Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики, управления и технологий

ДИСЦИПЛИНА:

Интеграция и развертывание программного обеспечения с помощью контейнеров

Работа на лекции

Тема:

«Основы работы с Kubernetes»

Выполнил(а): st_98, группа: АДЭУ-211

Преподаватель:

Москва

2025

Цель работы: получить практические навыки работы с кластером Kubernetes, включая развертывание базовых компонентов, настройку мониторинга и работу с service mesh.

Задачи:

1. Изучить основные концепции Kubernetes через практические вопросы.
2. Научиться анализировать и применять манифесты Kubernetes.
3. Ответить на теоретические вопросы.
4. Выполнить индивидуальное задание.

Вариант 11 (st_98) Kubernetes. Часть 1 (phpMyAdmin):

1. Запустите Kubernetes локально (k3s или minikube). Проверьте работу системных контейнеров и приложите скриншот команды: kubectl get po -n kube-system.
2. Имеется YAML с деплоем для phpMyAdmin. Измените файл:
   - Настройте запуск без пароля;
   - Фиксируйте образ на версии 5.1;
   - Добавьте Service для доступа к интерфейсу.
   - Приложите итоговый YAML.
3. Выполнить команду ps aux внутри pod; просмотреть логи за 5 минут; удалить pod; пробросить порт для отладки.
4. Доп. задание*: Создайте YAML для:
   - ConfigMap с настройками для phpMyAdmin;
   - Deployment, использующий ConfigMap;
   - Ingress, направляющий запросы по пути /admin на сервис.

Ход работы:

Устанавливаем графический интерфейс Dashboard, на рисунке 1 показан итог установки.

