|  |  |  |
| --- | --- | --- |
|  | Изображение выглядит как зарисовка, рисунок, символ, корона  Автоматически созданное описание |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Инструментального и прикладного программного обеспечения

**ПРАКТИЧЕСКАЯ РАБОТА №5**

по дисциплине «Технологии виртуализации клиент-серверных приложений»

**Тема практической работы: «Взаимодействие с minikube»**

**Студент группы** ИКБО-20-21 Сидоров С.Д.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Руководитель практической работы** старший преподаватель Волков М.Ю.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Работа представлена «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Допущен к работе «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Москва 2024

**СОДЕРЖАНИЕ**

[ЦЕЛЬ РАБОТЫ 3](#_Toc181974021)

[ВЫВОД 8](#_Toc181974022)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 9](#_Toc181974023)

ЦЕЛЬ РАБОТЫ

Загрузить и установить minikube, создать deployment и pod, просмотреть события кластера, а также конфигурацию. Получить навыки запуска приложения с возможностью получать запросы из сети Интернет, а также продемонстрировать умение подключать дополнения.

ХОД РАБОТЫ

Для работы был установлен и запущен minikube (рисунок 1).

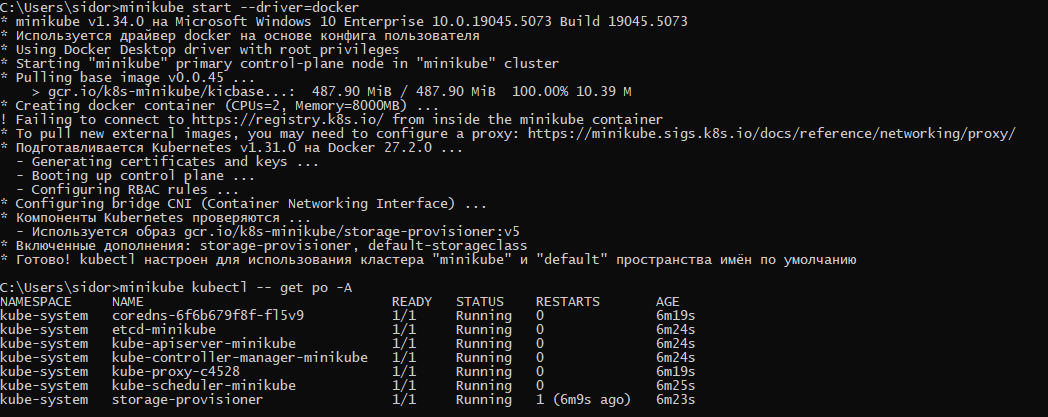


Рисунок 1 – Установка и запуск Minikube

Далее был создан файл с сервером и Dockerfile (рисунки 2, 3).

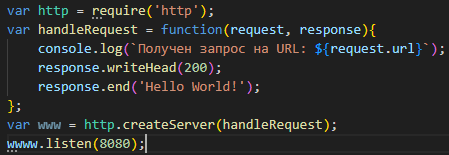


Рисунок 2 – server.js

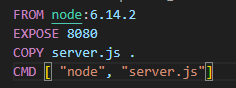


Рисунок 3 – Dockerfile

После этого был собран образ (рисунок 4).



Рисунок 4 – Сборка

Была применена конфигурация deployment.yaml (рисунок 5).

