

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

Макарова Екатерина Павловна

ЛАБОРАТОРНАЯ РАБОТА 3.1.

Docker Compose для мультиконтейнерных приложений

Интеграция и развертывание программного обеспечения с помощью
контейнеров

Направление подготовки

38.03.05 Бизнес-информатика

Профиль подготовки

Аналитика данных и эффективное управление

Курс обучения: 4

Форма обучения: очная

Преподаватель: кандидат технических наук,

доцент Босенко Тимур Муртазович

Москва

2025

Цель работы: освоить использование Docker Compose для управления многоконтейнерными приложениями.

Задачи:

1. Создать файл docker-compose.yml для указанного многоконтейнерного приложения.
2. Запустить приложение с помощью Docker Compose.
3. Проверить работоспособность приложения и взаимодействие между контейнерами.
4. Выполнить индивидуальное задание.

Вариант 8 (st_95):

1. Создать файл docker-compose.yml для системы управления проектами (Ruby on Rails + MySQL).
2. Запустить приложение и проверить создание задач.
3. Реализовать расчет процента выполнения проектов.



Постановка задач:

1. Генерирует синтетические данные о проектах
2. Сохраняет данные в MySQL
3. Sidekiq: выполняет фоновую задачу по расчёту процента выполнения проекта, Redis хранит очереди задач для Sidekiq.
4. Разворачивается через Docker Compose

Технический стек:

Backend: Ruby on Rails

База данных: MySQL

Фоновые задачи: Sidekiq + Redis

Контейнеризация: Docker + Docker Compose

Дерево проекта:

```
kate@beady:~/lb_3_1$ tree -L 1
.
├── Dockerfile
├── Gemfile
├── Gemfile.lock
├── README.md
├── Rakefile
├── app
├── bin
├── config
├── config.ru
├── db
├── docker-compose.yml
├── lib
├── log
├── public
├── storage
├── test
├── tmp
└── vendor
```

Функциональные требования:

1. Генерация данных:

Создание 100 записей о проектах

Случайный выбор статуса проекта из predetermined списка

Генерация случайной стоимости

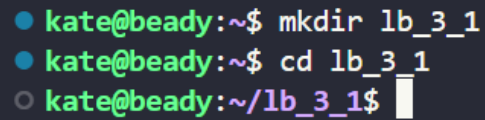
2. Хранение данных:

Сохранение всех транзакций

3. Обновление данных о прогрессе в соответствии с статусом проекта

Ход работы:

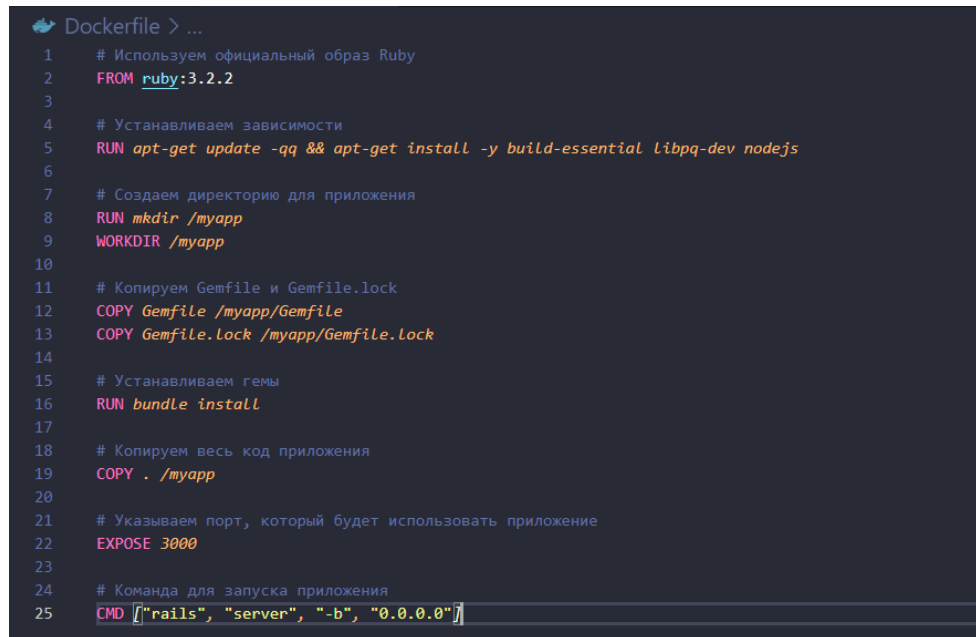
1. Создание директории и переход в неё (Рис. 1)



```
kate@beady:~$ mkdir lb_3_1
kate@beady:~$ cd lb_3_1
kate@beady:~/lb_3_1$
```

