Департамент образования города Москвы

Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»

Институт цифрового образования
Департамент информатики, управления и технологий

Лабораторная работа 4.1
по дисциплине «Интеграция и развертывание программного обеспечения с помощью контейнеров»

Тема: «Создание и развертывание полнофункционального приложения»

Направление подготовки 38.03.05 – бизнес-информатика
Профиль подготовки «Аналитика данных и эффективное управление»
(очная форма обучения)

Выполнила:
Студентка группы АДЭУ-211
St_88

Москва
2025

ВВЕДЕНИЕ

Цель работы: применить полученные знания по созданию и развертыванию полнофункционального приложения с использованием Docker и Kubernetes.

Задачи:

1. Создать фронтенд-приложение на React.
2. Создать бэкенд-приложение на Node.js с использованием Express и MongoDB.
3. Контейнеризировать фронтенд и бэкенд приложения с помощью Docker.
4. Создать манифесты Kubernetes для развертывания приложения.
5. Развернуть приложение в кластере Kubernetes.
6. Протестировать работоспособность приложения.

Вариант 1.

Создайте приложение "Список задач" с использованием React, Node.js, Express и MongoDB.

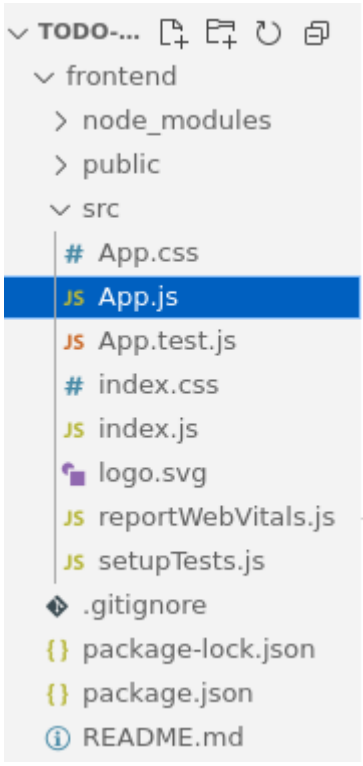
ХОД РАБОТЫ

Приложение «Список задач» будет представлять из себя веб-страницу с полем для ввода задач и их записью в таблицу.

1. Инициализация фронтовой составляющей приложения на React:

```
mgpu@mgpu-VirtualBox:~/todo-app$ npx create-react-app frontend
Need to install the following packages:
create-react-app@5.1.0
Ok to proceed? (y) y

mgpu@mgpu-VirtualBox:~/todo-app$ cd frontend
mgpu@mgpu-VirtualBox:~/todo-app/frontend$
```

Результат выполнения команды (папка была пуста):



2. Код файла App.js (на React) - фронтенд:

App.js ●

frontend > src > App.js > default

```
import React, { useState, useEffect } from 'react';
import './App.css';

function App() {
  const [tasks, setTasks] = useState([]);
  const [newTask, setNewTask] = useState('');

  useEffect(() => {
    fetchTasks();
  }, []);

  const fetchTasks = async () => {
    try {
      const response = await fetch('http://backend-service:5000/api/tasks');
      const data = await response.json();
      setTasks(data);
    } catch (error) {
      console.error('Error fetching tasks:', error);
    }
  };

  const addTask = async () => {
    if (!newTask.trim()) return;

    try {
      const response = await fetch('http://backend-service:5000/api/tasks', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ text: newTask }),
      });
      const data = await response.json();
      setTasks([...tasks, data]);
      setNewTask('');
    } catch (error) {
      console.error('Error adding task:', error);
    }
  };

  const updateTask = async (id, updatedText) => {
    try {
      await fetch(`http://backend-service:5000/api/tasks/${id}`, {
        method: 'PUT',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ text: updatedText }),
      });
      fetchTasks();
    } catch (error) {
      console.error('Error updating task:', error);
    }
  };

```

```
  const deleteTask = async (id) => {
    try {
      await fetch(`http://backend-service:5000/api/tasks/${id}`, {
        method: 'DELETE',
      });
      setTasks(tasks.filter(task => task._id !== id));
    } catch (error) {
      console.error('Error deleting task:', error);
    }
  };

  return (
    <div className="App">
      <h1>Список задач</h1>
      <div className="task-input">
        <input
          type="text"
          value={newTask}
          onChange={(e) => setNewTask(e.target.value)}
          placeholder="Добавить новую задачу"
        />
        <button onClick={addTask}>Добавить</button>
      </div>
      <ul className="task-list">
        {tasks.map((task) => (
          <li key={task._id} className="task-item">
            <input
              type="text"
              defaultValue={task.text}
              onBlur={(e) => updateTask(task._id, e.target.value)}
            />
            <button onClick={() => deleteTask(task._id)}>Удалить</button>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

3. Докерфайл для фронтенд сервера приложения:

App.js Dockerfile ●

frontend > Dockerfile

