

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

Выполнила: st_95

Лабораторная работа 4_1

Создание и развертывание полнофункционального приложения

Интеграция и развертывание программного обеспечения с помощью
контейнеров

Направление подготовки

38.03.05 Бизнес-информатика

Профиль подготовки

Аналитика данных и эффективное управление

Москва

2025

Вариант 8. Создайте приложение "Форум" с использованием Angular, Node.js, Express и MongoDB.

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

Задачи:

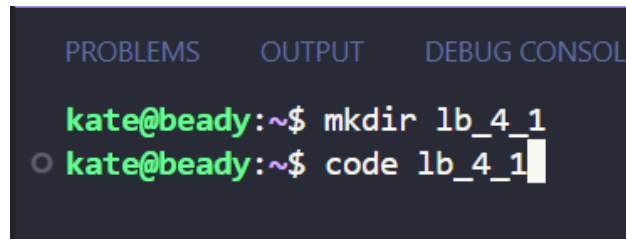
1. Создать фронтенд-приложение на Angular.
2. Создать бэкенд-приложение на Node.js с использованием Express и MongoDB.
3. Контейнеризировать фронтенд и бэкенд приложения с помощью Docker.
4. Создать манифесты Kubernetes для развертывания приложения.
5. Развернуть приложение в кластере Kubernetes.
6.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

Архитектура приложения:

- Frontend: Angular (отображение списка тем, создание тем, комментарии).
- Backend: Node.js + Express (REST API для работы с темами и комментариями).
- Database: MongoDB (хранение тем и комментариев).
- Infrastructure: Docker + Kubernetes (развертывание).

Ход работы

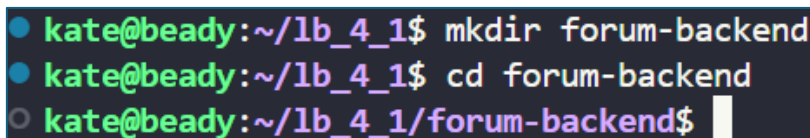
1. Инициализация проекта и настройка зависимостей
2. Создание директории проекта (Рисунок 1).



```
PROBLEMS OUTPUT DEBUG CONSOLE
kate@beady:~$ mkdir lb_4_1
kate@beady:~$ code lb_4_1
```

Рисунок 1

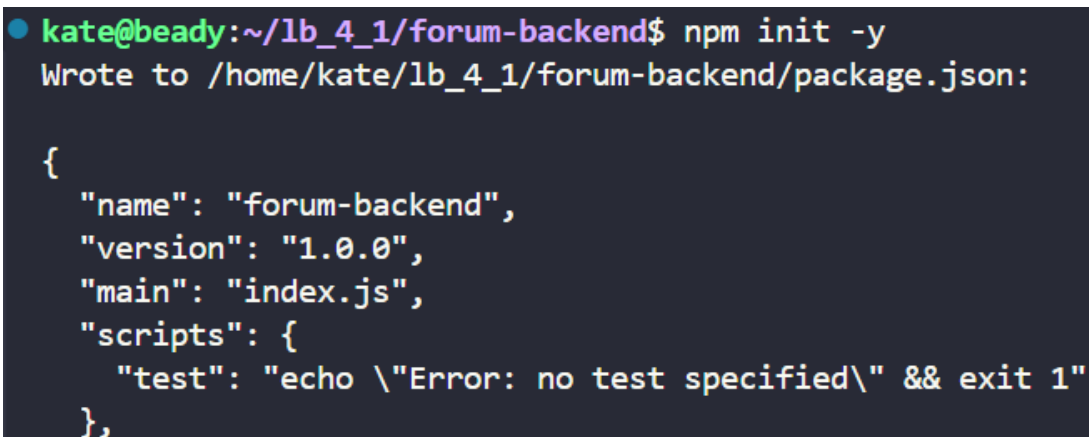
- 2.1. Создание директории бекенда (Рисунок 2).



```
kate@beady:~/lb_4_1$ mkdir forum-backend
kate@beady:~/lb_4_1$ cd forum-backend
kate@beady:~/lb_4_1/forum-backend$
```

Рисунок 2

- 2.2. Инициализация проекта Node.js (Рисунок 3).