Рис. 1

2. Создание DockerFile (Рис. 2).



```
Dockerfile > ...
1  # Используем официальный образ Ruby
2  FROM ruby:3.2.2
3
4  # Устанавливаем зависимости
5  RUN apt-get update -qq && apt-get install -y build-essential libpq-dev nodejs
6
7  # Создаем директорию для приложения
8  RUN mkdir /myapp
9  WORKDIR /myapp
10
11 # Копируем Gemfile и Gemfile.lock
12 COPY Gemfile /myapp/Gemfile
13 COPY Gemfile.lock /myapp/Gemfile.lock
14
15 # Устанавливаем gems
16 RUN bundle install
17
18 # Копируем весь код приложения
19 COPY . /myapp
20
21 # Указываем порт, который будет использовать приложение
22 EXPOSE 3000
23
24 # Команда для запуска приложения
25 CMD ["rails", "server", "-b", "0.0.0.0"]
```

Рис. 2

3. Создание docker-compose.yml (Рис. 3).

```

docker-compose.yml
1  version: '3.8'
2
3  > Run All Services
services:
4  > Run Service
  db:
5    image: mysql:5.7
6    volumes:
7      - db_data:/var/lib/mysql
8    environment:
9      MYSQL_ROOT_PASSWORD: rootpassword
10     MYSQL_DATABASE: project_management_development
11     MYSQL_USER: user
12     MYSQL_PASSWORD: password
13    networks:
14      - app-network
15
16  > Run Service
  web:
17    build: .
18    command: bash -c "rm -f tmp/pids/server.pid && rails server -b 0.0.0.0"
19    volumes:
20      - ./myapp
21    ports:
22      - "3000:3000"
23    depends_on:
24      - db
25    environment:
26      DATABASE_HOST: db
27      DATABASE_USERNAME: user
28      DATABASE_PASSWORD: password
29    networks:
30      - app-network
31
32  volumes:
33    db_data:
34
35  networks:
36    app-network:
37    driver: bridge

```

Рис. 3

4. Создание Gemfile (Рис. 4).

```

Gemfile
1  source 'https://rubygems.org'
2
3  gem 'rails', '~> 7.0.4'
4  gem 'mysql2', '~> 0.5.4'
5

```

Рис. 4

5. Создание пустого файла Gemfile.lock (Рис. 5).

```

● kate@beady:~/lb_3_1$ touch Gemfile.lock
○ kate@beady:~/lb_3_1$

```

Рис. 5

6. Генерация Rails-приложения

1) Запуск контейнера web и генерирует приложение Rails (Рис. 6)

```

● kate@beady:~/lb_3_1$ docker-compose run web rails new . --force --database=mysql
WARN[0000] Found orphan containers ([lb_3_1-web-run-8b8abd40f342 lb_3_1-web-run-d2605df7ea7f lb_3_1-web-run-c1d8216396cf lb_3_1-web-run-c268268cdb75]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans flag to clean it up.
[+] Creating 1/1
✓ Container lb_3_1-db-1 Running 0.0s
  exist
  create README.md
  create Rakefile
  create .ruby-version
  create config.ru
  create .gitignore
  create .gitattributes
  force Gemfile
  run git init -b main from "."

```

Рис. 6

Успешная генерация дерева проекта (Рис. 7).

```

● kate@beady:~/lb_3_1$ tree -L 1
.
├── Dockerfile
├── Gemfile
├── Gemfile.lock
├── README.md
├── Rakefile
├── app
├── bin
├── config
├── config.ru
├── db
├── docker-compose.yml
├── lib
├── log
├── public
├── storage
├── test
├── tmp
└── vendor

```

Рис. 7

2) Настройка базы данных ().

```
config > ! database.yml
#
# And be sure to use new-style password hashing:
# https://dev.mysql.com/doc/refman/5.7/en/password-hashing.html
#
default: &default
  adapter: mysql2
  encoding: utf8mb4
  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
  username: user
  password: password
  host: db

development:
  <<: *default
  database: myapp_development

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
  <<: *default
  database: myapp_test
```

