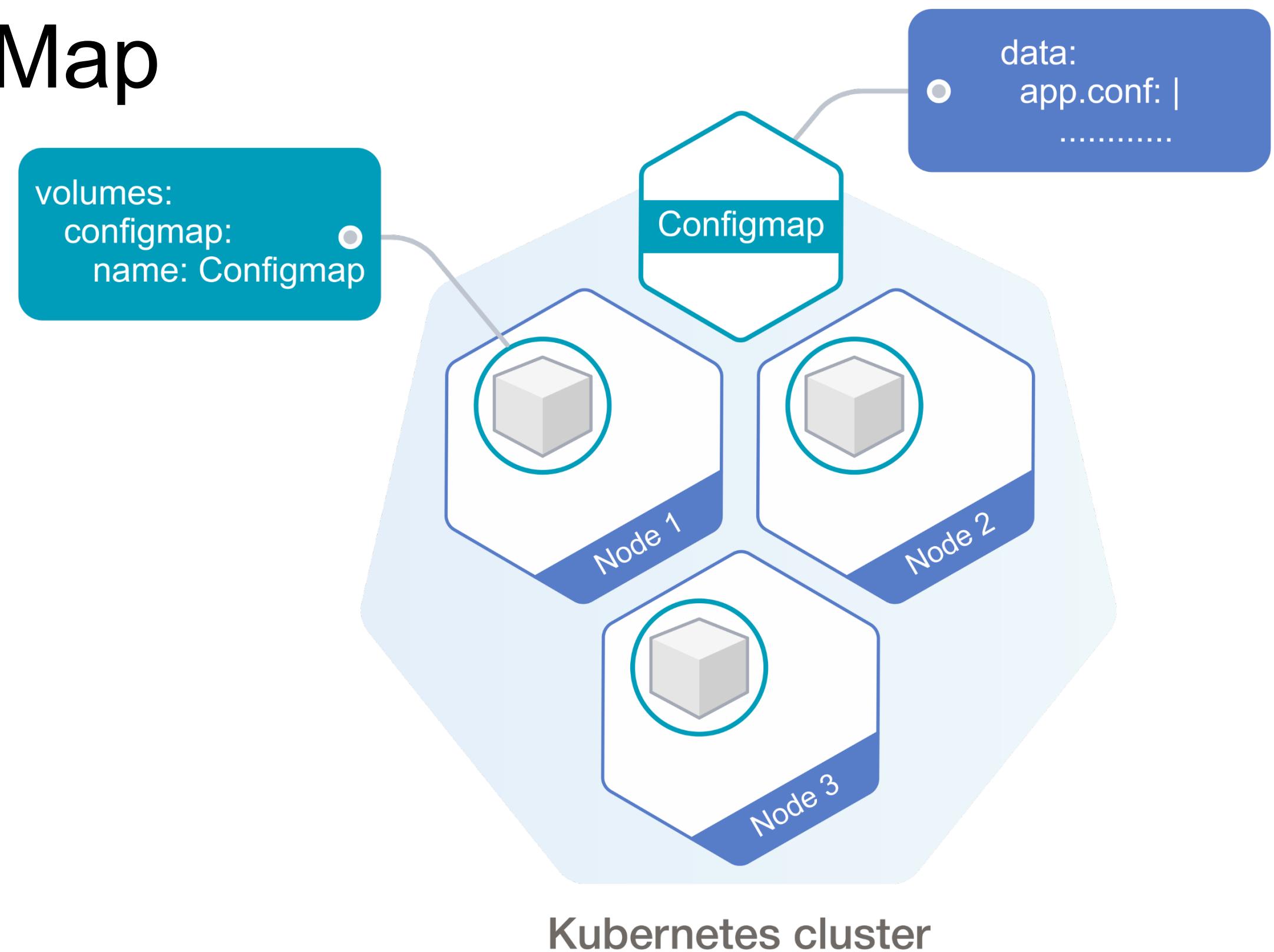
Например установите cpu requests = 10

Обратите внимание, что requests не может быть больше limits.

2. Посмотрите в описание (describe) poda в состоянии Pending.

Вас должны интересовать Events, объясняющие почему pod не может запуститься.