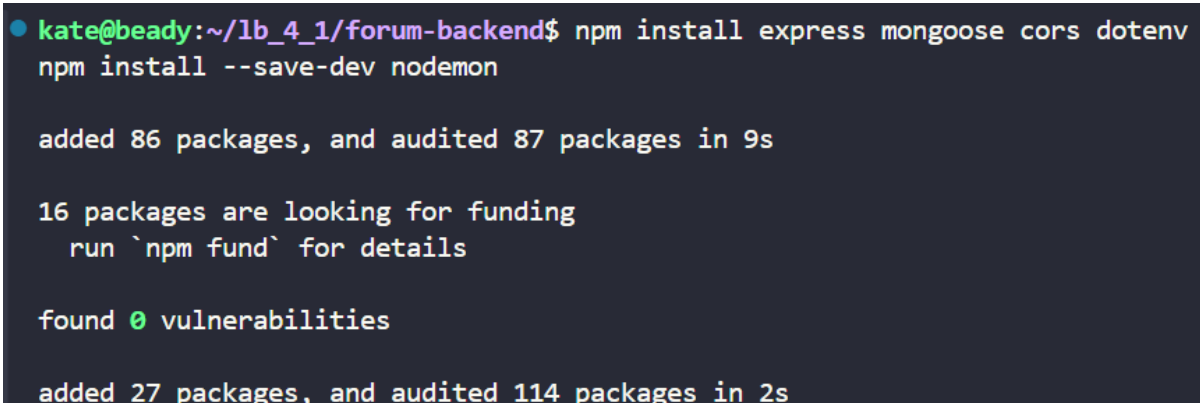


```
kate@beady:~/lb_4_1/forum-backend$ npm init -y
Wrote to /home/kate/lb_4_1/forum-backend/package.json:

{
  "name": "forum-backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
}
```

Рисунок 3

- 2.3. Установка зависимостей (Рисунок 4).



```
kate@beady:~/lb_4_1/forum-backend$ npm install express mongoose cors dotenv
npm install --save-dev nodemon

added 86 packages, and audited 87 packages in 9s

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

found 0 vulnerabilities

added 27 packages, and audited 114 packages in 2s
```

Рисунок 4

3. Настройка Express и MongoDB

3.1. Создание файла app.js (Рисунок 5).

```
forum-backend > src > JS app.js > ...
1  const express = require('express');
2  const mongoose = require('mongoose');
3  const cors = require('cors');
4  require('dotenv').config();
5
6  const app = express();
7
8  // Middleware
9  app.use(cors());
10 app.use(express.json());
11
12 // Подключение к MongoDB
13 mongoose.connect(process.env.MONGODB_URI, {
14   useNewUrlParser: true,
15   useUnifiedTopology: true
16 })
17 .then(() => console.log('Connected to MongoDB'))
18 .catch(err => console.error('MongoDB connection error:', err));
19
20 // Тестовый роут
21 app.get('/', (req, res) => {
22   res.send('Forum Backend is running!');
23 });
24
25 // Роуты API
26 const postsRouter = require('./routes/posts');
27 app.use('/api/posts', postsRouter);
28
29 module.exports = app;
```

Рисунок 5

3.2. Создание файла server.js (Рисунок 6).

```
forum-backend > src > JS server.js > ...
1  const app = require('./app');
2  const PORT = process.env.PORT || 5000;
3
4  app.listen(PORT, () => {
5    console.log(`Server is running on port ${PORT}`);
6  });
```

Рисунок 6

4. Создание моделей и роутов для постов

4.1. Создание модели Post.js (Рисунок 7).

```

forum-backend > src > models > JS Post.js > ...
1  const mongoose = require('mongoose');
2
3  const PostSchema = new mongoose.Schema({
4    title: { type: String, required: true },
5    content: { type: String, required: true },
6    author: { type: String, default: 'Anonymous' },
7    createdAt: { type: Date, default: Date.now }
8  });
9
10 module.exports = mongoose.model('Post', PostSchema);