Рис. 8

3) Запуск контейнера (Рис. 9, Рис. 10).

```
kate@beady:~/lb_3_1$ docker compose up
[+] Running 3/3
 ✓ Network lb_3_1_app-network Created 0.5s
 ✓ Container lb_3_1-db-1 Created 0.3s
 ✓ Container lb_3_1-web-1 Created 0.3s
Attaching to db-1, web-1
db-1 | 2025-03-15 19:32:03+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.44-1.e17 started.
db-1 | 2025-03-15 19:32:04+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
```

Рис. 9

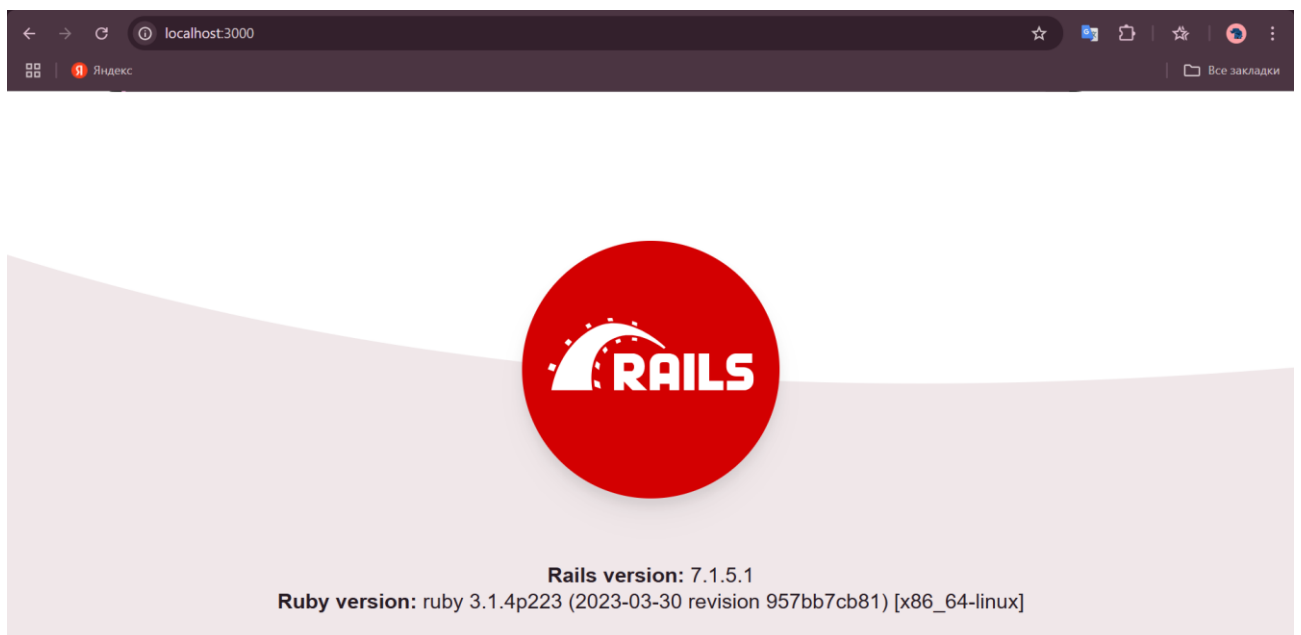


Рис. 10

7. Модель для генерации данных о проектах

1) Создание модели Project, которая генерирует таблицу (Рис. 11).

```
kate@beady:~/lb_3_1$ docker-compose run web rails generate model Project name:string status:string cost:integer progress:integer
WARN[0000] Found orphan containers ([lb_3_1-web-run-6425f9779e9b lb_3_1-web-run-1c4f78ee0cbc lb_3_1-web-run-2cfb159f63d7 lb_3_1-
-web-run-d695dfbc868b lb_3_1-web-run-b5ba33ac8291 lb_3_1-web-run-81d873355f26 lb_3_1-web-run-8441beb879f7 lb_3_1-web-run-091934
02604e lb_3_1-web-run-fd0ceefa484e]) for this project. If you removed or renamed this service in your compose file, you can run
this command with the --remove-orphans flag to clean it up.
[+] Creating 1/1
✓ Container lb_3_1-db-1 Created 0.0s
[+] Running 1/1
✓ Container lb_3_1-db-1 Started 0.4s
  invoke active_record
The name 'Project' is either already used in your application or reserved by Ruby on Rails. Please choose an alternative or use
--skip-collision-check or --force to skip this check and run this generator again.
```

