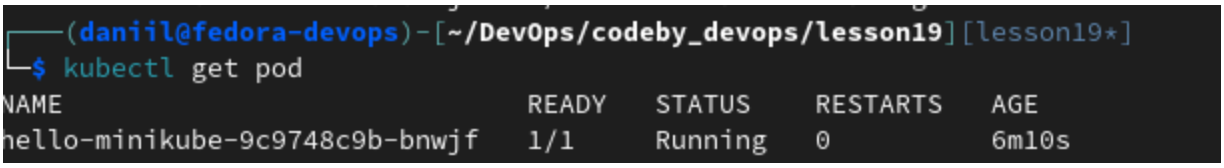
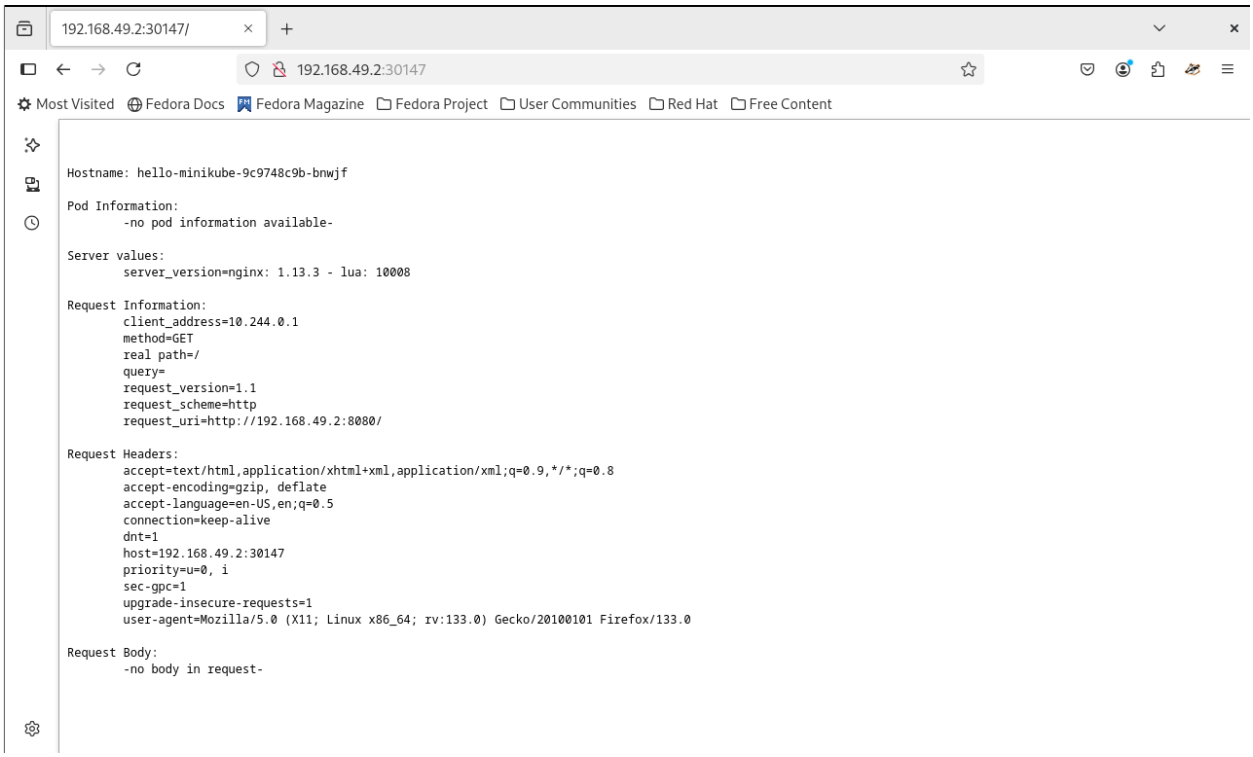
# Отчет

