|  |  |  |
| --- | --- | --- |
|  | 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 №6**

по дисциплине «Технологии виртуализации клиент-серверных приложений»

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

**Студент группы** ИКБО-20-21 Фомичев Р.А.

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

(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](#_Toc182036357)

[ВОПРОСЫ 15](#_Toc182036358)

[ВЫВОД 17](#_Toc182036359)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 18](#_Toc182036360)

ЦЕЛЬ РАБОТЫ

Изучить и реализовать процесс развертывания контейнеризованных приложений в среде Kubernetes, освоить управление конфигурацией и настройкой компонентов системы, таких как Deployment, Service, Ingress, StatefulSet, ConfigMap и Secret. Овладеть навыками организации и параметризации приложений, автоматизации развертывания с использованием CI/CD и обеспечением внешнего доступа для сервисов, а также применить на практике механизмы хранения данных через PersistentVolume и PersistentVolumeClaim.

ХОД РАБОТЫ

Был создан ресурс deployment, он представлен на рисунке 1.



Рисунок 1 – Deployment

После был создан сервис для балансировки нагрузки. Сервис представлен на рисунке 2.

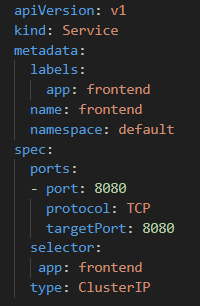


Рисунок 2 – Сервис

Deployment был обновлен. Обновленный deployment представлен на рисунке 3.



Рисунок 3 – Deployment

Был создан пароль для redis. Создание пароля представлено на рисунке 4.



Рисунок 4 – Создание пароля

Deployment был обновлен. Обновленный Deployment представлен на рисунке 5.

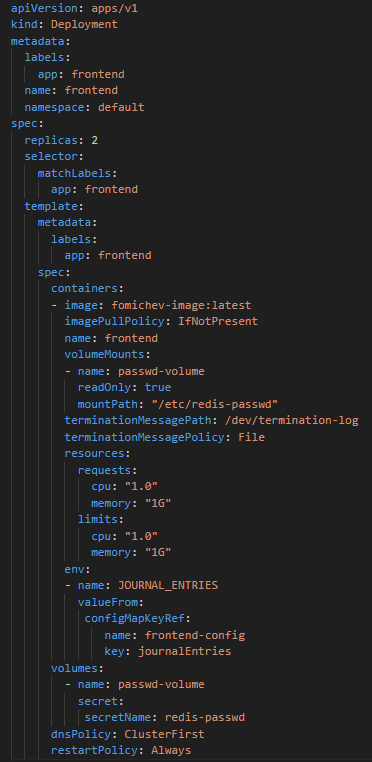


Рисунок 5 – Deployment

Был создан StatefulSet для Redis для управления состоянием подов. Statefulset представлен на рисунке 6.

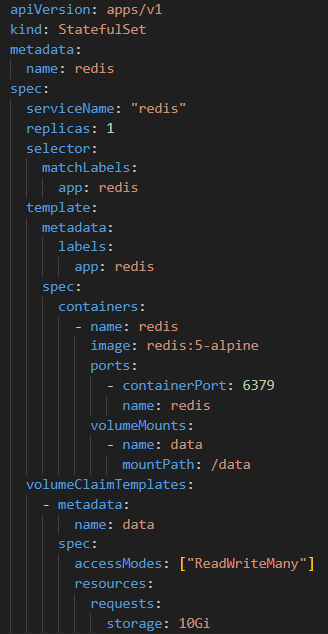


Рисунок 6 – Statefulset

Был создан сценарий для запуска во всех контейнерах. Сценарий представлен на рисунке 7

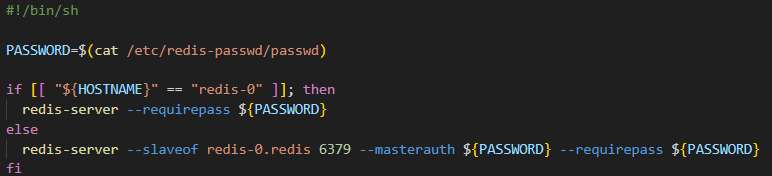


Рисунок 7 – Сценарий запуска

Сценарий был оформлен в виде ConfigMap. Создание ConfigMap представлено на рисунке 8.