Рис. 11

2) Запуск миграции (Рис. 12).

```
kate@beady:~/lb_3_1$ docker-compose run web rails db:migrate
WARN[0000] Found orphan containers ([lb_3_1-web-run-0eddd3cf98b2 lb_3_1-web-run-6425f9779e9b lb_3_1-web-run-1c4f78ee0cbc lb_3_1-
-web-run-2cfb159f63d7 lb_3_1-web-run-d695dfbc868b lb_3_1-web-run-b5ba33ac8291 lb_3_1-web-run-81d873355f26 lb_3_1-web-run-8441be
b879f7 lb_3_1-web-run-09193402604e lb_3_1-web-run-fd0ceefa484e]) for this project. If you removed or renamed this service in yo
ur compose file, you can run this command with the --remove-orphans flag to clean it up.
[+] Creating 1/1
✓ Container lb_3_1-db-1 Running 0.0s
== 20250315202253 CreateProjects: migrating =====
-- create_table(:projects)
   -> 0.0549s
== 20250315202253 CreateProjects: migrated (0.0553s) =====
```

Рис. 12

3) Проверка созданной таблицы в MySQL (Рис. 13).

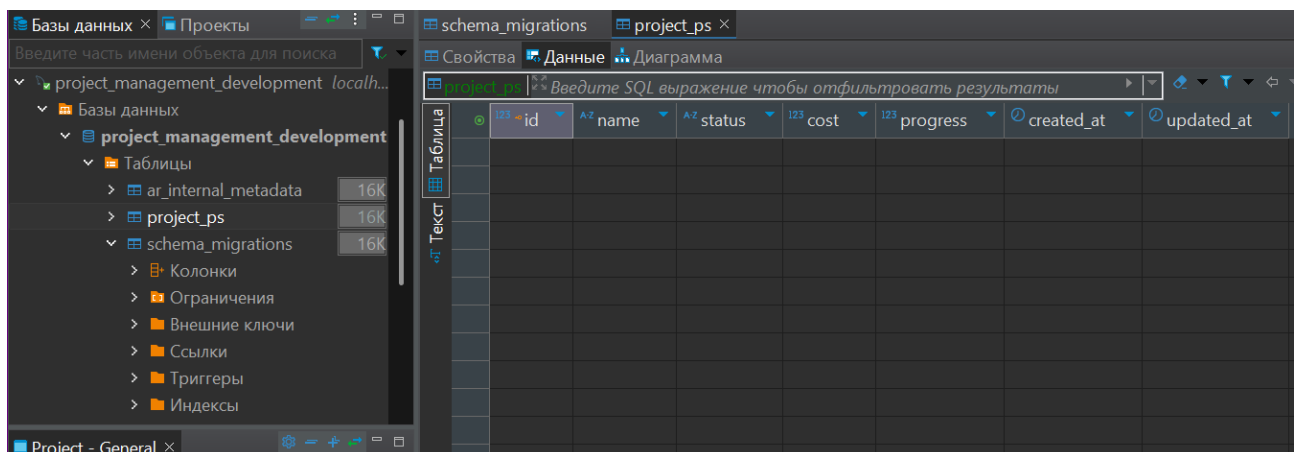


Рис. 13

4) Создание задачи для генерации данных (Рис. 14).

```
kate@beady:~/lb_3_1$ docker-compose exec web rails generate task projects generate
create lib/tasks/projects.rake
```

Рис. 14

5) Добавление гемов faker и sidekiq в Gemfile (Рис. 15).

```
Gemfile
28 # Build JSON APIs with ease [https://github.com/rails/jbuilder]
29 gem "jbuilder", "~> 2.13.0"
30
31 gem 'sidekiq'
32
33 gem 'faker'
```

Рис. 15

Пересборка контейнера после изменения Gemfile (Рис. 16).

```
kate@beady:~/lb_3_1$ docker-compose run --user root web bundle install
WARN[0003] Found orphan containers ([lb_3_1-web-run-08fb7d37fd84 lb_3_1-web-run-9cfeb54b7461 lb_3_1-web-run-82c67a68c289]) for
this project. If you removed or renamed this service in your compose file, you can run this command with the --remove-orphans f
lag to clean it up.
[+] Creating 1/1
✓ Container lb_3_1-db-1 Running 0.0s
Fetching gem metadata from https://rubygems.org/.....
Resolving dependencies...
Using rake 13.2.1
WARN: Unresolved or ambiguous specs during Gem::Specification.reset:
      stringio (>= 0)
      Available/installed versions of this gem:
      - 3.1.5
      - 3.0.1
WARN: Clearing out unresolved specs. Try 'gem cleanup <gem>'
Please report a bug if this causes problems.
Using benchmark 0.4.0
Using bigdecimal 3.1.9
Using minitest 5.25.5
Using connection_pool 2.5.0
Using drb 2.2.1
Using logger 1.6.6
Using base64 0.2.0
Using mutex m 0.3.0
```