## Minikube

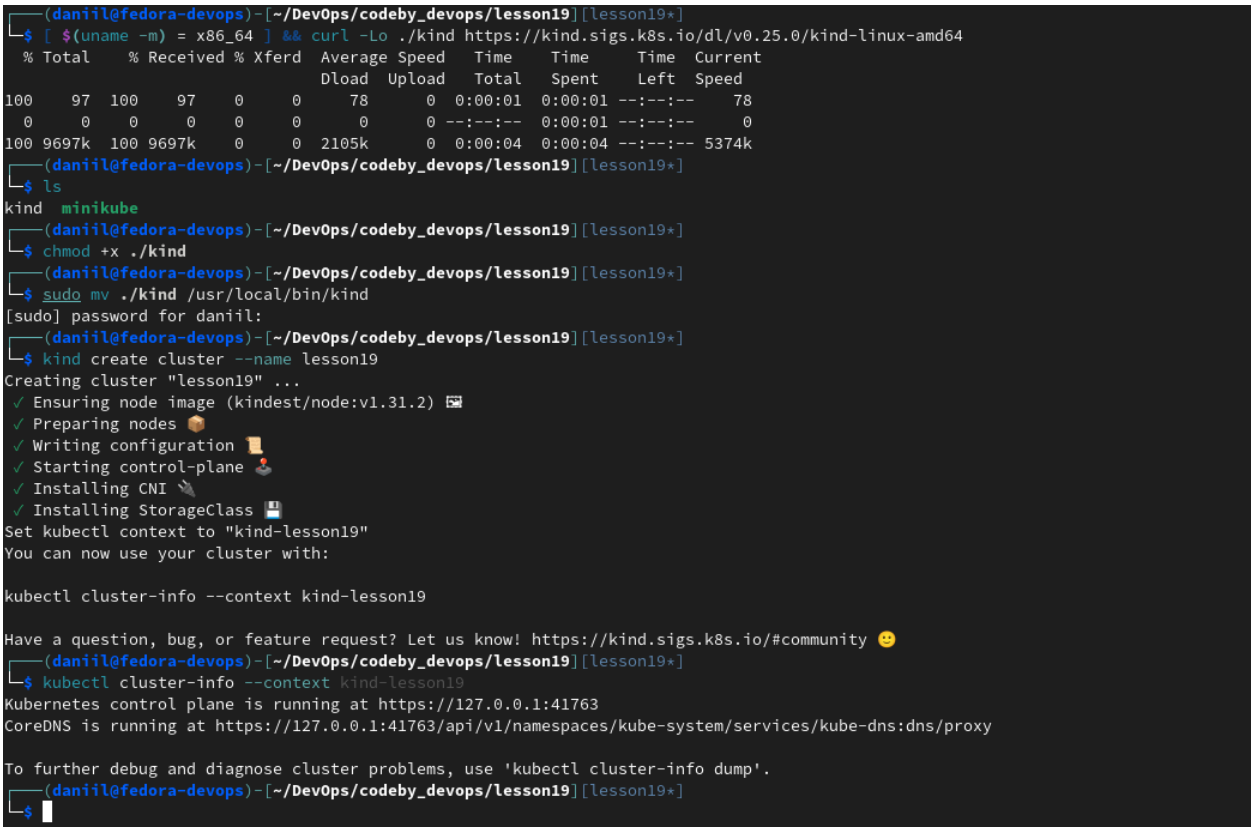
```
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ minikube start
😄  minikube v1.34.0 on Fedora 40
E1215 03:46:09.837665   68770 start.go:812] api.Load failed for minikube: filestore "minikube": Docker machine "minikube" does not exist. Use "docker-machine
ls" to list machines. Use "docker-machine create" to add a new one.
✨  Using the docker driver based on existing profile
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.45 ...
    > index.docker.io/kicbase/sta...:  487.90 MiB / 487.90 MiB  100.00% 29.25 M
❗  minikube was unable to download gcr.io/k8s-minikube/kicbase:v0.0.45, but successfully downloaded docker.io/kicbase/stable:v0.0.45 as a fallback image
🔥  Creating docker container (CPUs=2, Memory=2200MB) ...
🐳  Preparing Kubernetes v1.31.0 on Docker 27.2.0 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
🔗  Configuring bridge CNI (Container Networking Interface) ...
🔎  Verifying Kubernetes components...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
🌟  Enabled addons: default-storageclass, storage-provisioner
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ kubectl cluster-info
Kubernetes control plane is running at https://192.168.49.2:8443
CoreDNS is running at https://192.168.49.2:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ kubectl create deployment hello-minikube --image=registry.k8s.io/echoserver:1.10
deployment.apps/hello-minikube created
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ kubectl expose deployment hello-minikube --type=NodePort --port=8080
service/hello-minikube exposed
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ kubectl get pod
NAME                              READY   STATUS              RESTARTS   AGE
hello-minikube-9c9748c9b-bnwjf   0/1     ContainerCreating   0          27s
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$
```




```
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ kubectl delete services hello-minikube
service "hello-minikube" deleted
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ kubectl delete deployment hello-minikube
deployment.apps "hello-minikube" deleted
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ minikube stop
✋  Stopping node "minikube"  ...
🛑  Powering off "minikube" via SSH ...
🛑  1 node stopped.
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$ minikube delete
🔥  Deleting "minikube" in docker ...
🔥  Deleting container "minikube" ...
🔥  Removing /home/daniil/.minikube/machines/minikube ...
💀  Removed all traces of the "minikube" cluster.
┌──(daniil@fedora-devops)-[~/DevOps/codeby_devops/lesson19][lesson19*]
└─$
```