```

Рисунок 7

4.2. Создание роута post.js (Рисунок 8).

```

forum-backend > src > routes > JS posts.js > ...
1  const express = require('express');
2  const router = express.Router();
3  const Post = require('../models/Post');
4
5  // Получить все посты
6  router.get('/', async (req, res) => {
7    try {
8      const posts = await Post.find().sort({ createdAt: -1 });
9      res.json(posts);
10     } catch (err) {
11       res.status(500).json({ message: err.message });
12     }
13   });
14
15  // Создать новый пост
16  router.post('/', async (req, res) => {
17    const post = new Post({
18      title: req.body.title,
19      content: req.body.content,
20      author: req.body.author
21    });
22
23    try {
24      const newPost = await post.save();
25      res.status(201).json(newPost);
26    } catch (err) {
27      res.status(400).json({ message: err.message });
28    }
29   });
30
31  module.exports = router;

```

Рисунок 8

5. Настройка package.json (Рисунок 9).

```

forum-backend > {} package.json > ...
1   {
2     "name": "forum-backend",
3     "version": "1.0.0",
4     "main": "index.js",
5     ▷ Debug
6     "scripts": {
7       "start": "node src/server.js",
8       "dev": "nodemon src/server.js"
9     },
10    "keywords": [],
11    "author": "",
12    "license": "ISC",
13    "description": "",
14    "dependencies": {
15      "cors": "^2.8.5",
16      "dotenv": "^16.5.0",
17      "express": "^5.1.0",
18      "mongoose": "^8.13.2"
19    },
20    "devDependencies": {
21      "nodemon": "^3.1.9"
22    }
23  }

```

Рисунок 9

6. Установка MongoDB

6.1. Установка MongoDB с помощью Docker (Рисунок 10).

```

● kate@beady:~/lb_4_1$ sudo docker run --name mongodb -d -p 27017:27017 mongo:latest
Unable to find image 'mongo:latest' locally
latest: Pulling from library/mongo
66eecf3d03b3: Download complete
8fc21ef9255d: Download complete
adf7a2b7606d: Download complete
2726e237d1a3: Download complete
1a08ed22982f: Download complete
29fee425ac1e: Download complete
90ade5227b11: Download complete
7e4d144a0672: Download complete

```

Рисунок 10

6.2. Проверка контейнера (Рисунок 11)

```
kate@beady:~/lb_4_1$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
21d70d210a40	mongo:latest	"docker-entrypoint.s..."	19 seconds ago	Up 16 seconds	0.0.0.0:27017
->27017/tcp	mongodb				

Рисунок 11

```
forum-backend > .env
1 MONGODB_URI=mongodb://localhost:27017/forum
```

Рисунок 12

6.3. Проверка подключения (Рисунок 13)

```
test> db.test.insertOne({ message: "Hello from Docker!" })
{
  acknowledged: true,
  insertedId: ObjectId('67f9b044515a88deb3d861e0')
}
test> db.test.find()
[
  {
    _id: ObjectId('67f9b044515a88deb3d861e0'),
    message: 'Hello from Docker!'
  }
]
```

Рисунок 13

7. Запуск и тестирование

7.1. Запуск (Рисунок 14).

```
○ kate@beady:~/lb_4_1/forum-backend$ npm run dev

> forum-backend@1.0.0 dev
> nodemon src/server.js

[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node src/server.js`
```

Рисунок 14

7.2. Проверка работоспособности (Рисунок 15)

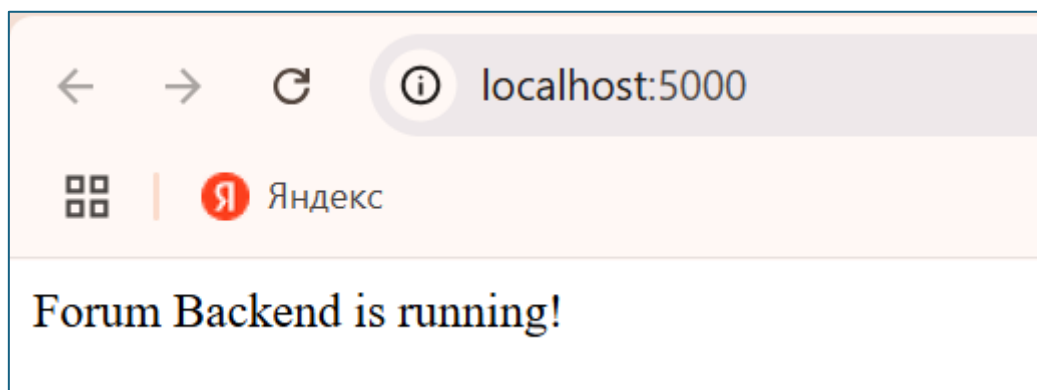
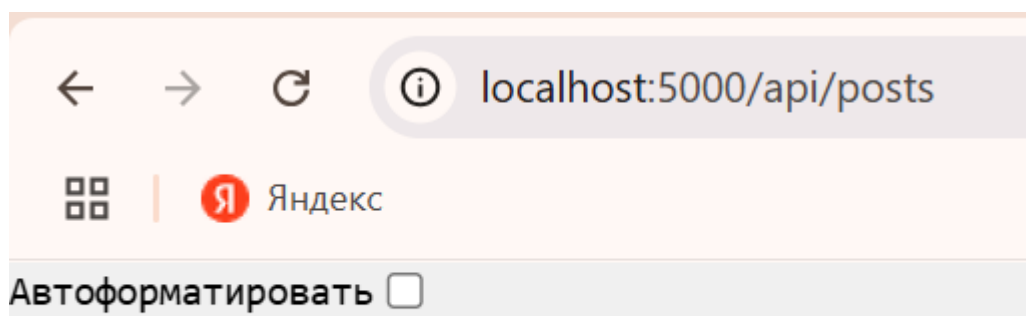


Рисунок 15

7.3. Проверка GET – запроса (Рисунок 16).



[]

Рисунок 16

7.4. Проверка PUT запроса (Рисунок 17)