Рис. 16

6) Листинг кода task “project” (Рис. 17).

```
lib > tasks > projects.rake
1 namespace :db do
2   desc "Generate project data using Faker"
3   task generate_project_data: :environment do
4     require 'faker'
5
6     # Количество записей для генерации
7     num_projects = 100
8
9     num_projects.times do
10      Project.create!(
11        name: Faker::Company.name,
12        status: ["active", "inactive", "pending"].sample,
13        cost: Faker::Number.number(digits: 5),
14        progress: Faker::Number.between(from: 0, to: 100),
15        created_at: Faker::Time.backward(days: 365),
16        updated_at: Faker::Time.backward(days: 365)
17      )
18    end
19
20    puts "Generated #{num_projects} project records."
21  end
22 end
```

Рис. 17

Выполнение задачи (Рис. 18).

```
kate@beady:~/lb_3_1$ docker-compose exec web bash
rails@907c7f52af40:/rails$ rails db:generate_project_data
Generated 100 project records.
rails@907c7f52af40:/rails$
```

Рис. 18

Проверка данных в таблице *project* (Рис. 19).

	id	name	status	cost	progress	created_at	updated_at
1	1	Reichel and Sons	pending	67 896	84	2024-09-30 12:38:37	2025-01-07 20:20:00
2	2	Haley and Sons	inactive	24 916	30	2024-06-13 04:52:07	2024-11-10 06:33:31
3	3	Hoeger, Heidenreich and MacGyver	active	56 592	91	2025-01-21 18:39:10	2024-08-31 10:45:48
4	4	Borer-Denesik	inactive	86 031	37	2024-12-08 09:30:08	2024-06-21 08:17:21
5	5	Sanford-Kutch	pending	95 146	18	2025-01-23 17:47:06	2025-03-03 07:36:48
6	6	Haag LLC	pending	10 590	84	2024-08-30 09:19:50	2024-04-04 05:05:38
7	7	Jakubowski LLC	active	27 031	2	2024-05-26 09:30:23	2025-02-11 10:03:59
8	8	Abshire-Monahan	active	56 060	92	2024-10-24 11:44:11	2025-02-21 10:06:43
9	9	Mertz, Schoen and Brakus	pending	27 819	25	2024-04-09 19:39:00	2024-03-18 18:36:11
10	10	Conn-Russel	pending	74 935	8	2024-08-27 15:02:03	2024-12-25 04:02:00
11	11	Cummerata, Schiller and Gleichner	active	26 365	66	2024-10-16 19:30:55	2024-10-24 07:46:01
12	12	Mante Inc	pending	73 389	25	2024-07-02 13:46:25	2024-09-20 18:16:44
13	13	Gislason-Stanton	active	80 350	98	2024-07-16 23:11:52	2024-03-16 02:12:39
14	14	Armstrong-Bayer	active	46 243	59	2025-01-05 19:47:32	2024-06-20 18:55:03
15	15	Sipes Inc	active	37 782	31	2024-05-18 05:42:59	2024-10-27 09:44:59
16	16	Klocko, Marquardt and Beahan	pending	63 977	1	2024-06-27 06:41:06	2024-08-18 13:55:05
17	17	Nienow-Kautzer	active	46 191	38	2024-08-14 04:21:41	2024-09-07 07:34:53
18	18	Osinski-Hyatt	active	10 544	79	2025-01-04 22:38:49	2025-01-15 02:51:06
19	19	Dooley and Sons	inactive	33 922	21	2024-06-07 15:55:09	2024-06-22 13:15:17
20	20	Christiansen, O'Kon and Kertzmann	active	63 923	86	2024-05-25 22:50:05	2025-01-15 12:32:15
21	21	Kuhlman, Strosin and Connelly	inactive	30 306	17	2024-08-02 15:35:26	2024-07-27 01:58:15
22	22	Feeney, Runte and Renner	active	44 350	70	2024-04-06 10:21:06	2024-07-02 17:37:08
23	23	Predovic, Greenholt and Harber	inactive	37 936	39	2024-11-02 18:18:41	2024-09-03 06:32:02

Рис. 19

Индивидуальное задание «Расчёт процента выполнения»:

Так как данные были сгенерированы случайно, реализуем логику, по которой прогресс будет проставляться в зависимости от статуса проекта:

- если статус проекта «active», то прогресс будет увеличиваться на случайное число, но не выходя за 100;
- если статус проекта «inactive» (т.е. завершён), проставляем процента 100.