# Kind




# Развертывание pod

## Императивная команда

1. Создаём namespace: `kubectl create namespace lesson19`

   Проверяем: `kubectl get namespaces`

```
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl get namespaces
NAME                 STATUS   AGE
default              Active   16m
kube-node-lease      Active   16m
kube-public          Active   16m
kube-system          Active   16m
local-path-storage   Active   16m
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl config get-contexts

CURRENT   NAME            CLUSTER         AUTHINFO        NAMESPACE
*         kind-lesson19   kind-lesson19   kind-lesson19
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl config current-context
kind-lesson19
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl create namespace codeby
namespace/codeby created
```

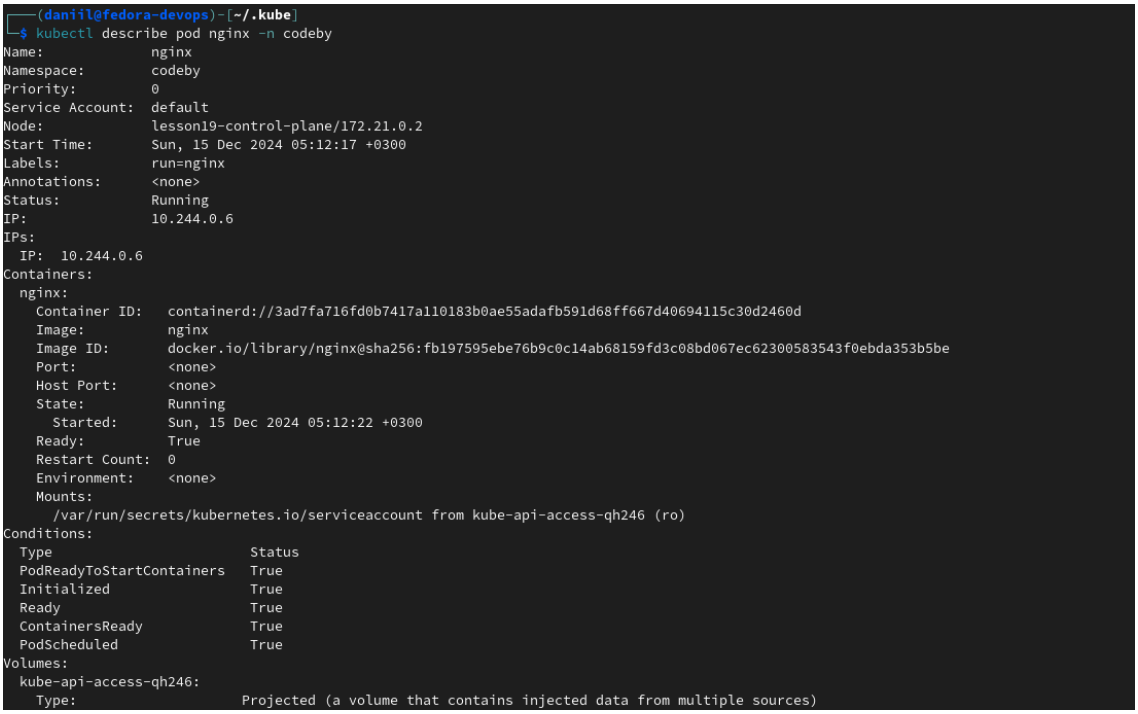
2. Разворачиваем Pod `kubectl run nginx --image nginx --namespace codeby`

```
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl run nginx --image nginx --namespace codeby
pod/nginx created
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl get pods
No resources found in default namespace.
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl get pods -n codeby
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          28s
┌──(daniil@fedora-devops)-[~/.kube]
└─$
```