ConfigMap

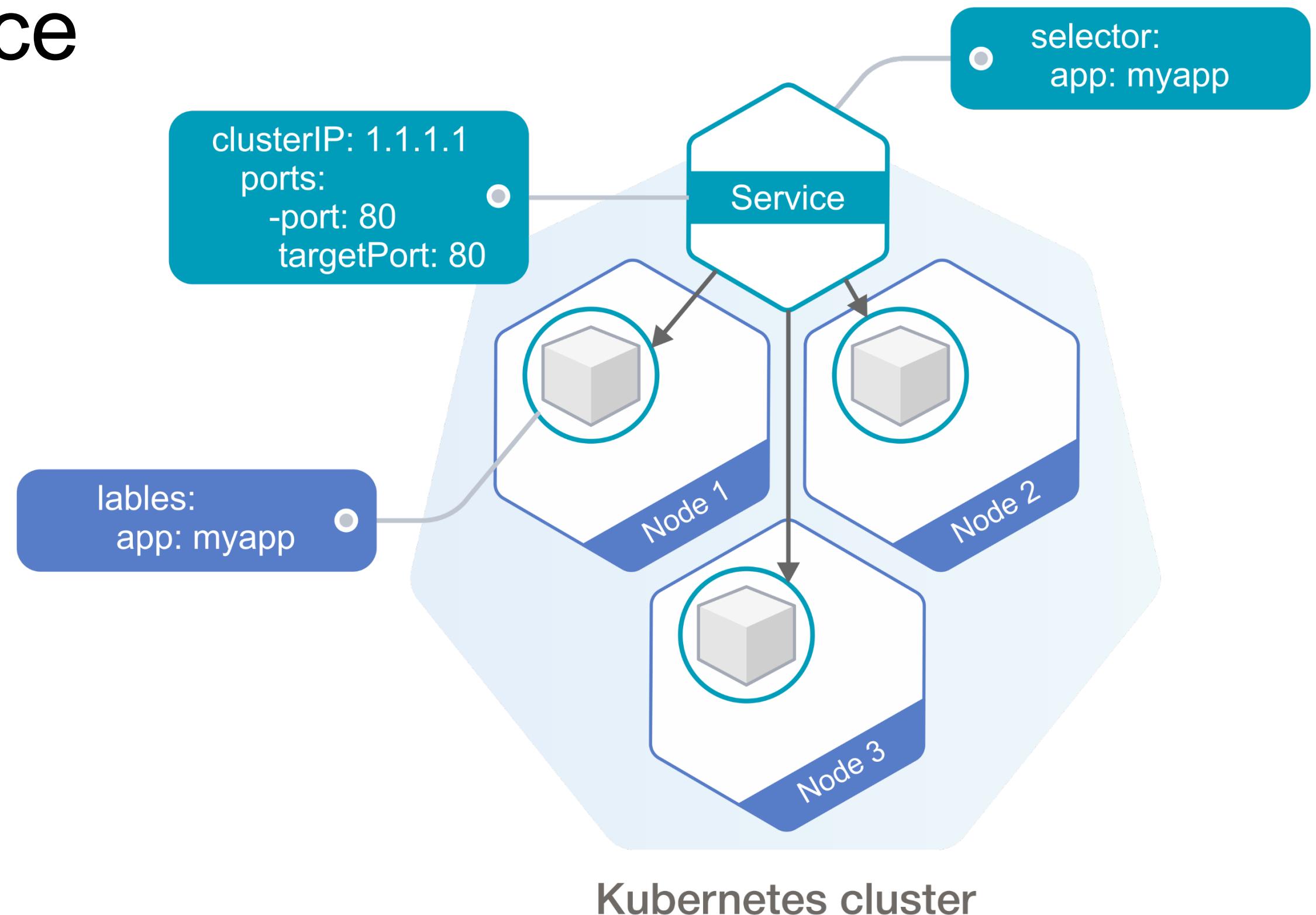


Secret

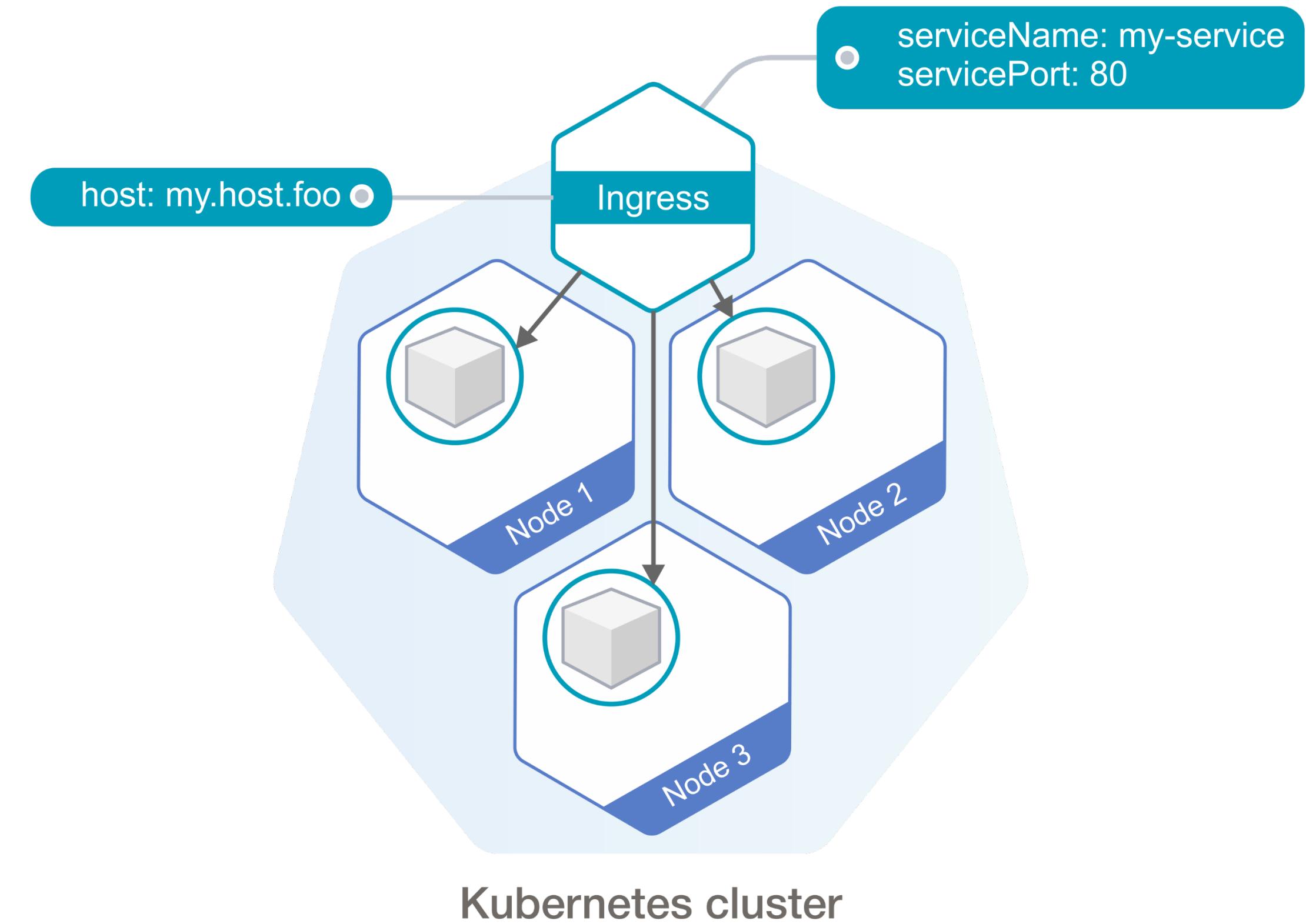
- Пароли / токены для приложений
- Данные авторизации в Registry
- TLS сертификаты для Ingress

CONFIDENTIAL

Service



Ingress



Ingress

1. Создайте и запустите deployment с именем my-app-1 из образа nginx 1.12 и deployment из образа nginx:1.13 с именем my-app-2. В оба деплоймента замонтируйте конфигурацию из configтаров по аналогии с предыдущими примерами. В одной конфигурации nginx должен отдавать "I am 1.12", в другой "I am 1.13"
2. Создайте services для обоих деплойментов с именами my-app-1 и my-app-2 соответственно.
3. Создайте ingress для обоих сервисов с именем хоста student<номер своего логина>.k8s.slurm.io таким образом, чтобы запросы на /1 приходили на my-app-1, а запросы на /2 приходили на my-app-2.