1) Создание job для вычисления процента, исходя из условия (Рис. 20).

```

app > jobs > update_project_progress_job.rb
1  class UpdateProjectProgressJob
2      include Sidekiq::Worker
3
4      def perform
5          ProjectPs.all.each do |project|
6              if project.status == "inactive"
7                  # Если статус "inactive", проставляем прогресс 100
8                  project.update(progress: 100)
9              elsif project.status == "active"
10                 # Если статус "active", увеличиваем прогресс на случайное значение
11                 new_progress = project.progress + rand(1..10)
12                 new_progress = 100 if new_progress > 100 # Не превышаем 100%
13                 project.update(progress: new_progress)
14             end
15         end
16     end
17 end

```

Рис. 20

Для проверки выполнения возьмём текущие данные до обновления, например, id = 2 и 8 (Рис. 21).

projects * <project_management_development> Script x

```

select *
from projects p
where id in (2, 8);

```

projects 1 projects 2 projects 3 x

select * from projects p where id in (2, 8); Введите SQL выражение чтобы отфильтровать результаты

	id	name	status	cost	progress	created_at	updated_at
1	2	Haley and Sc	inactive	24 916	30	4-06-13 04:52:07	24-11-10 06:33:31
2	8	Abshire-Mon	active	56 060	92	4-10-24 11:44:11	25-02-21 10:06:43

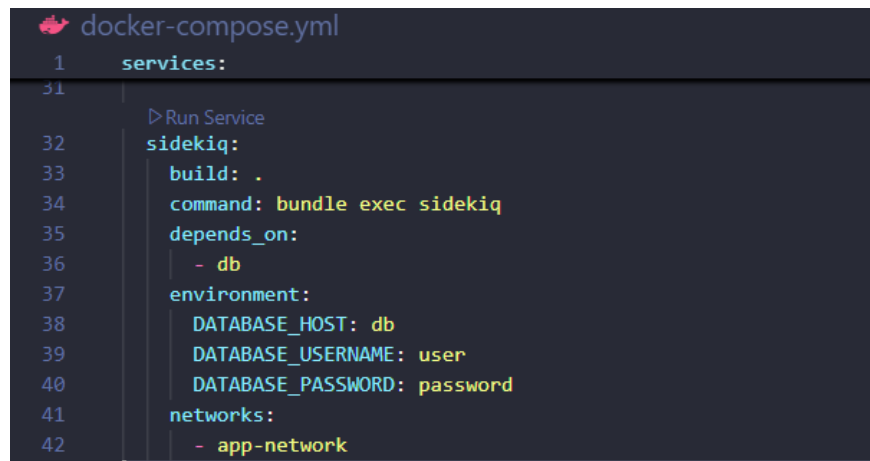
Рис. 21

Чтобы данные обновлялись периодически, используем sidekiq, добавим в gemfile и пересоберём контейнер (Рис. 22).

```
gem 'sidekiq-cron'
```

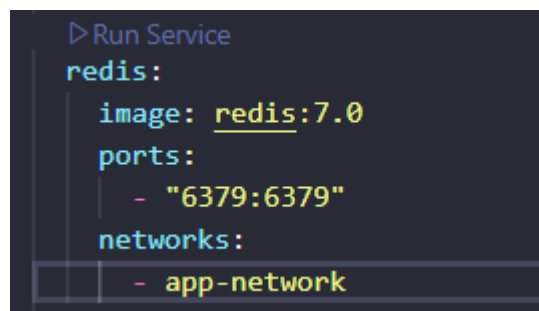
Рис. 22

Добавление sidekiq в docker-compose (Рис. 23) и redis (Рис. 24).



```
1  services:
31
32    > Run Service
33    sidekiq:
34      build: .
35      command: bundle exec sidekiq
36      depends_on:
37        - db
38      environment:
39        DATABASE_HOST: db
40        DATABASE_USERNAME: user
41        DATABASE_PASSWORD: password
42      networks:
43        - app-network
```

Рис. 23



```
> Run Service
redis:
  image: redis:7.0
  ports:
    - "6379:6379"
  networks:
    - app-network
```