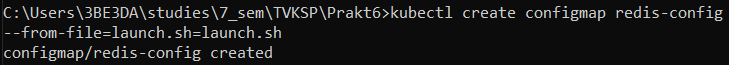


Рисунок 8 – ConfigMap

Полное определение сервиса Redis с тремя репликами представлено на рисунке 9.

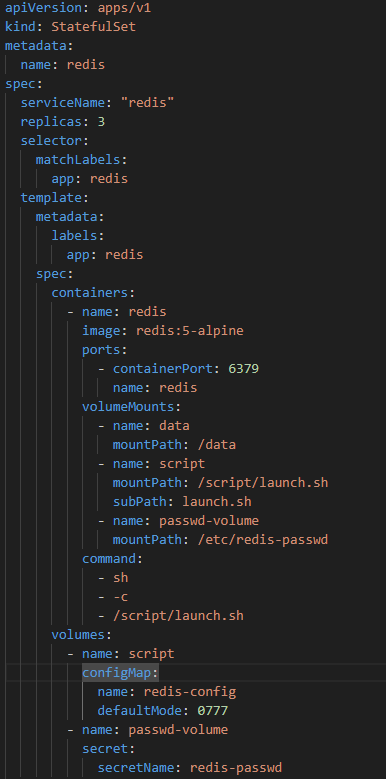


Рисунок 9 – StatefulSet

Было создано два сервиса Kubernetes для чтения данных из Redis и выполнения записи. Сервисы представлены на рисунках 10, 11.

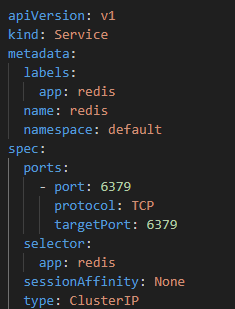


Рисунок 10 – Service

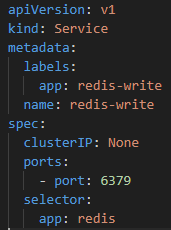


Рисунок 11 – Service

Статические образы были собраны в контейнер nginx. Deployment представлен на рисунке 12.



Рисунок 12 – Deployment

Был создан сервис для балансировки нагрузки. Сервис представлен на рисунке 13.

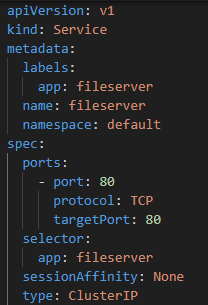


Рисунок 13 – Service

Был создан Ingress для маршрутизации. Ingress представлен на рисунке 14.

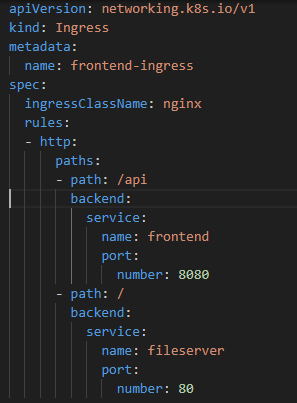


Рисунок 14 – Ingress

Были применены конфигурация, описанные выше. Применение конфигураций представлено на рисунках 15 - 17.