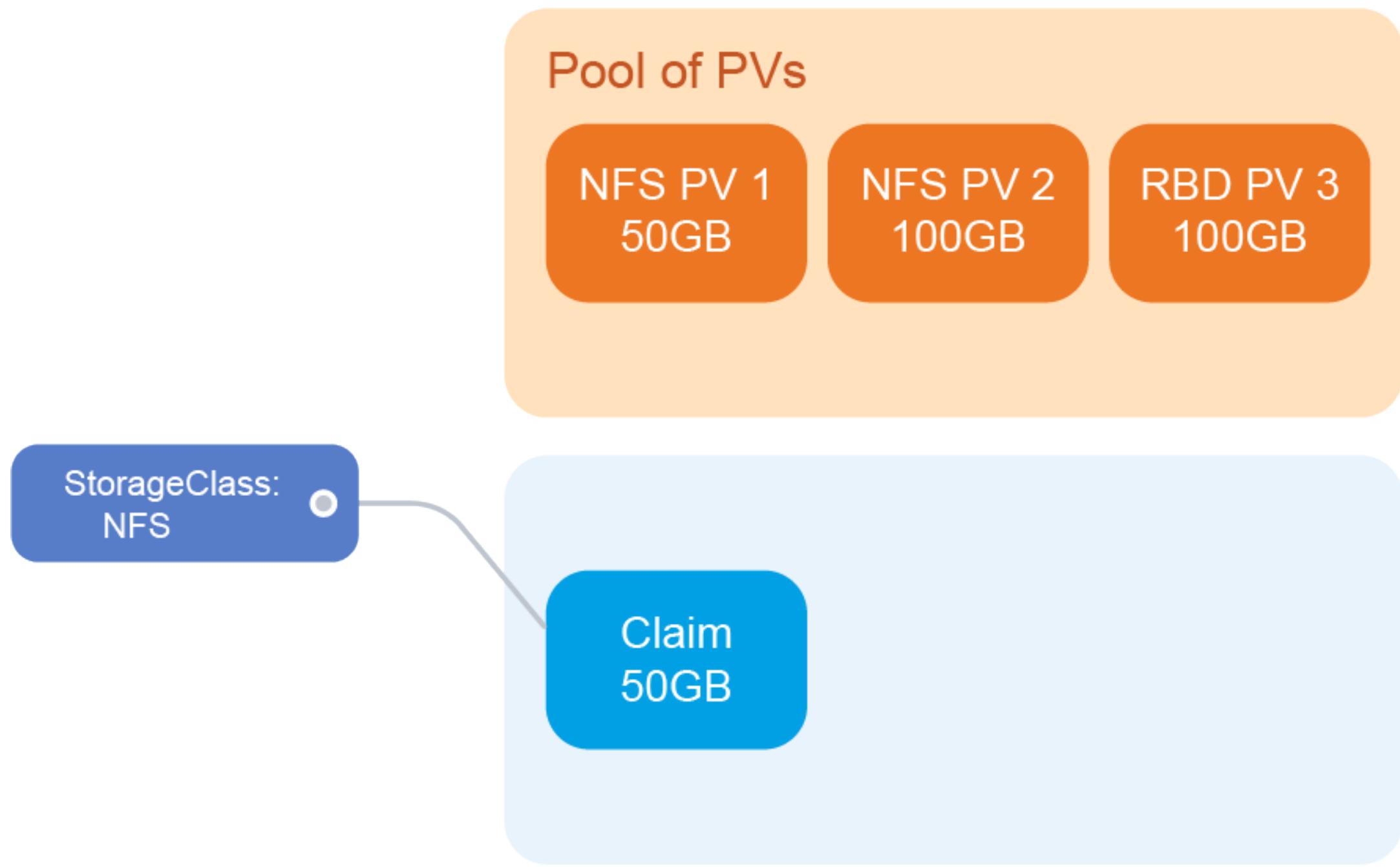
Для указания пути в ingressе используется поле path на одном уровне с backend .