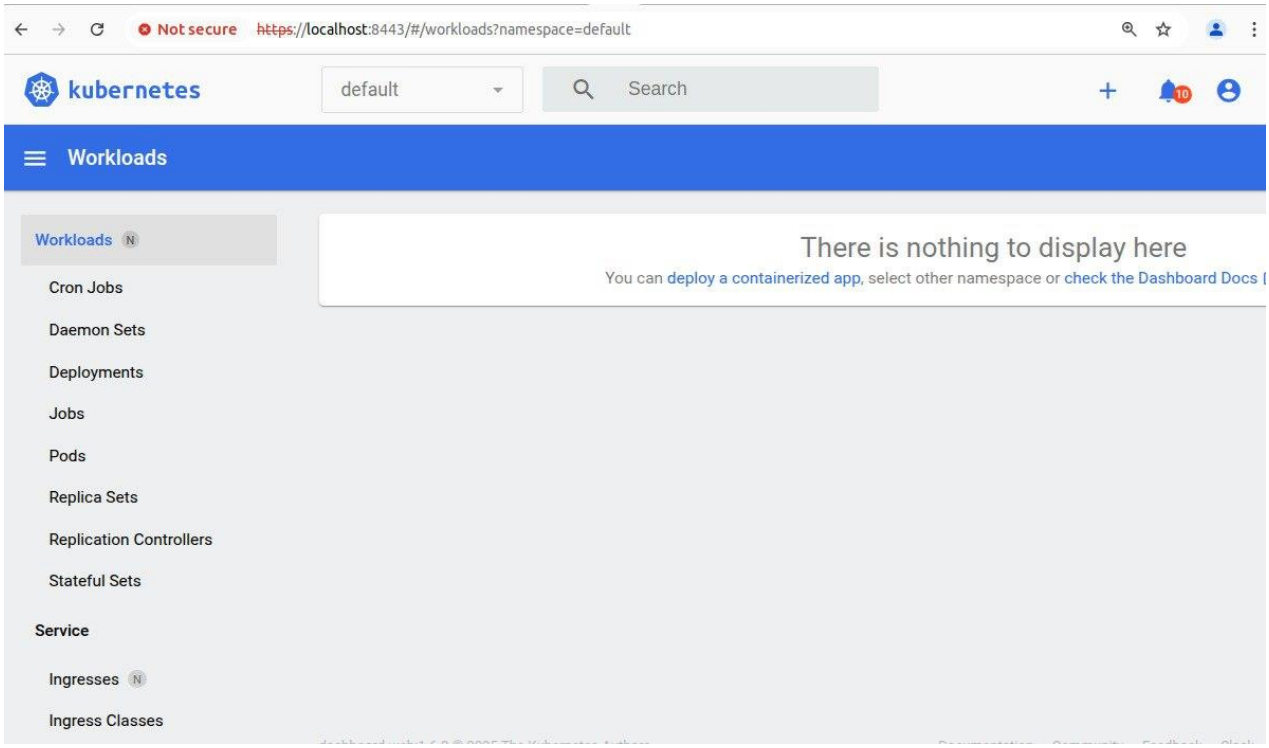

Рисунок 1 – Dashboard Kubernetes

Осуществляем запуск объектов (рисунок 2).

```
dev@dev-vm:~/lr4_1/practice/lab4_1$ kubectl create -f configmap.yml
kubectl create -f secret.yml
kubectl create -f fastapi-deployment-and-service.yml
kubectl create -f redis-deployment-and-service.yml
configmap/fastapi-config created
secret/fastapi-secret created
deployment.apps/fastapi-deployment created
service/fastapi-service created
deployment.apps/redis-deployment created
service/redis-service created
dev@dev-vm:~/lr4_1/practice/lab4_1$
```

Рисунок 2 – Запуск

Проверяем, что веб-сервис функционирует, подтверждение работоспособности на рисунке 3.

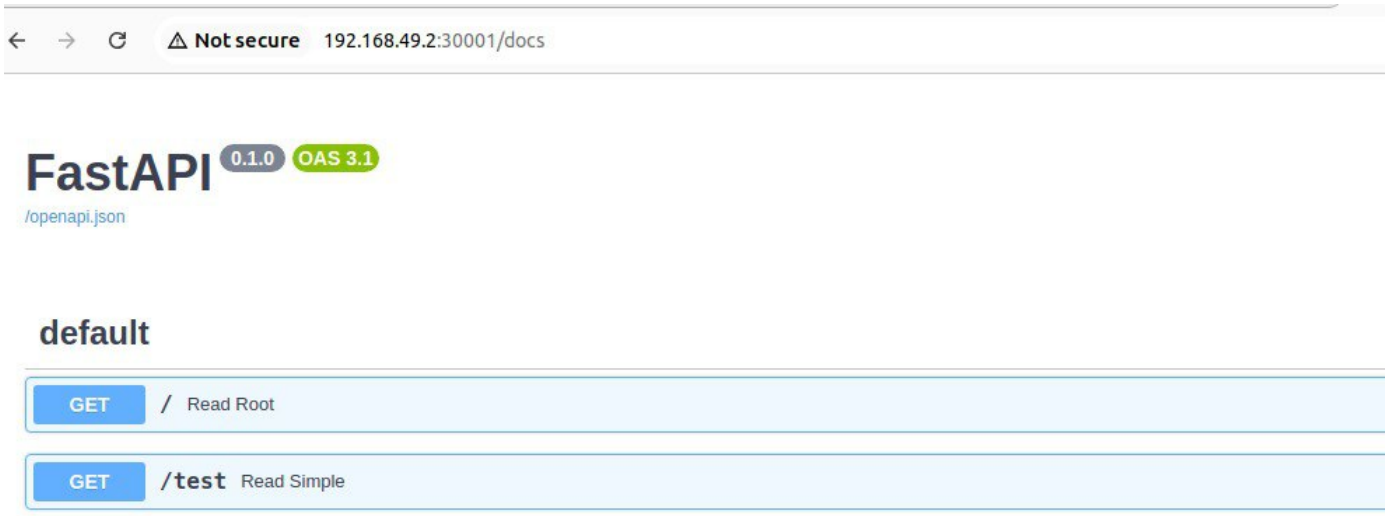

Рисунок 3 – FastAPI

Переходим к выполнению индивидуального задания:

1) **Запустите Kubernetes локально (k3s или minikube). Проверьте работу системных контейнеров и приложите скриншот команды: kubectl get po -n kube-system.**

Проверяем работу системных контейнеров, результат проверки представлен на рисунке 4.