```
1   #Sborka
2   FROM node:16 AS build
3
4   WORKDIR /app
5
6   COPY package*.json ./
7
8   RUN npm install
9   COPY . .
10
11  RUN npm run build
12
13  #Proda
14  FROM nginx:alpine
15
16  COPY --from=build /app/build /usr/share/nginx/html
17
18  EXPOSE 80
19
20  CMD ["nginx", "-g", "daemon off;"]
```

4. В новой папке проекта backend инициализируем составляющую приложения, отвечающую за бек на Node.js:

```
mgpu@mgpu-VirtualBox:~/todo-app$ cd backend
mgpu@mgpu-VirtualBox:~/todo-app/backend$ npm init -y
Wrote to /home/mgpu/todo-app/backend/package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

5. Загрузка всех необходимых пакетов (express, mongoose, cors) и зависимостей, наполнит файл package.json с основной информацией о приложении:

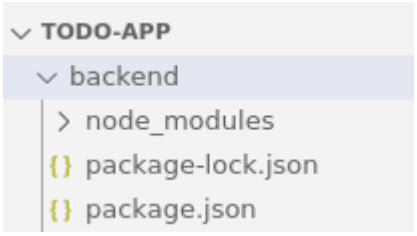
```
mgpu@mgpu-VirtualBox:~/todo-app/backend$ npm install express mongoose cors

added 85 packages, and audited 86 packages in 1m

15 packages are looking for funding
  run `npm fund` for details
```

Содержимое файла package.json:

backend > {} package.json > ...

```json
{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^4.18.2",
    "mongoose": "^8.0.3",
    "cors": "^2.8.5"
  }
}
```

Результат выполнения вышеперечисленных команд (папка была пуста):



6. Код файла server.js (отвечает за основную логику приложения на беке):

db-configmap.yaml | mongodb-secret.yaml | backend-deployment.yaml | server.js | mongodb-service.y

backend > server.js > ...

```js
// Получаем настройки из переменных окружения
const {
  MONGODB_HOST,
  MONGODB_PORT,
  MONGODB_DATABASE,
  MONGODB_USERNAME,
  MONGODB_PASSWORD
} = process.env;

// Формируем строку подключения
const mongoUri = `mongodb://${MONGODB_USERNAME}:${MONGODB_PASSWORD}@${MONGODB_HOST}:${MONGODB_P

// Подключаемся к MongoDB
mongoose.connect(mongoUri, {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
```

```
// Модель задачи
const Task = mongoose.model('Task', {
  text: String,
  createdAt: { type: Date, default: Date.now }
});

// API маршруты
app.get('/api/tasks', async (req, res) => {
  try {
    const tasks = await Task.find().sort({ createdAt: -1 });
    res.json(tasks);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

app.post('/api/tasks', async (req, res) => {
  try {
    const task = new Task({ text: req.body.text });
    await task.save();
    res.status(201).json(task);
  } catch (error) {
    res.status(400).json({ error: error.message });
  }
});

app.put('/api/tasks/:id', async (req, res) => {
  try {
    const task = await Task.findByIdAndUpdate(
      req.params.id,
      { text: req.body.text },
      { new: true }
    );
    res.json(task);
  } catch (error) {
    res.status(404).json({ error: 'Task not found' });
  }
});

app.delete('/api/tasks/:id', async (req, res) => {
  try {
    await Task.findByIdAndDelete(req.params.id);
    res.status(204).end();
  } catch (error) {
    res.status(404).json({ error: 'Task not found' });
  }
});

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

7. Докерфайл для бекенд сервера приложения:

App.js | Dockerfile frontend | server.js | Dockerfile backend ●

backend > Dockerfile

```
1   FROM node:16
2
3   WORKDIR /app
4
5   COPY package*.json ./
6
7   RUN npm install
8
9   COPY . .
10
11  EXPOSE 5000
12
13  CMD ["node", "server.js"]
```

8. Создание папки k8s в корне проекта, которая будет содержать в себе все необходимые для развертывания приложения с Kubernetes манифесты:

Манифест для бека и его сервера – разворачивает одну реплику, используя локальный образ:

k8s > ! backend-deployment.yaml