3. Проверяем статус Pod
   a. “Краткая информация `kubectl get pods (-o wide) -n lesson19`

```
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl get pods -n codeby
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          28s
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl get pods -o wide -n codeby
NAME    READY   STATUS    RESTARTS   AGE    IP           NODE                    NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          119s   10.244.0.6   lesson19-control-plane   <none>           <none>
```

b. Также можно подробней `kubectl describe pod nginx -n codeby`



```
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl describe pod nginx -n codeby
Name:             nginx
Namespace:        codeby
Priority:         0
Service Account:  default
Node:             lesson19-control-plane/172.21.0.2
Start Time:       Sun, 15 Dec 2024 05:12:17 +0300
Labels:           run=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.6
IPs:
  IP:  10.244.0.6
Containers:
  nginx:
    Container ID:   containerd://3ad7fa716fd0b7417a110183b0ae55adafb591d68ff667d40694115c30d2460d
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:fb197595ebe76b9c0c14ab68159fd3c08bd067ec62300583543f0ebda353b5be
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sun, 15 Dec 2024 05:12:22 +0300
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-qh246 (ro)
Conditions:
  Type                        Status
  PodReadyToStartContainers   True
  Initialized                 True
  Ready                       True
  ContainersReady             True
  PodScheduled                True
Volumes:
  kube-api-access-qh246:
    Type:                    Projected (a volume that contains injected data from multiple sources)
```

c. Либо вывести в yaml формате `kubectl get pod nginx -o yaml -n codeby`

```
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl get pod nginx -o yaml -n codeby
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2024-12-15T02:12:17Z"
  labels:
    run: nginx
  name: nginx
  namespace: codeby
  resourceVersion: "4215"
  uid: e8a8f834-220a-4ebc-82f8-e3fd844dfbb7
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: nginx
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-qh246
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: lesson19-control-plane
  preemptionPolicy: PreemptLowerPriority
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
  tolerations:
  - effect: NoExecute
    key: node.kubernetes.io/not-ready
    operator: Exists
    tolerationSeconds: 300
  - effect: NoExecute
    key: node.kubernetes.io/unreachable
```

4. Удаляем Pod `kubectl delete pod nginx -n codeby`

```
┌──(daniil@fedora-devops)-[~/.kube]
└─$ kubectl delete pod nginx -n codeby
pod "nginx" deleted
```

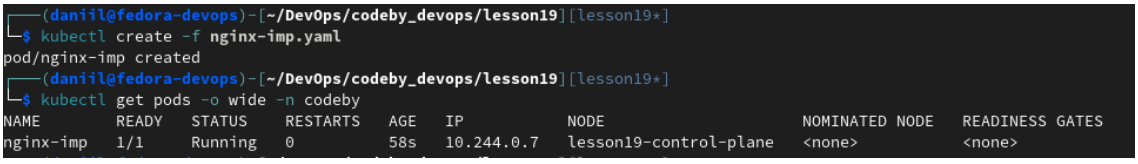
# Императивная конфигурация

При использовании императивной конфигурации объект описывается в YAML-файле. Команда `kubectl create` применяется для выполнения заданного действия (создание, удаление, замена и т. д.).