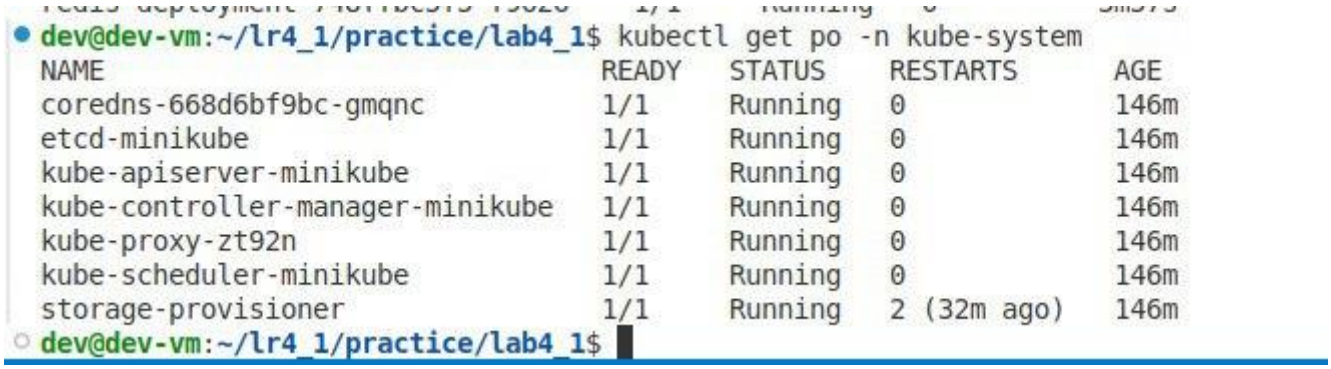

Рисунок 4 – Проверка контейнеров

2) **Имеется YAML с деплоем для phpMyAdmin. Измените файл:**
   - **Настройте запуск без пароля;**
   - **Фиксируйте образ на версии 5.1;**
   - **Добавьте Service для доступа к интерфейсу.**
   - **Приложите итоговый YAML.**

Запускаем deployment и service, используя файл phpMyAdmin-deployment-service.yaml. Команда представлена на рисунке 5. Код файла см. в Приложении.

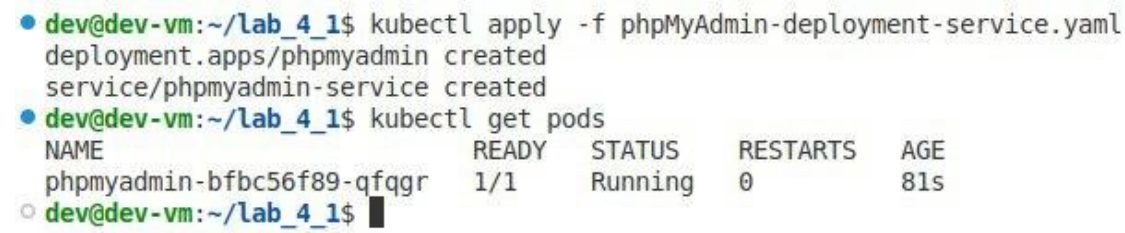

Рисунок 5 – Запуск deployment и service

Удостоверимся в работоспособности phpMyAdmin (рисунок 6).