```
1   apiVersion: apps/v1
2   kind: Deployment
3   metadata:
4     name: backend-deployment
5   spec:
6     replicas: 1
7     selector:
8       matchLabels:
9         app: backend
10    template:
11      metadata:
12        labels:
13          app: backend
14      spec:
15        containers:
16        - name: backend
17          image: todo-backend
18          imagePullPolicy: Never
19          ports:
20          - containerPort: 5000
21          env:
22          - name: PORT
23            value: "5000"
24          # Переменные из ConfigMap
25          - name: MONGODB_DATABASE
26            valueFrom:
27              configMapKeyRef:
28                name: mongodb-config
29                key: MONGODB_DATABASE
30          - name: MONGODB_HOST
```

```
        valueFrom:
          configMapKeyRef:
            name: mongodb-config
            key: MONGODB_HOST
      - name: MONGODB_PORT
        valueFrom:
          configMapKeyRef:
            name: mongodb-config
            key: MONGODB_PORT
      # Секретные переменные
      - name: MONGODB_USERNAME
        valueFrom:
          secretKeyRef:
            name: mongodb-secret
            key: MONGODB_USERNAME
      - name: MONGODB_PASSWORD
        valueFrom:
          secretKeyRef:
            name: mongodb-secret
            key: MONGODB_PASSWORD
      readinessProbe:  # Проверка готовности
        httpGet:
          path: /api/tasks
          port: 5000
        initialDelaySeconds: 10
        periodSeconds: 5
---
apiVersion: v1
kind: Service
metadata:
  name: backend-service
spec:
  selector:
    app: backend
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
```

Манифест для фронта и его сервера – разворачивает одну реплику, используя локальный образ:

k8s > ! frontend-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 1  # Только один экземпляр
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: frontend
        image: todo-frontend  # Локальный образ
        imagePullPolicy: Never
        ports:
        - containerPort: 80
        env:
        - name: REACT_APP_API_URL
          value: "http://backend-service:5000"
---
apiVersion: v1
kind: Service
metadata:
  name: frontend-service
spec:
  type: LoadBalancer
  selector:
    app: frontend
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Манифест для развертывания MongoDB – разворачивает одну реплику, используя официальный образ версии 4.4.18 (версия не новая, так как процессор не поддерживает AVX):

k8s > ! mongodb-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  replicas: 1  # Только один экземпляр
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb
        image: mongo:4.4.18
        ports:
        - containerPort: 27017
        env:
        - name: MONGO_INITDB_ROOT_USERNAME
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: MONGODB_USERNAME
        - name: MONGO_INITDB_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mongodb-secret
              key: MONGODB_PASSWORD
        volumeMounts:
        - name: mongodb-data
          mountPath: /data/db
      volumes:
      - name: mongodb-data
        persistentVolumeClaim:
          claimName: mongodb-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

Configmap.yaml для MongoDB, включающий в себя достаточно конфигурационных данных, которые используются контейнерами в поде:

k8s > ! mongodb-configmap.yaml

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongodb-config
data:
  MONGODB_DATABASE: "todoapp"  # Название БД
  MONGODB_HOST: "mongodb-service"  # Сервис MongoDB в Kubernetes
  MONGODB_PORT: "27017"  # Порт
```

PVC.yaml для MongoDB – том для хранения данных:

k8s > ! mongodb-pvc.yaml

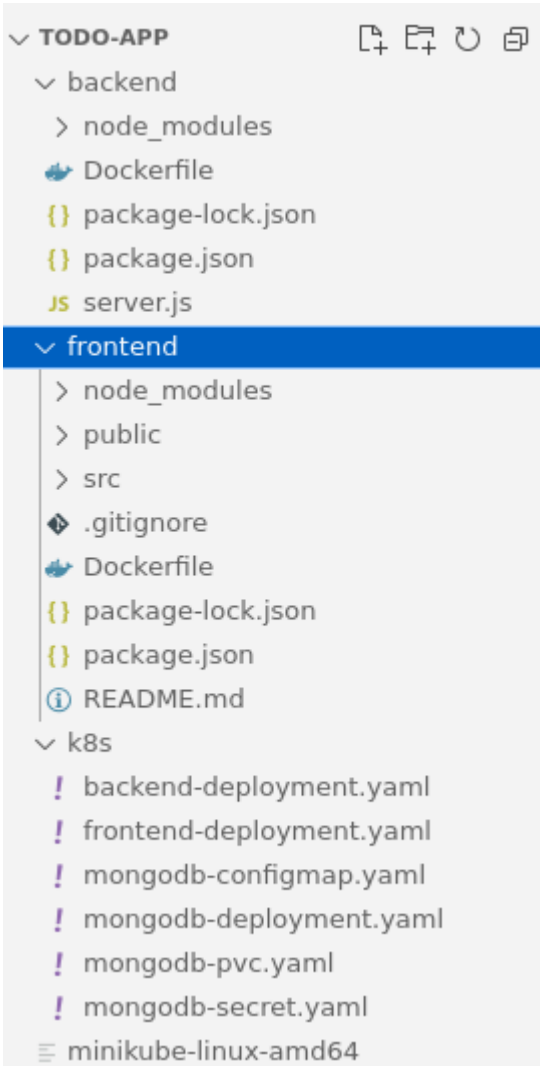
```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi

```

Sectret.yaml для MongoDB, в котором хранятся учетные данные:

k8s > ! mongodb-secret.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
type: Opaque
stringData:
  MONGODB_USERNAME: "admin"
  MONGODB_PASSWORD: "password123"
```

9. Итоговая структура проекта:



10.Установка Minikube:

```
mgpu@mgpu-VirtualBox:~/todo-app$ curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  119M  100  119M    0     0  11.7M      0  0:00:10  0:00:10 --:--:-- 22.2M
mgpu@mgpu-VirtualBox:~/todo-app$
```

```
mgpu@mgpu-VirtualBox:~/todo-app$ sudo install minikube-linux-amd64 /usr/local/bin/minikube
[sudo] password for mgpu:
mgpu@mgpu-VirtualBox:~/todo-app$
```

11.Добавление пользователя в группу Docker:

```
mgpu@mgpu-VirtualBox:~/todo-app$ sudo usermod -aG docker $USER && newgrp docker
mgpu@mgpu-VirtualBox:~/todo-app$
```

12. Установка kubectl – инструмента для управления кластерами Kubernetes:

```
mgpu@mgpu-VirtualBox:~/todo-app$ sudo snap install kubectl --classic
snap "kubectl" is already installed, see 'snap help refresh'
mgpu@mgpu-VirtualBox:~/todo-app$
```

13. Запуск Minikube с ограничением выделенной памяти:

```
mgpu@mgpu-VirtualBox:~/todo-app$ minikube start --memory=2048mb --driver=docker
😄  minikube v1.35.0 on Ubuntu 20.04 (vbox/amd64)
✨  Using the docker driver based on user configuration
📌  Using Docker driver with root privileges
👍  Starting "minikube" primary control-plane node in "minikube" cluster
🚜  Pulling base image v0.0.46 ...
🔥  Creating docker container (CPUs=2, Memory=2048MB) ...
🐳  Preparing Kubernetes v1.32.0 on Docker 27.4.1 ...
    ▪ 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: storage-provisioner, default-storageclass
🏄  Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
mgpu@mgpu-VirtualBox:~/todo-app$
```

14. Настройка окружения командной строки Ubuntu для работы с Docker, который управляется Minikube:

```
mgpu@mgpu-VirtualBox:~/todo-app$ eval $(minikube docker-env)
mgpu@mgpu-VirtualBox:~/todo-app$
```

15. Билд докер-контейнеров:

```
mgpu@mgpu-VirtualBox:~/Downloads/4.1/todo-app$ docker build -t todo-frontend ./frontend
[+] Building 262.1s (14/14) FINISHED                                                docker:default
 => [internal] load build definition from Dockerfile                                         0.0s
 => => transferring dockerfile: 260B                                                         0.0s
 => [internal] load metadata for docker.io/library/nginx:alpine                             13.4s
 => [internal] load metadata for docker.io/library/node:16                                  23.9s
 => [internal] load .dockerignore                                                            0.1s
 => => transferring context: 2B                                                              0.0s