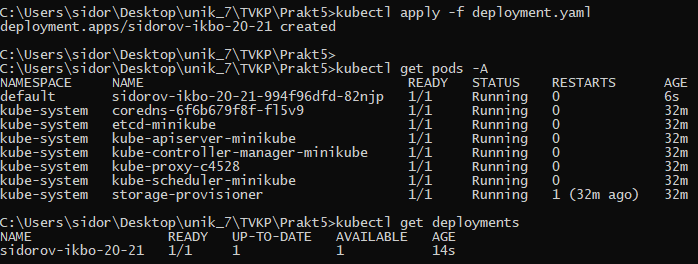


Рисунок 5 – Применение конфигурации

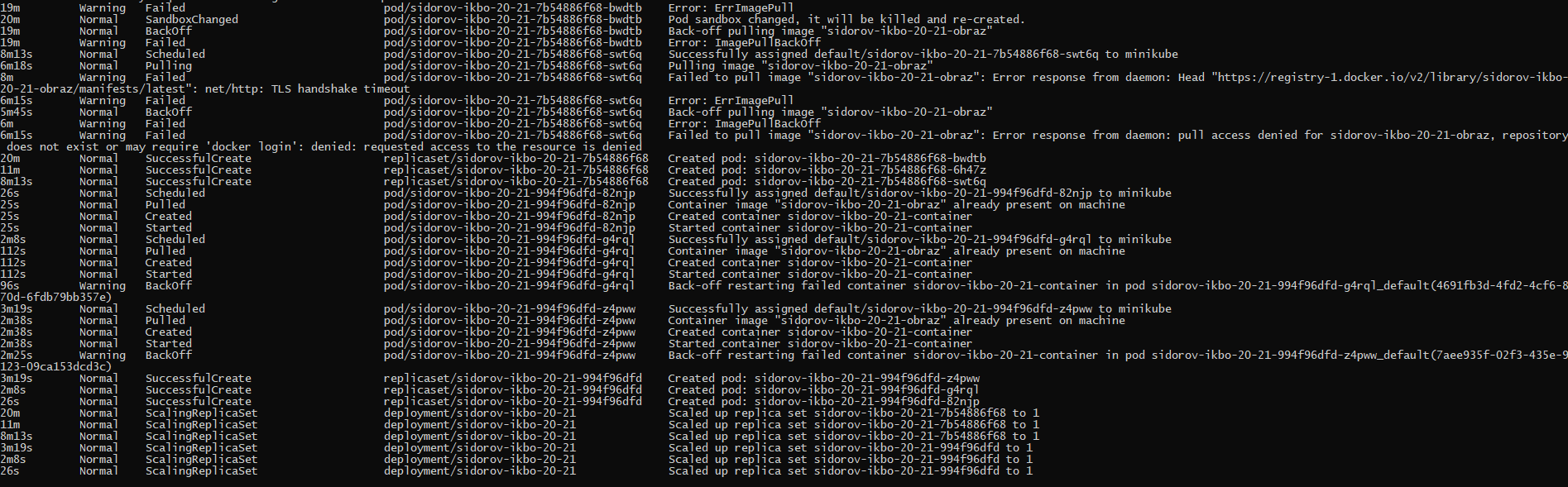


Рисунок 6 – События кластера

Конфигурация представлена на рисунке 7.

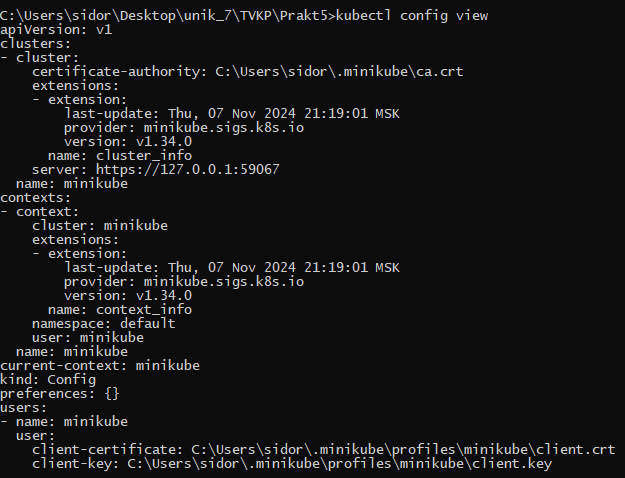


Рисунок 7 – Конфигурация kubectl

Был создан под с deployment, доступным для публичной сети интернет (рисунки 8, 9).

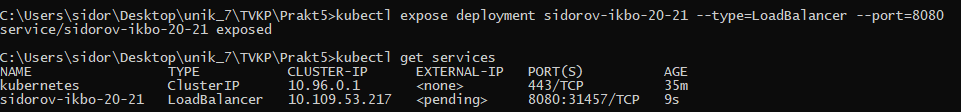


Рисунок 8 – Сервисы



Рисунок 9 – Запущенный сервис hello-node

Было включено дополнение ingress (рисунок 10).