Рис. 24

Добавление зависимостей (Рис. 25).

```

> Run Service
web:
  build: .
  command: bash -c "rm -f tmp/pids/server.pid && rails server -b 0.0.0.0"
  volumes:
    - ./rails
  ports:
    - "3000:3000"
  depends_on:
    - db
    - redis
  environment:
    DATABASE_HOST: db
    DATABASE_USERNAME: user
    DATABASE_PASSWORD: password
    REDIS_URL: redis://redis:6379/1
  networks:
    - app-network

> Run Service
sidekiq:
  build: .
  command: bundle exec sidekiq
  depends_on:
    - db
    - redis
  environment:
    DATABASE_HOST: db
    DATABASE_USERNAME: user
    DATABASE_PASSWORD: password
    REDIS_URL: redis://redis:6379/1
  networks:
    - app-network

```

Рис. 25

Настройка выполнения Job с помощью периодической задачи sidekiq (Рис. 26).

```

config > ! sidekiq.yml
1  :schedule:
2    update_project_progress:
3      cron: '0 * * * *' # Каждую минуту
4      class: 'UpdateProjectProgressJob'
5      queue: default
6
7  :redis:
8    :url: <%= ENV['REDIS_URL'] || 'redis://localhost:6379/1' %>

```

Рис. 26

Запуск контейнера (Рис. 27).

```
kate@beady:~/lb_3_1$ docker-compose up
[+] Running 9/9
✓ redis Pulled
✓ 73586727588c Download complete
✓ bcec16c47e35 Download complete
✓ b59068c7715f Download complete
✓ 96796a2514ff Download complete
✓ 4f4fb700ef54 Already exists
✓ fcf1b4feeffd Download complete
✓ b8c430d0189b Download complete
✓ a2318d6c47ec Download complete
[+] Running 5/5
✓ Network lb_3_1_app-network Created
✓ Container lb_3_1-db-1 Created
✓ Container lb_3_1-redis-1 Created
✓ Container lb_3_1-sidekiq-1 Created
✓ Container lb_3_1-web-1 Created
Attaching to db-1, redis-1, sidekiq-1, web-1
```

Рис. 27

И просмотр логов (Рис. 28), Job выполнена.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
web-1 | * Min threads: 5
web-1 | * Max threads: 5
web-1 | * Environment: development
web-1 | * PID: 1
web-1 | * Listening on http://0.0.0.0:3000
web-1 | Use Ctrl-C to stop
sidekiq-1 | 2025-03-16T21:30:25.806Z pid=1 tid=cgd class=UpdateProjectProgressJob jid=30c2cd12cce309d9dda5aec INFO
: start
sidekiq-1 | 2025-03-16T21:30:27.185Z pid=1 tid=cgd class=UpdateProjectProgressJob jid=30c2cd12cce309d9dda5aec elapsed=1.359 INFO: done
[]
View in Docker Desktop View Config Enable Watch
```

Рис. 28

Просмотр данных в таблице (Рис. 29).

projects `*<project_management_development> Script`

```
select *
from projects p
where id in (2, 8);
```

projects 1 projects 2 projects 3

```
select * from projects p where id in (2, 8);
```

	id	name	status	cost	progress	created_at	updated_at
1	2	Haley and Sc	inactive	24 916	100	4-06-13 04:52:07	6 21:30:26.247419
2	8	Abshire-Mon	active	56 060	95	4-10-24 11:44:11	6 21:30:26.354256

Рис. 29

Данные были обновлены в соответствии с заданным условием, для сравнения можно обратиться к рисунку 21 (Рис. 21).

Листинг кода файла docker-compose.yml:

```
services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    ports:
      - "3306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: rootpassword
      MYSQL_DATABASE: project_management_development
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    networks:
      - app-network

  redis:
    image: redis:7.0
    ports:
      - "6379:6379"
    networks:
      - app-network