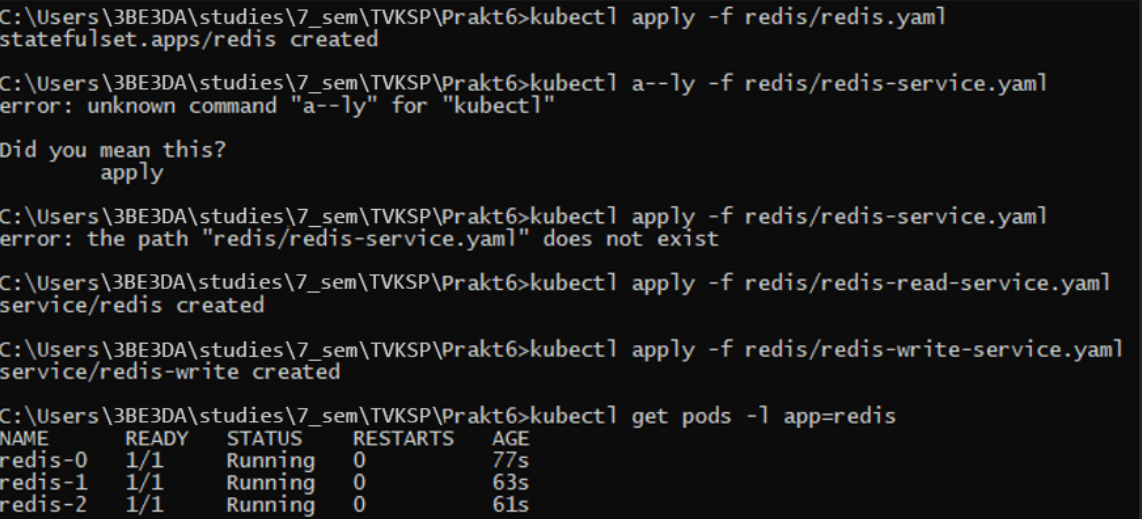


Рисунок 15 – Применение конфигурации redis

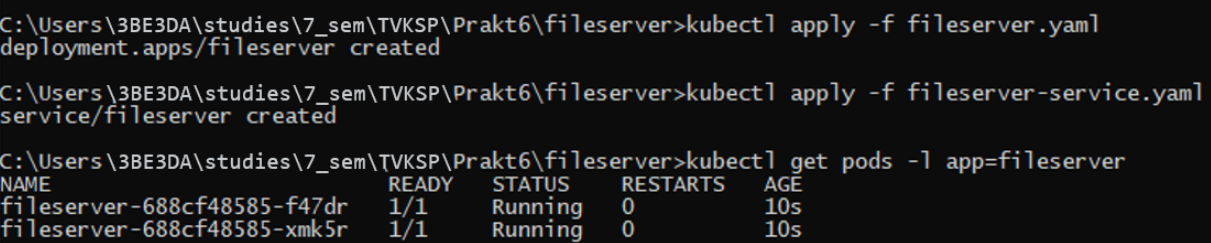


Рисунок 16 – Применение конфигурации fileserver



Рисунок 17 – Применение конфигурации ingress

Была получена информация о ресурсах ingress. Информация представлена на рисунке 18.

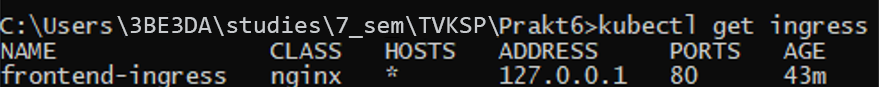


Рисунок 18 – Информация о ресурсах ingress

Работоспособность программы представлена на рисунках 19 – 21.