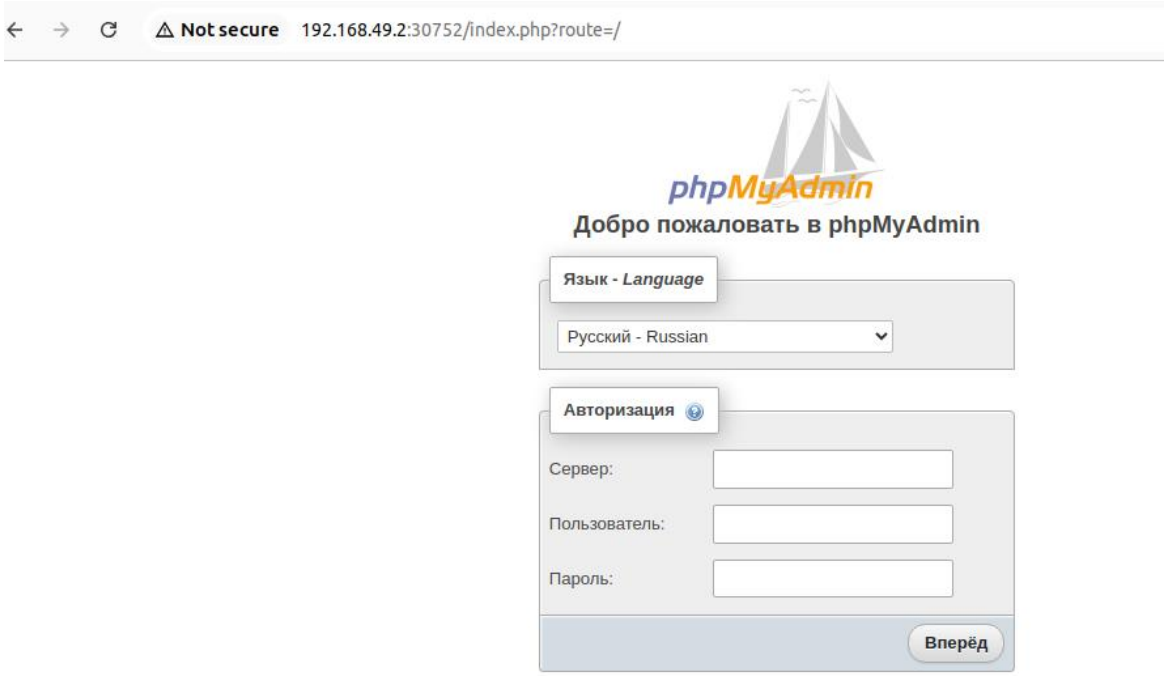

Рисунок 6 – phpMyAdmin

3) Выполнить команду ps aux внутри pod; просмотреть логи за 5 минут; удалить pod; пробросить порт для отладки.

Выполняем команду ps aux внутри pod (рисунок 7).

```
dev@dev-vm:~/lab_4_1$ kubectl exec phpmyadmin-865b679b4f-z8dsj -- ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.6 219560 28020 ?        Ss   22:56   0:00 apache2 -DFOREGROUND
www-data      23  0.0  0.2 219592  8176 ?        S    22:56   0:00 apache2 -DFOREGROUND
www-data      24  0.0  0.2 219592  8176 ?        S    22:56   0:00 apache2 -DFOREGROUND
www-data      25  0.0  0.2 219592  8048 ?        S    22:56   0:00 apache2 -DFOREGROUND
www-data      26  0.0  0.2 219592  8108 ?        S    22:56   0:00 apache2 -DFOREGROUND
www-data      27  0.0  0.2 219592  8176 ?        S    22:56   0:00 apache2 -DFOREGROUND
root          28  0.0  0.0   6700  2560 ?        Rs   23:03   0:00 ps aux
```

Рисунок 7 –Команда «ps aux»

Выполняем команду для просмотра логов за последние 5 минут. В результате ничего не оказалось, для дополнительной проверки просмотрим логи за последние 10 минут. Результат на рисунке 8.

```
[Thu Apr 03 22:56:04.383420 2025] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
dev@dev-vm:~/lab_4_1$ kubectl logs phpmyadmin-865b679b4f-z8dsj --since=5m
dev@dev-vm:~/lab_4_1$ kubectl logs phpmyadmin-865b679b4f-z8dsj --since=10m
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 10.244.0.
56. Set the 'ServerName' directive globally to suppress this message
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 10.244.0.
56. Set the 'ServerName' directive globally to suppress this message
[Thu Apr 03 22:56:04.383370 2025] [mpm_prefork:notice] [pid 1] AH00163: Apache/2.4.53 (Debian) PHP/8.0.1
8 configured -- resuming normal operations
[Thu Apr 03 22:56:04.383420 2025] [core:notice] [pid 1] AH00094: Command line: 'apache2 -D FOREGROUND'
```

Рисунок 8 – Просмотр логов

Удаляем pod (рисунок 9).

```
dev@dev-vm:~/lab_4_1$ kubectl delete pod phpmyadmin-865b679b4f-z8dsj
pod "phpmyadmin-865b679b4f-z8dsj" deleted
dev@dev-vm:~/lab_4_1$
```

Рисунок 9 – Удаление Pod

Пробрасываем порт для отладки (рисунок 10).

```
dev@dev-vm:~/lab_4_1$ kubectl port-forward phpmyadmin-865b679b4f-2g6kg 8080:80
Forwarding from 127.0.0.1:8080 -> 80
Forwarding from [::1]:8080 -> 80
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
Handling connection for 8080
```

Рисунок 10 – Порт для отладки

Вывод:

1) Освоена базовая работа с Kubernetes:
2) Освоено управление подами через Deployment.
3) Освоено обеспечение доступа через Service
4) Освоена отладка и мониторинг (logs, port-forward).

Контрольные вопросы:

1. Почему Pod может находиться в состоянии "Ожидание" (Pending)?

Pod остается в состоянии Pending по нескольким основным причинам:

Во-первых, это может быть связано с недостатком ресурсов в кластере. Когда на узлах недостаточно свободных CPU или памяти для размещения нового пода, Kubernetes не может его запустить. Для диагностики этой проблемы следует выполнить команду `kubectl describe nodes` и проверить доступные ресурсы.

Во-вторых, причина может заключаться в отсутствии подходящего узла для размещения пода. Это происходит при использовании nodeSelector, taints и tolerations, которые ограничивают размещение пода на определенных узлах. Анализ событий пода через `kubectl describe pod <pod-name>` поможет выявить эту проблему.