mgpu@mgpu-VirtualBox:~/Downloads/4.1/todo-app$ docker build -t todo-backend ./backend
[+] Building 15.0s (10/10) FINISHED                                                 docker:default
 => [internal] load build definition from Dockerfile                                         0.1s
 => => transferring dockerfile: 152B                                                         0.0s
 => [internal] load metadata for docker.io/library/node:16                                   3.0s
 => [internal] load .dockerignore                                                            0.1s
 => => transferring context: 2B                                                              0.0s
 => [1/5] FROM docker.io/library/node:16@sha256:f77a1aef2da8d83e45ec990f45df50f1a286c5fe8bbfb8c6e4246c  0.0s
 => [internal] load build context                                                            2.4s
 => => transferring context: 13.29MB                                                         2.4s
 => CACHED [2/5] WORKDIR /app                                                                0.0s
 => [3/5] COPY package*.json ./                                                              0.3s
 => [4/5] RUN npm install                                                                    7.8s
 => [5/5] COPY . .                                                                           0.5s
 => exporting to image                                                                       0.5s
 => => exporting layers                                                                      0.5s
 => => writing image sha256:52c5a07e4ba1f153685afdab9848c75019a2c56940269ba8496d09b30f743f16  0.0s
 => => naming to docker.io/library/todo-backend                                              0.0s
mgpu@mgpu-VirtualBox:~/Downloads/4.1/todo-app$
```

16. Создание и применение конфигураций (манифестов):

```
mgpu@mgpu-VirtualBox:~/todo-app$ kubectl create -f k8s/
deployment.apps/backend-deployment created
service/backend-service created
deployment.apps/frontend-deployment created
service/frontend-service created
configmap/mongodb-config created
deployment.apps/mongodb-deployment created
persistentvolumeclaim/mongodb-pvc created
secret/mongodb-secret created
service/mongodb-service created
mgpu@mgpu-VirtualBox:~/todo-app$
```

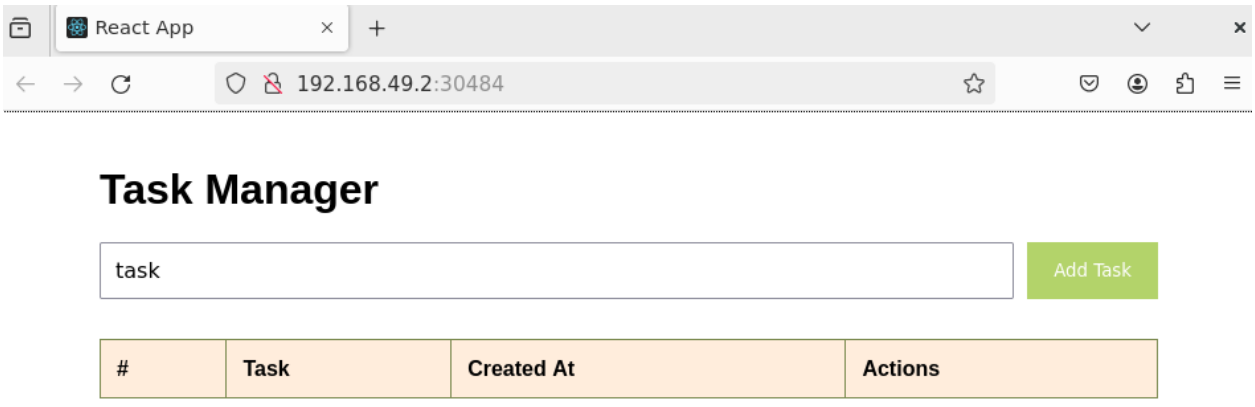
17. Проверка успешности запуска всех подов:

```
mgpu@mgpu-VirtualBox:~/todo-app$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
backend-deployment-5bb99bf754-xkjzt   1/1     Running   0          27s
frontend-deployment-95ddc88f7-rwlzk   1/1     Running   0          62s
mongodb-deployment-6b5b8856b4-zpdxg   1/1     Running   0          34m
mgpu@mgpu-VirtualBox:~/todo-app$
```

18. Вывод адреса, на котором доступен сервер приложения:

```
mgpu@mgpu-VirtualBox:~/todo-app$ minikube service frontend-service --url
http://192.168.49.2:30468
mgpu@mgpu-VirtualBox:~/todo-app$
```

19. Проверка доступности сервиса:



(Далее, к сожалению, возникает ошибка при добавлении задачи, связанная с неточностью написания кода)

ВЫВОДЫ

В ходе выполнения лабораторной работы были реализованы почти все задачи, а именно:

1. Создано фронтенд-приложение на React и бэкенд-приложение на Node.js с использованием Express и MongoDB;
2. Контейнеризированы фронтенд и бэкенд приложения с помощью Docker;
3. Созданы манифесты Kubernetes для развертывания приложения;
4. Развернуто приложение в кластере Kubernetes.

Таким образом, была достигнута главная цель работы - применить полученные знания по созданию и развертыванию полнофункционального приложения с использованием Docker и Kubernetes.

От себя лично хочется сказать, что Kubernetes – это трудно. И в процессе выполнения работ по нему, был навсегда усвоен урок: без реально обоснованной целесообразности Kubernetes лучше даже не упоминать.