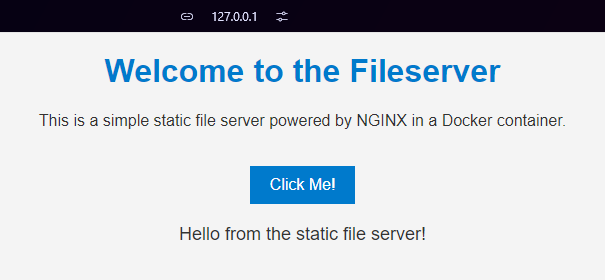


Рисунок 19 – Работоспособность системы

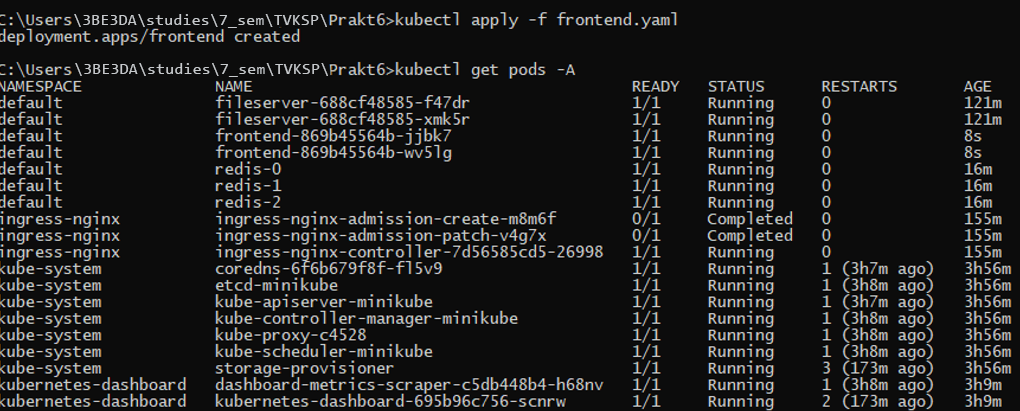


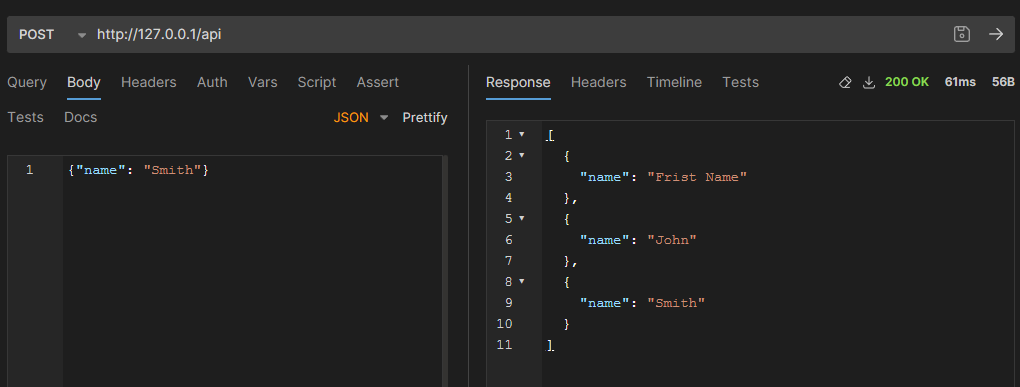
Рисунок 20 – Применение конфигурации frontend  


Рисунок 21 – Отправка post запроса

ВОПРОСЫ

1. Для чего нужен ресурс Deployment?

Ресурс Deployment в Kubernetes используется для управления жизненным циклом приложений, поддерживая масштабирование, отказоустойчивость и обновления без простоя. Он позволяет создавать и управлять репликами контейнеров, автоматизируя развертывание, обновление и откат изменений приложения.

2. Почему не стоит хранить пароли в ConfigMap?

ConfigMap не обеспечивает достаточного уровня безопасности для хранения конфиденциальной информации, такой как пароли. Данные в ConfigMap не зашифрованы и могут быть легко прочитаны, поэтому для хранения чувствительных данных используется ресурс Secret, который предназначен для безопасного хранения и передачи таких данных в контейнеры.

1. Что необходимо для настройки внешнего доступа для HTTP-трафика? Назовите шаги.

Чтобы настроить внешний доступ для HTTP-трафика в Kubernetes, нужно:

* Создать ресурс Service с типом LoadBalancer или NodePort, который будет распределять HTTP-трафик внутри кластера.
* Настроить Ingress для маршрутизации HTTP-запросов к нужным сервисам. Ingress также позволяет управлять SSL-сертификатами для HTTPS и маршрутизацией на основе путей и доменных имен.
* Убедиться, что в кластере установлен и настроен контроллер Ingress (например, nginx или другой), который будет обрабатывать внешние запросы.

1. Чем отличается развертывание stateful от развертывания клиентского приложения?

Развертывание stateful-приложений (например, Redis) требует сохранения состояния, поэтому для него используется StatefulSet, который обеспечивает постоянное хранилище данных с привязкой к определённым репликам. Это важно для обеспечения уникальности и порядка реплик. В отличие от этого, развертывание клиентских приложений обычно является stateless, то есть не хранит состояния и может быть реплицировано с помощью Deployment, обеспечивая масштабируемость и отказоустойчивость без необходимости сохранять данные между перезапусками.

1. Где хранится том с секретными данными?

Том с секретными данными создается в оперативной памяти (tmpfs), что обеспечивает его недоступность на физическом диске и повышает безопасность. Секретные данные монтируются в контейнеры через объект Volume, а сами секреты хранятся в объекте Secret в Kubernetes.

1. Как работает связка PersistentVolume и PersistentVolumeClaim?

PersistentVolume (PV) представляет собой выделенное хранилище в кластере, которое администратор создает заранее или которое создается динамически. PersistentVolumeClaim (PVC) — это запрос на ресурс хранилища, создаваемый приложением, когда ему нужно постоянное хранилище. Когда PVC соответствует PV по заданным критериям (размер, тип доступа), Kubernetes связывает их, предоставляя контейнеру постоянный том для хранения данных.

ВЫВОД

В данной работе были рассмотрены основные шаги по развертыванию приложения для Node.js в Kubernetes с использованием контейнеризации и декларативного подхода к конфигурации системы.

Полученные знания позволят применять лучшие практики при работе с облачными платформами и эффективно управлять контейнеризированными приложениями в среде Kubernetes.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

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

2. Kubernetes – Текст: электронный [сайт]. – URL: <https://kubernetes.io/>

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

4. 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/>