  web:
    build: .
    command: bash -c "rm -f tmp/pids/server.pid && rails server -b 0.0.0.0"
    volumes:
      - ./rails
    ports:
      - "3000:3000"
    depends_on:
      - db
      - redis
```


environment:

DATABASE_HOST: db

DATABASE_USERNAME: user

DATABASE_PASSWORD: password

REDIS_URL: redis://redis:6379/1

networks:

- app-network

sidekiq:

build: .

command: bundle exec sidekiq

depends_on:

- db

- redis

environment:

DATABASE_HOST: db

DATABASE_USERNAME: user

DATABASE_PASSWORD: password

REDIS_URL: redis://redis:6379/1

networks:

- app-network

volumes:

db_data:

networks:

app-network:

driver: bridge

Выводы по работе:

В ходе выполнения лабораторной работы были освоены основные принципы работы с Docker Compose для управления многоконтейнерными приложениями. Были выполнены следующие задачи:

1. Создание файла docker-compose.yml:
 - Разработан файл конфигурации для многоконтейнерного приложения, включающего сервисы: Ruby on Rails, MySQL, Redis, Sidekiq.
 - Указаны зависимости между сервисами, настройки портов, переменные окружения и тома для хранения данных.
2. Запуск приложения с помощью Docker Compose:
 - Приложение успешно запущено с использованием команды docker-compose up.
 - Проверено взаимодействие между контейнерами: Rails-приложение подключилось к базе данных MySQL и Redis для работы с фоновыми задачами через Sidekiq.
3. Генерация данных и их сохранение в MySQL:
 - Создана модель Project для хранения данных о проектах.
 - Реализована задача для генерации синтетических данных с использованием гема Faker.
 - Данные успешно сохранены в таблице projects базы данных MySQL.
4. Реализация расчета процента выполнения проектов:
 - Разработан джоб UpdateProjectProgressJob, который обновляет прогресс выполнения проектов в зависимости от их статуса:
 - Для проектов со статусом "active" прогресс увеличивается на случайное значение, но не превышает 100%.
 - Для проектов со статусом "inactive" прогресс устанавливается в 100%.
 - Настроен Sidekiq для выполнения фоновых задач и Redis для хранения очередей задач.
5. Проверка работоспособности приложения:
 - Проверено, что данные успешно генерируются и сохраняются в базе данных.

- Убедились, что фоновые задачи выполняются корректно и обновляют прогресс выполнения проектов.

Контрольные вопросы

1. Что такое Docker Compose и для чего он используется?

Docker Compose — это инструмент, который позволяет управлять многоконтейнерными приложениями. Он используется для одновременного запуска и управления несколькими контейнерами, входящими в состав приложения, что особенно полезно для сложных приложений с несколькими взаимосвязанными сервисами. Docker Compose работает на основе файла конфигурации `docker-compose.yml`, в котором описываются все необходимые настройки и зависимости между контейнерами.

2. Какие основные преимущества использования Docker Compose для управления многоконтейнерными приложениями?

- Упрощение управления: Docker Compose позволяет управлять несколькими контейнерами через один файл конфигурации (`docker-compose.yml`), что упрощает настройку и запуск сложных приложений.
- Автоматизация: Compose автоматически создает и запускает все сервисы, указанные в конфигурации, в правильном порядке, учитывая зависимости между контейнерами.
- Изоляция окружения: Каждый сервис (контейнер) работает в изолированном окружении, что уменьшает вероятность конфликтов между компонентами приложения.
- Масштабируемость: Compose поддерживает масштабирование сервисов, что позволяет легко увеличивать количество экземпляров контейнеров.
- Повторяемость: Конфигурация в `docker-compose.yml` гарантирует, что приложение будет запускаться одинаково на разных окружениях (разработка, тестирование, production).
- Удобство разработки: Compose упрощает локальную разработку, позволяя быстро запускать и останавливать все компоненты приложения.

3. Какие основные разделы и директивы используются в файле `docker-compose.yml`?

- `version`: Указывает версию формата файла (например, '3.8').
- `services`: Основной раздел, где описываются все сервисы (контейнеры) приложения.
- `image`: Указывает образ Docker, который будет использоваться для создания контейнера.
- `build`: Указывает путь к Dockerfile для сборки образа.
- `ports`: Пробрасывает порты из контейнера на хост.
- `volumes`: Определяет тома для хранения данных или монтирования директорий.
- `environment`: Задаёт переменные окружения.
- `depends_on`: Указывает зависимости между сервисами (например, база данных должна запускаться перед приложением).
- `networks`: Подключает сервис к определенной сети.
- `command`: Переопределяет команду по умолчанию для запуска контейнера.
- `networks`: Определяет пользовательские сети для связи между контейнерами.
- `volumes`: Создает именованные тома для хранения данных.

4. Как запустить многоконтейнерное приложение с помощью Docker Compose?

- Создайте файл `docker-compose.yml` с описанием сервисов.
- Перейдите в директорию, где находится файл `docker-compose.yml`.
- Выполните команду:
- `bash`
- `Copy`
- `docker-compose up`

5. Как остановить и удалить контейнеры, запущенные с помощью Docker Compose?

Остановка контейнеров: `docker-compose down`

Эта команда останавливает и удаляет контейнеры, сети и тома, созданные командой `docker-compose up`.

Остановка без удаления: `docker-compose stop`