```

kate@beady:~/lb_4_1/forum-backend$ curl -X POST -H "Content-Type: application/json" -d '{"title": "Первое сообщение", "content": "Привет, это тест форума!", "author": "Аноним"}' http://localhost:5000/api/posts
{"title": "Первое сообщение", "content": "Привет, это тест форума!", "author": "Аноним", "_id": "67f9b1ec99b73f03b5a4e4ff", "createdAt": "2025-04-12T00:21:00.369Z", "__v": 0}kate@beady:~/lb_4_1/forum-backend$

```

Рисунок 17

Проверка получения данных через браузер (Рисунок 18).

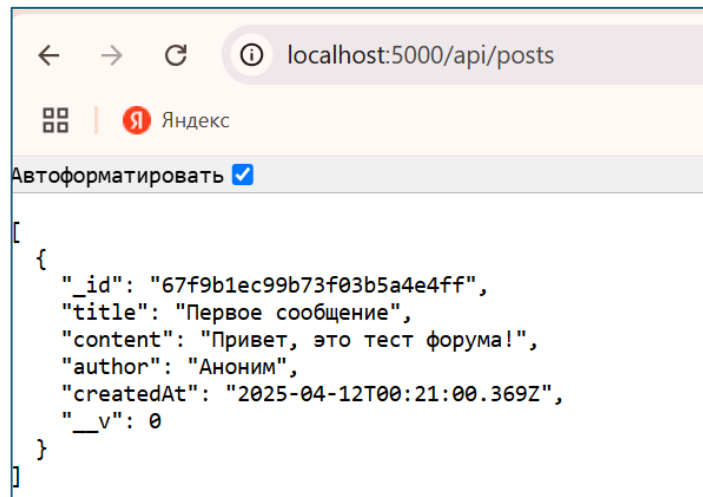


Рисунок 18

8. Docker – образ для приложения (Рисунок 19).

```

forum-backend > Dockerfile > ...
1  # Используем официальный образ Node.js
2  FROM node:18-alpine
3
4  # Создаем рабочую директорию
5  WORKDIR /app
6
7  # Копируем package.json и package-lock.json
8  COPY package*.json ./
9
10 # Устанавливаем зависимости
11 RUN npm install --production
12
13 # Копируем исходный код
14 COPY . .
15
16 # Открываем порт 5000
17 EXPOSE 5000
18
19 # Запускаем приложение
20 CMD ["node", "src/server.js"]

```

Рисунок 19

9. `.dockerignore` (чтобы исключить ненужные файлы) (Рисунок 20)

```
forum-backend > .dockerignore
1  node_modules
2  npm-debug.log
3  .env
4  .git
5  .gitignore
```

Рисунок 20

10. `Docker-compose.yml` для локального тестирования (Рисунок 21)