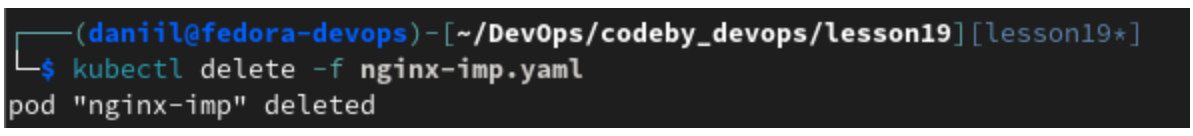
1. Создаём файл **nginx-imp.yaml:**

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-imp
  namespace: codeby
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

2. Создаём Pod `kubectl create -f nginx-imp.yaml`:
   a. Здесь команда напрямую создаёт объект, используя файл конфигурации.
   b. После создания объект существует в кластере.




3. Удаляем Pod `kubectl delete -f nginx-imp.yaml`:



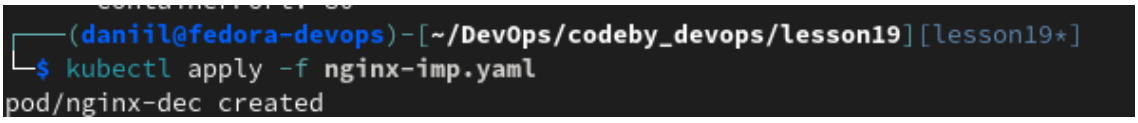
# Декларативная конфигурация

Декларативный подход работает с файлами YAML, но команды (`kubectl apply`) не привязаны к конкретным действиям. Kubernetes сам определяет, что нужно создать, обновить или удалить, чтобы достичь описанного состояния.

1. Используем файл **nginx-imp.yaml**, но немного поменяем его**:**

```
apiVersion: v1
kind: Pod
metadata:
    name: nginx-dec
    namespace: codeby
spec:
    containers:
    - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

2. Применяем декларативную конфигурацию `kubectl apply -f nginx-imp.yaml`:
    - Команда `apply` сохраняет описание объекта как «источник правды».
    - Kubernetes автоматически приводит объект к состоянию, описанному в YAML.



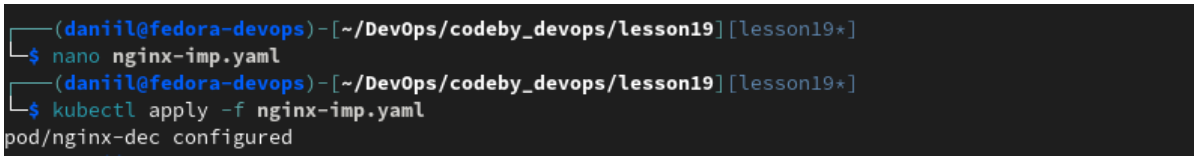

3. **Вносим изменения в конфигурацию:**

    Например, изменим базовый образ:

```
spec:
  containers:
```

```
    - name: nginx
      image: nginx:1.14.2
```

Применяем изменения `kubectl apply -f nginx-imp.yaml`:




4. Удаляем объект `kubectl delete -f nginx-imp.yaml`:

Небольшая теория для себя

## Сравнение подходов в контексте лабораторной

| Характеристика | Императивные команды | Императивная конфигурация | Декларативная конфигурация |
|---|---|---|---|
| **Где описываются объекты** | Только в команде | YAML-файл | YAML-файл |
| **Как выполняются действия** | Прямые команды | По команде `create`, `replace` | По команде `apply` |
| **Изменение объекта** | Требуется новая команда | Редактирование файла + `replace` | Редактирование файла + `apply` |
| **Журнал изменений** | Нет | Можно хранить файл в Git | Можно хранить файл в Git |
| **Работа с директориями** | Нет | Только по одному файлу | Да, можно применять к директориям |
| **Пример команды** | `kubectl run nginx` | `kubectl create -f file.yaml` | `kubectl apply -f file.yaml` |