9. Работа с Deployment

9.1. Создание Deployment через команду

1. Проверим список деплойментов:

```
kubectl get deploy  # Показывает текущие Deployment в кластере
```

2. Создадим деплой с одним подом:

```
kubectl create deployment first-deploy --image=k8sphp  # Создает Deployment с указанным образом контейнера
```

3. Проверим список деплойментов:

```
kubectl get deploy  # Проверяем, что Deployment появился в списке
```

4. Проверим статус подов:

```
kubectl get pods  # Смотрим, какие поды были созданы
```

5. Описание деплоя:

```
kubectl describe deployments first-deploy  # Подробная информация о Deployment
```

9.2. Масштабирование Deployment

1. Масштабируем деплой до 4 реплик:

```
kubectl scale deployment first-deploy --replicas=4  # Указываем нужное число реплик
```

2. Проверяем количество реплик:

```
kubectl get rs  # Просматриваем ReplicaSet, управляющий подами
```

3. Проверяем количество подов:

```
kubectl get pods  # Убеждаемся, что подов стало 4
```

4. Удалим один из подов для проверки автоматического восстановления:

```
kubectl delete pods имя_пода  # Kubernetes автоматически создаст новый под
kubectl get pods  # Проверяем, что новый под появился
```

9.3. Автомасштабирование Deployment

1. Настройка автомасштабирования по CPU:

```
kubectl autoscale deployment first-deploy --min=4 --max=6 --cpu-percent=80  # Минимум 4 пода, максимум 6, триггер на 80% загрузки CPU
```

2. Проверяем статус автомасштабирования:

```
kubectl get hpa  # Показывает текущее состояние HorizontalPodAutoscaler
```

9.4. Обновление Deployment

1. Просмотр текущего контейнера:

```
kubectl describe deployment first-deploy  # Узнаем, какой образ сейчас используется
```

2. Обновление образа контейнера:

```
kubectl set image deployment/first-deploy название_контейнера=новый_образ  # Меняем образ контейнера
kubectl rollout status deployments/first-deploy  # Проверяем, что обновление прошло успешно
```

3. Откат изменений:

```
kubectl rollout undo deployments/first-deploy  # Откатываем деплой на предыдущую версию
```

4. Перезапуск деплоя с новым образом:

```
kubectl rollout restart deployments/first-deploy  # Перезапускает все поды в Deployment
```

9.5. Создание Deployment через манифест

Вариант 1: Простой Deployment

1. Создаем файл `deploy-1-simple.yaml`:

```
apiVersion: apps/v1  # Определяем версию API
kind: Deployment  # Указываем, что создаем Deployment
metadata:
  name: my-web-deploy  # Имя Deployment
  labels:
    app: my-k8s-app  # Метка для идентификации
spec:
  selector:
    matchLabels:
      project: prod  # Указывает, какие поды управляются этим Deployment
  template:
    metadata:
      labels:
        project: prod  # Метки, которые должны соответствовать селектору
    spec:
      containers:
      - name: prod-web  # Имя контейнера
        image: k8sphp:version1  # Образ контейнера
        ports:
        - containerPort: 80  # Открываем порт 80
```

2. Применяем манифест:

```
kubectl apply -f deploy-1-simple.yaml  # Применяем файл
kubectl get deploy  # Проверяем, что Deployment создан
```

Вариант 2: Реплицированный Deployment

1. Создаем файл `deploy-2-replicas.yaml`:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-web-deploy-replicas
  labels:
    app: my-k8s-app
    env: prod
    owner: Engineer
spec:
  replicas: 3  # Количество реплик
  selector:
    matchLabels:
      project: repa
  template:
    metadata:
      labels:
        project: repa
    spec:
      containers:
      - name: prod-web
        image: k8sphp:version2
        ports:
        - containerPort: 80
```

2. Применяем манифест:

```
kubectl apply -f deploy-2-replicas.yaml
kubectl get deploy
kubectl get pods  # Проверяем созданные поды
```

3. Перенаправляем порты для тестирования доступа:

```
kubectl port-forward имя_пода 8888:80  # Доступ к поду на порту 8888
```

Вариант 3: Автомасштабируемый Deployment

1. Создаем файл `deploy-3-autoscaling.yaml`:

```
---
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: my-autoscale
spec:
  scaleTargetRef:
    apiVersion: apps/v1  # Указывает, что масштабируемый объект — это
Deployment
    kind: Deployment
    name: my-web-deploy-autoscaling  # Имя деплоя, который будет
масштабироваться
  minReplicas: 3  # Минимальное количество реплик
  maxReplicas: 5  # Максимальное количество реплик
  metrics:
  - type: Resource
    resource:
      name: cpu  # Настройка для масштабирования по загрузке CPU
```

```
      targetAverageUtilization: 70  # Средняя загрузка CPU, при которой
добавляется новая реплика
  - type: Resource
    resource:
      name: memory  # Настройка для масштабирования по загрузке памяти
      targetAverageUtilization: 80  # Средняя загрузка памяти, при которой
добавляется новая реплика

# HorizontalPodAutoscaler следит за загрузкой ресурсов (CPU и памяти) в подах
и автоматически изменяет количество реплик.
# Это особенно полезно для приложений с переменной нагрузкой, чтобы
обеспечить высокую доступность и оптимальное использование ресурсов.  #
Указывает, что автомасштабирование учитывает загрузку памяти
```

2. Применяем манифест:

```
kubectl apply -f deploy-3-autoscaling.yaml
kubectl get deploy
kubectl get pods
```

9.6. Удаление ресурсов

1. Удаляем все созданные ресурсы:

```
kubectl delete -f deploy-1-simple.yaml
kubectl delete -f deploy-2-replicas.yaml
kubectl delete -f deploy-3-autoscaling.yaml
kubectl delete deployments --all  # Удаляем все Deployment
```