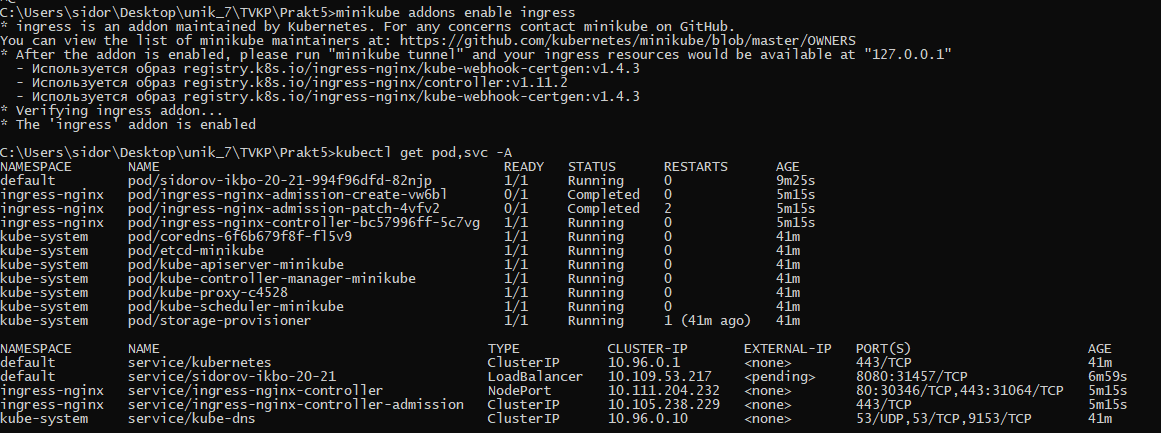


Рисунок 10 – Включение дополнения ingress

Был включен dashboard (рисунок 11).