Например:

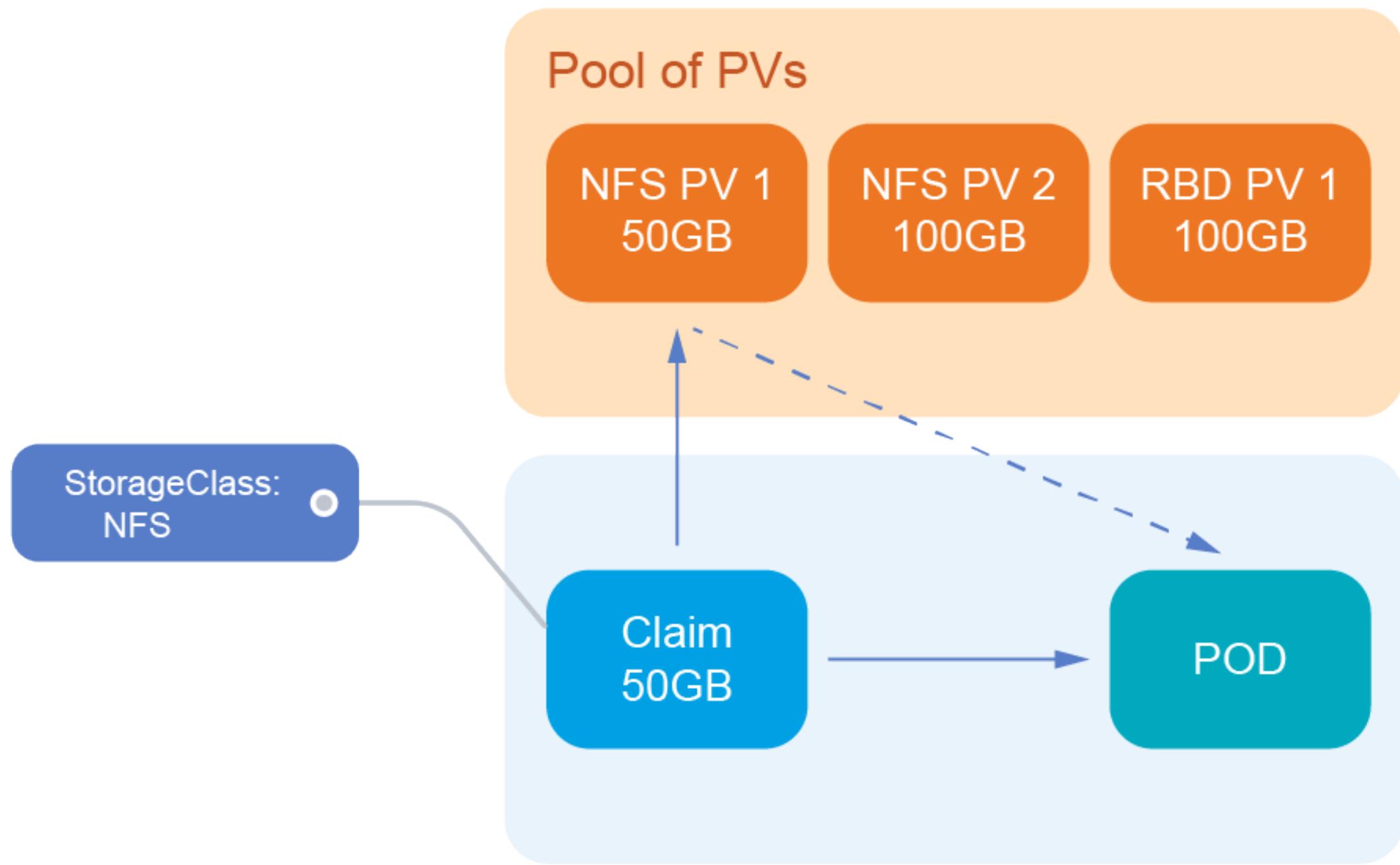
```
paths:  
- path: /testpath  
  backend:  
    serviceName: my-service  
    servicePort: 80
```

4. Проверьте с помощью **curl**, что правила в ingressе работают.