```
docker-compose.yml > ...
1  version: '3.8'
2
3  >Run All Services
4  services:
5    >Run Service
6    backend:
7      build: .
8      ports:
9        - "5000:5000"
10     environment:
11       - MONGODB_URI=mongodb://mongo:27017/forum
12     depends_on:
13       - mongo
14
15   >Run Service
16   mongo:
17     image: mongo:6
18     volumes:
19       - mongodb_data:/data/db
20     ports:
21       - "27017:27017"
22
23 volumes:
24   mongodb_data:
```

Рисунок 21

11. Сборка контейнера (Рисунок 22)

```
kate@beady:~/lb_4_1/forum-backend$ docker build -t forum-backend .
[+] Building 1.2s (10/10) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 522B                               0.0s
=> [internal] load metadata for docker.io/library/node:18-alpine 0.7s
=> [internal] load .dockerignore                                  0.0s
=> => transferring context: 87B                                     0.0s
=> [internal] load build context                                  0.0s
=> => transferring context: 357B                                    0.0s
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d1189 0.1s
=> => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d1189 0.1s
=> CACHED [2/5] WORKDIR /app                                     0.0s
```

Рисунок 22

12. Запуск (Рисунок 23)

```

kate@beady:~/lb_4_1/forum-backend$ docker-compose up
WARN[0000] /home/kate/lb_4_1/forum-backend/docker-compose.yaml: the attribute `version` is obsolete,
it will be ignored, please remove it to avoid potential confusion
[+] Building 2.2s (11/11) FINISHED                                docker:default
=> [backend internal] load build definition from Dockerfile        0.0s
=> => transferring dockerfile: 522B                                0.0s
=> [backend internal] load metadata for docker.io/library/node:18-alpine 1.1s
=> [backend internal] load .dockerignore                          0.0s
=> => transferring context: 87B                                     0.0s
=> [backend 1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249 0.1s

```

Рисунок 23

13. Проверка в браузере и POST-запрос (Рисунок 24).

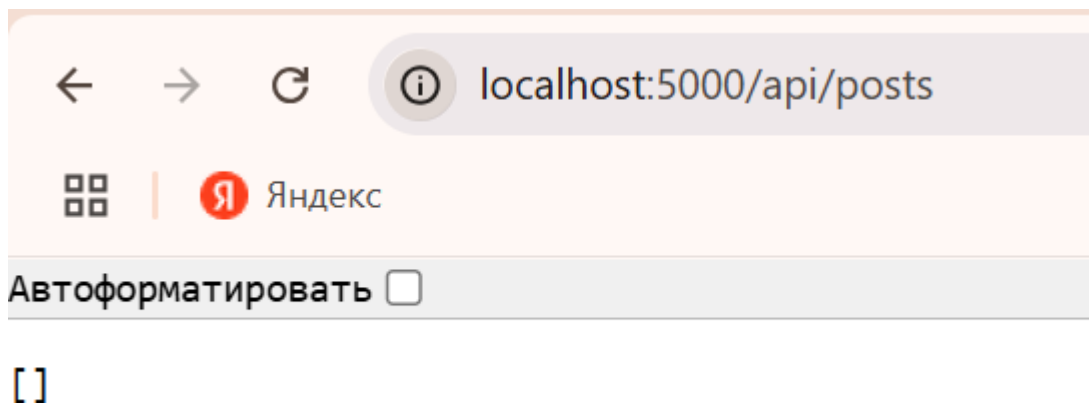


Рисунок 24

14. Post запрос :

echo '{"title":"Second Post","content":"My second post","author":"Katya"}'

> post.json

wget --post-file=post.json --header="Content-Type: application/json" \

<http://localhost:5000/api/posts>

```

/app # wget --post-file=post.json --header="Content-Type: application/json" \
> http://localhost:5000/api/posts
Connecting to localhost:5000 (:::1]:5000)
wget: can't open 'posts': File exists
/app #

```

The image shows a web browser window displaying the response of a POST request. The address bar shows 'localhost:5000/api/posts'. The page content displays a JSON array with two objects, each containing fields like '_id', 'title', 'content', 'author', 'createdAt', and 'v'.

15. Разворачивание приложения в k8s

15.1. Создание манифестов для mongo

15.1.1. statefulset.yaml (Рисунок 25).