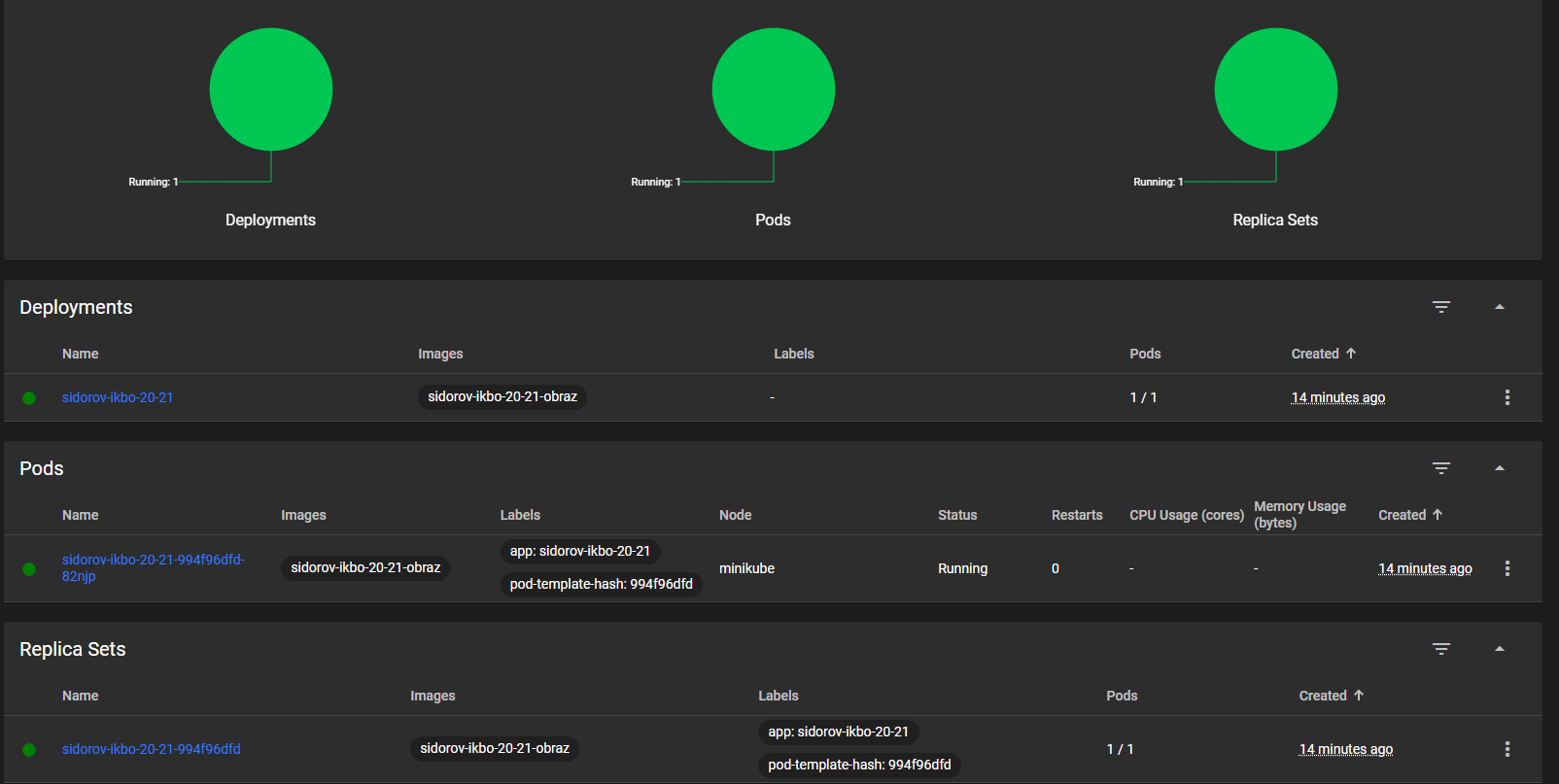


Рисунок 11 – dashboard

Вопросы

1. Назовите виды контроллеров в Kubernetes.

Основные виды контроллеров:

* Deployment: управляет репликами подов и автоматическими обновлениями приложений.
* ReplicaSet: гарантирует, что запущено определенное количество идентичных подов.
* DaemonSet: запускает по одному поду на каждом узле кластера (например, для системных процессов).
* StatefulSet: используется для управления состоянием подов, особенно для приложений, где важна последовательность запуска и доступ к стабильным именам и хранилищу.
* Job: запускает задачи, которые должны выполниться однократно и завершиться.
* CronJob: запускает задания по расписанию, похож на системный cron.

2. Как называется командная строка в Kubernetes?

kubectl

3. Что такое под?

Базовая единица развертывания в Kubernetes, представляющая собой один или несколько контейнеров (например, Docker), которые работают вместе на одном узле и имеют доступ к общему сетевому и файловому пространству. Контейнеры в поде могут обмениваться данными и взаимодействовать друг с другом.

4. Назовите 2 типа ресурсов, из которых состоит кластер Kubernetes.

Worker Nodes (узлы): рабочие узлы, на которых запускаются поды и контейнеры, и которые обрабатывают нагрузку приложений.

Master Node (узел управления): главный узел, который управляет кластером, распределяет задачи и следит за их состоянием.

5. Чем Kubernetes отличается от Docker Swarm?

Масштабируемость и сложность: Kubernetes предоставляет более богатый набор функций для автоматизации, масштабирования и управления состоянием приложений, но он сложнее в настройке и администрировании. Docker Swarm проще, но менее гибок и масштабируем.

Управление состоянием: Kubernetes активно управляет состоянием приложений (например, использует контроллеры для поддержания целевого состояния). Docker Swarm больше ориентирован на ручное управление и не имеет столь развитой системы контроля за состоянием.

Поддержка сетей и балансировки нагрузки: Kubernetes предоставляет более гибкие механизмы сетевого взаимодействия и маршрутизации, а также собственную балансировку нагрузки (Service). Docker Swarm также поддерживает балансировку, но имеет меньше возможностей настройки.

Обработка ошибок и самовосстановление: Kubernetes имеет более продвинутые механизмы обработки сбоев и самовосстановления приложений

ВЫВОД

В ходе выполнения данной работы были получены практические навыки развертывания и управления контейнеризованными приложениями в кластере Kubernetes с использованием Minikube. На практике были изучены принципы работы с контроллерами, обеспечивающими поддержание состояния приложений, такие как Deployment и Service. Освоены основные команды kubectl, а также процессы создания, настройки и мониторинга подов, сервисов и других ресурсов Kubernetes. Практическая часть работы помогла лучше понять, как Kubernetes автоматически масштабирует приложения, поддерживает отказоустойчивость и предоставляет гибкие возможности для управления сетевыми настройками. Полученные знания и навыки являются основой для последующего изучения Kubernetes и работы с контейнеризированными приложениями в реальных кластерах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Docker Documentation | Docker Documentation – Текст: электронный [сайт]. – URL: <https://docs.docker.com/>

2. Spring Boot Documentation | Spring – Текст: электронный [сайт]. – URL: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>

3. Minikube – Текст: электронный [сайт]. – URL: <https://kubernetes.io/ru/docs/tasks/tools/install-minikube/>

4. Docker Compose Documentation | Docker Documentation – Текст: электронный [сайт]. – URL: <https://docs.docker.com/compose/>

5. 50 вопросов по Docker, которые задают на собеседованиях, и ответы на них | Хабр. – Текст: электронный [сайт]. – URL: <https://habr.com/ru/company/southbridge/blog/528206/>