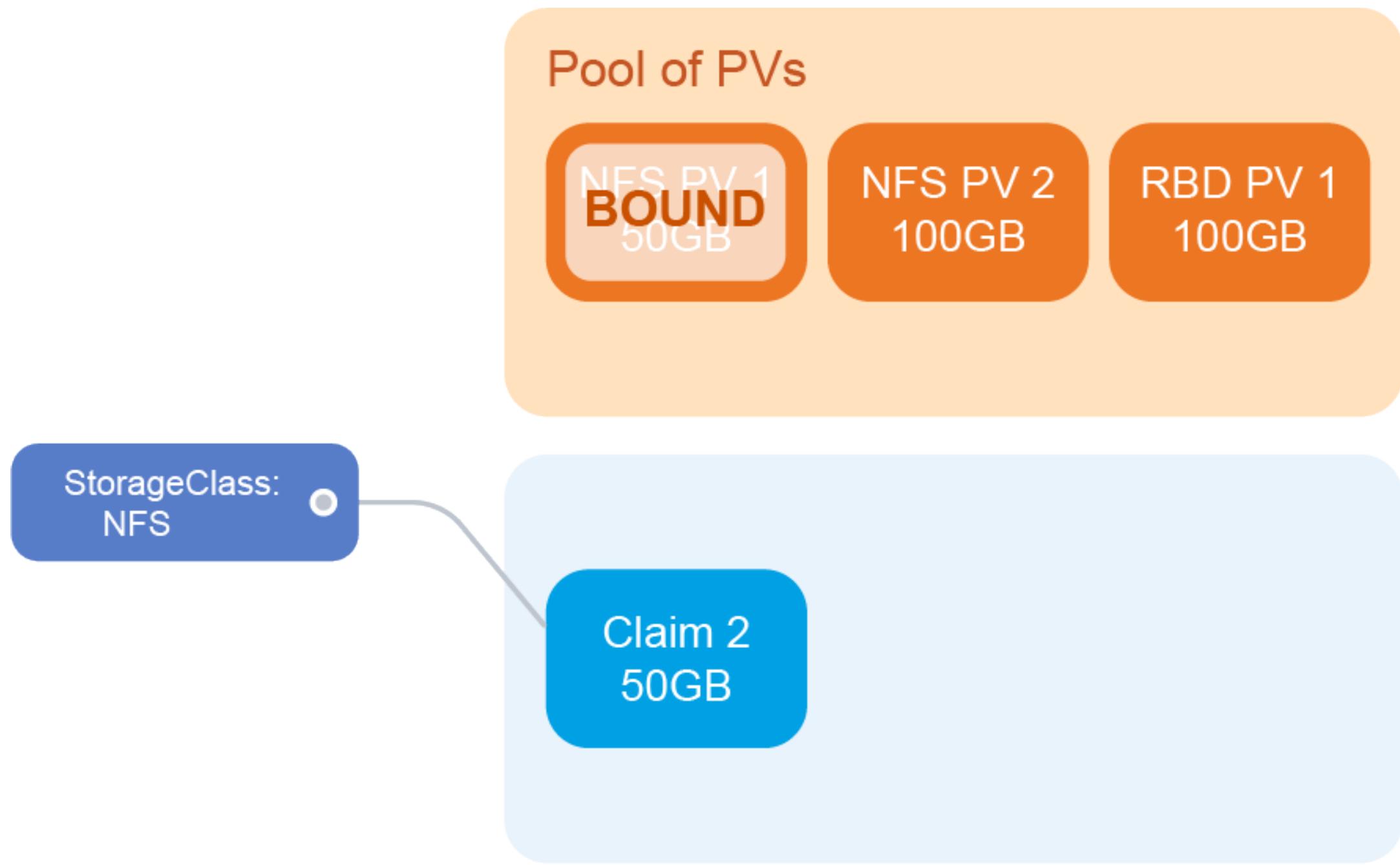
PV / PVC



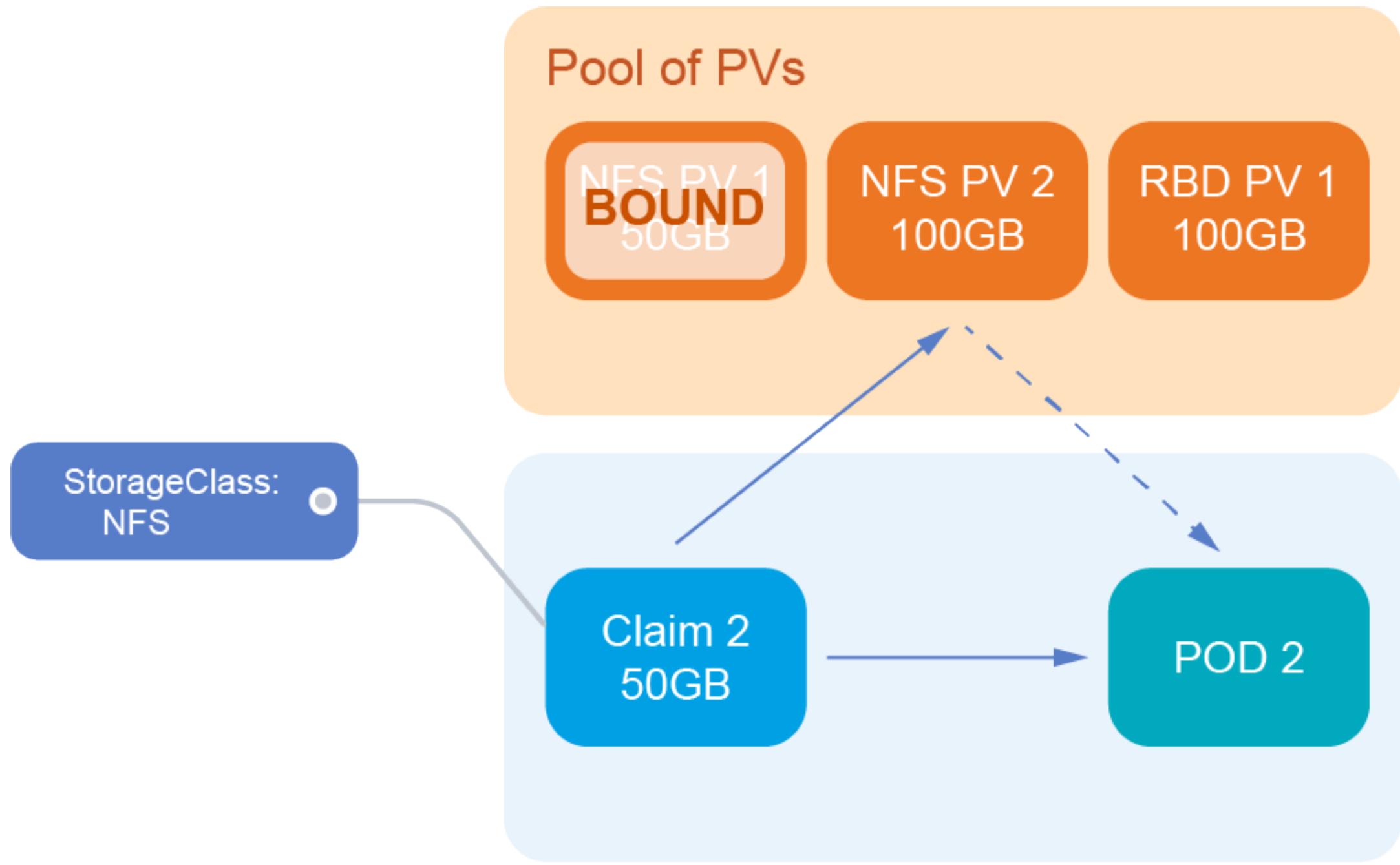
PV / PVC



PV / PVC

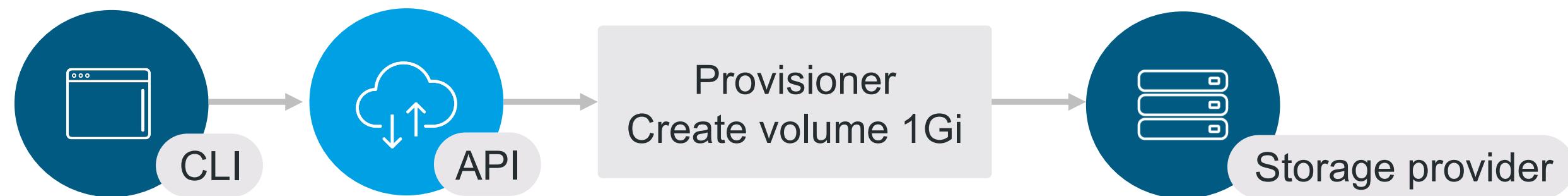


PV / PVC



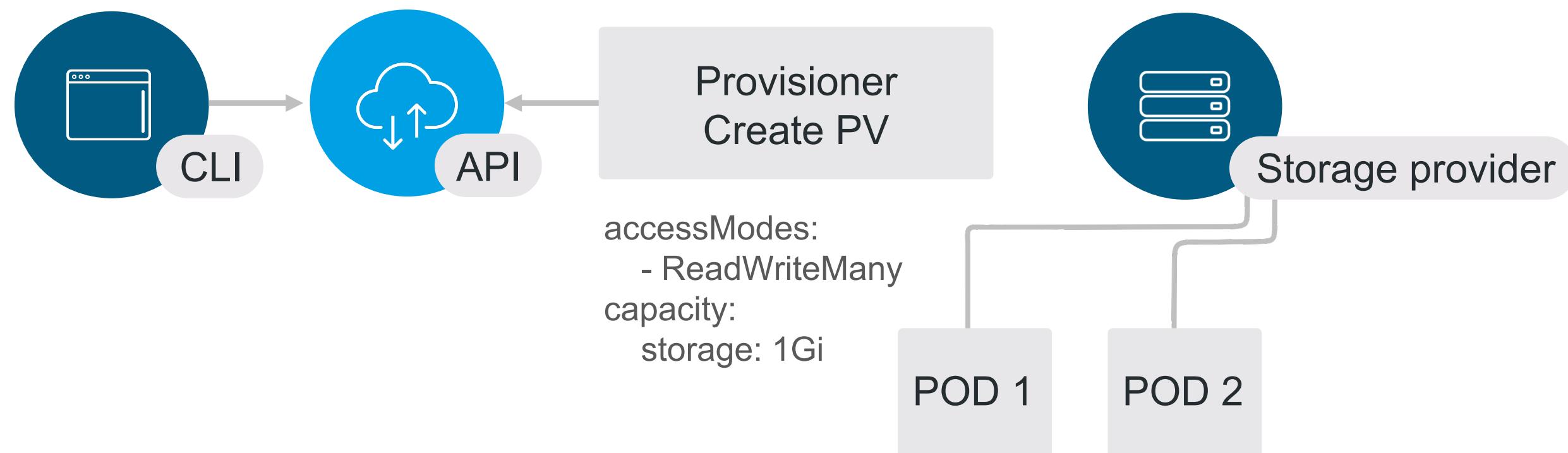
PV Provisioners

```
accessModes:  
  - ReadWriteMany  
resources:  
  requests:  
    storage: 1Gi
```

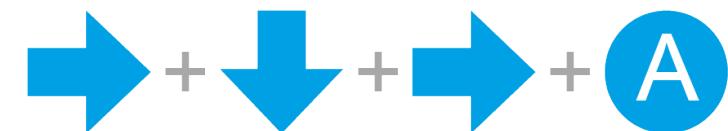


PV Provisioners

```
accessModes:  
  - ReadWriteMany  
resources:  
  requests:  
    storage: 1Gi
```



Kubectl cheat sheet



Create

Создание объекта из файла

```
kubectl create -f file.yaml
```

Создание или обновление объекта из файла

```
kubectl apply -f file.yaml
```

Создание деплоймента из cli

```
kubectl run --image image_name:tag name [command]