В-третьих, Pod может ожидать доступности PersistentVolume (PV) или успешного создания PersistentVolumeClaim (PVC). Необходимо проверить статус PVC командой `kubectl get pvc` и его описание через `kubectl describe pvc <pvc-name>`.

Другими возможными причинами являются:

- проблемы с загрузкой образа контейнера (неверный тег или недоступность репозитория)
- отсутствие свободных IP-адресов в сети кластера
- ограничения политик безопасности (SecurityContext, PodSecurityPolicy)

2. Как просмотреть журналы (логи) контейнера в Pod?

Для просмотра журналов контейнера в Kubernetes используются следующие команды:

Для пода с одним контейнером:

kubectl logs <pod-name>

Для пода с несколькими контейнерами необходимо указать конкретный контейнер:

kubectl logs <pod-name> -c <container-name>

Для отслеживания логов в реальном времени:

kubectl logs -f <pod-name>

Для просмотра логов за определенный временной период:

kubectl logs --since=5m <pod-name> # за последние 5 минут

3. Что произойдет при выполнении команды kubectl logs для пода с двумя контейнерами?

При попытке выполнить команду `kubectl logs some-pod` для пода, содержащего два контейнера, Kubernetes вернет ошибку: «error: a container name must be specified for pod some-pod, choose one of: [container1 container2]»

Это происходит потому, что:

1. Каждый контейнер в поде ведет собственный журнал событий

2. Kubernetes не может автоматически определить, логи какого контейнера нужно показать

3. Не существует понятия "главного" контейнера в поде

Для правильного просмотра логов необходимо явно указать имя контейнера с помощью флага -c:

kubectl logs some-pod -c container1

Если требуется просмотреть логи всех контейнеров в поде одновременно, можно использовать:

kubectl logs some-pod --all-containers

1. Что такое кластер Kubernetes?

Кластер Kubernetes — это набор узлов (серверов), объединенных в единую систему для запуска и управления контейнеризированными приложениями. Он обеспечивает:

- Автоматическое развертывание приложений
- Масштабирование (увеличение/уменьшение количества экземпляров)
- Балансировку нагрузки между подами
- Самовосстановление (перезапуск упавших контейнеров)
- Управление конфигурациями и секретами

Основные компоненты кластера

- Control Plane (Главный узел) – управляет кластером
- Worker Nodes (Рабочие узлы) – выполняют приложения
- Сеть (CNI) – обеспечивает связь между компонентами

2. Что такое узел (Node)?

Узел (Node) — это физическая или виртуальная машина, которая является частью кластера Kubernetes.

Типы узлов:

1. Master Node (Главный узел)

- Управляет кластером
- Отвечает за планирование задач, API, хранение состояния кластера

2. Worker Node (Рабочий узел)/Запускает пользовательские приложения (поды)

3. За что отвечает главный узел (Master Node)?

Главный узел управляет всем кластером и включает следующие компоненты:

- kube-apiserver
  - Главный API Kubernetes (через него работают все команды `kubectl`)
  - Проверяет запросы и передает их другим компонентам
- etcd
  - База данных кластера (хранит все настройки и состояние)
  - Работает по принципу key-value хранилища
- kube-scheduler
  - Распределяет поды по рабочим узлам
  - Учитывает:
    - Доступные ресурсы (CPU, RAM)
    - Ограничения (taints/tolerations)
    - Локацию данных (если используется PVC)
- kube-controller-manager
  - Отслеживает состояние кластера
  - Управляет:
    - Deployments (реплики подов)
    - Nodes (доступность рабочих узлов)
    - Services/Endpoints (сетевые правила)
- cloud-controller-manager (опционально)
  - Интеграция с облачными провайдерами (AWS, GCP, Azure)
  - Управляет:
    - Внешними балансировщиками нагрузки
    - Дисками (Persistent Volumes)
    - Сетевыми маршрутами

ПРИЛОЖЕНИЕ

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: phpmyadmin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: phpmyadmin
  template:
    metadata:
      labels:
        app: phpmyadmin
    spec:
      containers:
        - name: phpmyadmin
          image: phpmyadmin/phpmyadmin:5.1
          env:

            - name: ALLOW_NO_PASSWORD
              value: "yes"

          ports:
            - containerPort: 80
---
apiVersion: v1
kind: Service
metadata:
  name: phpmyadmin-service
spec:
  selector:
    app: phpmyadmin
```

```
ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 80  # Для phpMyAdmin

type: LoadBalancer
```