```
k8s > mongo > ! statefulset.yaml
1  apiVersion: apps/v1
2  kind: StatefulSet
3  metadata:
4    name: mongo
5  spec:
6    serviceName: mongo
7    replicas: 1
8    selector:
9      matchLabels:
10       app: mongo
11   template:
12     metadata:
13       labels:
14         app: mongo
15     spec:
16       containers:
17         - name: mongo
18           image: mongo:6
19           ports:
20             - containerPort: 27017
21           volumeMounts:
22             - name: mongodb-data
23               mountPath: /data/db
24   volumeClaimTemplates:
25     - metadata:
26       name: mongodb-data
27     spec:
28       accessModes: [ "ReadWriteOnce" ]
```

Рисунок 25

15.1.2. service.yaml (Рисунок 26)

```

k8s > mongo > ! service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: mongo
5  spec:
6    selector:
7      app: mongo
8    ports:
9      - port: 27017
10     targetPort: 27017

```

Рисунок 26

15.1.3. deployment.yaml (Рисунок 27)

```

k8s > mongo > ! deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: forum-backend
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: forum-backend
10   template:
11     metadata:
12       labels:
13         app: forum-backend
14     spec:
15       containers:
16         - name: backend
17           image: forum-backend:latest
18           ports:
19             - containerPort: 5000
20       env:
21         - name: MONGODB_URI
22           value: "mongodb://mongo:27017/forum"
23         - name: PORT
24           value: "5000"

```

Рисунок 27

15.2. Создание манифестов для backend

15.2.1. service.yaml (Рисунок 28)

```

k8s > backend > ! service.yaml
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: forum-backend
5  spec:
6    selector:
7      app: forum-backend
8    ports:
9      - protocol: TCP
10      port: 80
11      targetPort: 5000
12    type: LoadBalancer

```

Рисунок 28

15.2.2. deployment.yaml (Рисунок 29)

```

k8s > backend > ! deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: forum-backend
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: forum-backend
10   template:
11     metadata:
12       labels:
13         app: forum-backend
14     spec:
15       containers:
16         - name: backend
17           image: forum-backend:latest
18           ports:
19             - containerPort: 5000
20           env:
21             - name: MONGODB_URI
22               value: "mongodb://mongo:27017/forum"
23             - name: PORT
24               value: "5000"

```

Рисунок 29

15.3. Сборка контейнера (Рисунок 30).

```

kate@beady:~/lb_4_1/forum-backend$ eval $(minikube docker-env)
kate@beady:~/lb_4_1/forum-backend$ docker build -t forum-backend:latest .
[+] Building 29.0s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 522B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> [auth] library/node:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 87B
=> [internal] load build context
=> => transferring context: 53.08kB
=> [1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
=> => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
=> => sha256:1e5a4c89cee5c0826c540ab06d4b6b491c96eda01837f430bd47f0d26702d6e3 1.26MB / 1.26MB

```

Рисунок 30

15.4. Применение манифестов (Рисунок 31).

```

• kate@beady:~/lb_4_1/k8s$ kubectl apply -f mongo/
deployment.apps/forum-backend created
service/mongo created
statefulset.apps/mongo created
• kate@beady:~/lb_4_1/k8s$ kubectl apply -f backend/
deployment.apps/forum-backend unchanged
service/forum-backend created
○ kate@beady:~/lb_4_1/k8s$

```

Рисунок 31

Проверка подов (Рисунок 32)

```

ⓧ kate@beady:~/lb_4_1/k8s/backend$ kubectl get pods -w
NAME                                READY   STATUS    RESTARTS   AGE
forum-backend-64c4d5868-mzfxm      1/1     Running   0           21s
forum-backend-64c4d5868-ptr7d      1/1     Running   0           24s
mongo-0                             1/1     Running   0           5m41s





```



Рисунок 32

```

/app # echo '{"title":"Second Post","content":"My second post","author":"Katya"}' > post.json
/app # wget --post-file=post.json --header="Content-Type: application/json" \
> http://localhost:5000/api/posts
Connecting to localhost:5000 (:::1):5000)
saving to 'posts'
posts          100% |*****| 147 0:00:00 ETA
'posts' saved

```

 127.0.0.1:34739/api/posts

 |  Яндекс

Автоформатировать ☐

```
[{"_id": "67f9e4cada003fa64c97aff5", "title": "Second Post", "content": "My second post", "author": "Katya", "createdAt": "2025-04-12T03:58:03.019Z", "__v": 0}]
```