```

Interact

Выполнение команды внутри пода

```
kubectl exec -t -i pod_name command
```

Просмотр логов

```
kubectl logs pod_name
```

List

Получения списка объектов

```
kubectl get [pod|replicaset|deployment|...]
```

Полезные параметры для команды get:

- o wide – Расширенный вывод + IP подов и имена нод
- o yaml – Получение полного описания объекта в yaml
- n ns_name – Получение объектов в конкретном нэймспэйсе

Получение описания объекта и событий по нему

```
kubectl describe [pod|replicaset|deployment|...]
```

Clean up

Удаление объекта

```
kubectl delete [pod|replicaset|deployment|...] object_name
```

Удаление всех объектов

```
kubectl delete [pod|replicaset|deployment|...] --all
```

Удаление всех объектов (не включая ингрессы и конфигмапы)

```
kubectl delete all --all
```

Удаление объектов объявленных в файле

```
kubectl delete -f file.yaml
```

Update

Изменение объекта на лету

```
kubectl edit [pod|replicaset|deployment|...] object_name
```

Обновление имаджа

```
kubectl set image [deployment|...] container=image:tag
```

Просмотр файла в контейнере

```
docker exec container_name cat /etc/config.conf
```

Get Help

Получение описания команды, примеров использования и опций

```
kubectl [command] --help
```

Получение списка доступных команд

```
kubectl help
```

Получение описания объекта определенного типа

```
kubectl explain [pod|replicaset|deployment|...]
```

СЛЁРМ

+



Southbridge

slurm.io