Введение

В рамках практической работы создано web-приложение, запускаемое с помощью Python http.server. Приложение упаковано в Docker image и развернуто в Kubernetes. Ниже описан процесс выполнения задания.

Ход работы

1. Работа с Docker

На этом этапе была создана веб-страница с шапкой и пятью фото, создан dockerfile, собран docker-образ, запущен docker-контейнер и сделана публикация образа на Docker Hub. Ход работы представлен ниже.

1. Верстка веб-страницы

Ниже представлен html-код веб-страницы.

| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Моя страница</title>  <style>  body, html {  margin: 0;  padding: 0;  width: 100%;  height: 100%;  }  .header {  display: flex;  flex-direction: column;  align-items: center;  justify-content: center;  height: 20%;  background-color: #f0f0f0;  }  .avatar {  width: 100px;  height: 100px;  border-radius: 50%;  background-size: cover;  background-image: url('source/avatar.jpg');  }  .name {  margin-top: 10px;  font-size: 20px;  color: #333;  }  .photo-gallery {  display: grid;  grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));  gap: 10px;  padding: 20px;  }  .photo {  width: 100%;  height: 100px;  background-size: cover;  }  </style>  </head>  <body>  <div class="header">  <div class="avatar"></div>  <div class="name">Антон Краснояров</div>  </div>  <div class="photo-gallery">  <div class="photo" style="background-image: url('source/1.jpg');"></div>  <div class="photo" style="background-image: url('source/2.jpg');"></div>  <div class="photo" style="background-image: url('source/3.jpg');"></div>  <div class="photo" style="background-image: url('source/4.jpg');"></div>  <div class="photo" style="background-image: url('source/5.jpg');"></div>  </div>  </body>  </html> |
| --- |

1. Создание Dockerfile

Для сборки образа Docker был использован базовый образ python:3-alpine. В Dockerfile (рисунок 1) выполнены следующие действия:

* создан каталог /app, назначенный рабочим каталогом (WORKDIR);
* добавлен файл index.html с шапкой и 5 фотографиями;
* настроен запуск веб-сервера на Python.

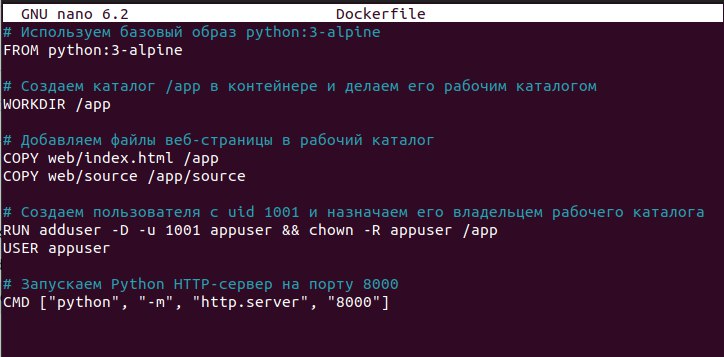


Рисунок 1. Содержимое файла «Dockerfile»

1. Сборка Docker Image

Ниже представлена команда сборки Docker Image с тегом 1.0.0.

| docker build -t course-work:1.0.0 |
| --- |

1. Запуск Docker Container

Контейнер был запущен с помощью следующей команды.

| docker run -d -p 8001:8000 course-work:1.0.0 |
| --- |

Далее была произведена проверка загрузки страницы в браузере [http://127.0.0.1:800](http://127.0.0.1:8001/%D0%9C%D0%BE%D1%8F)1.

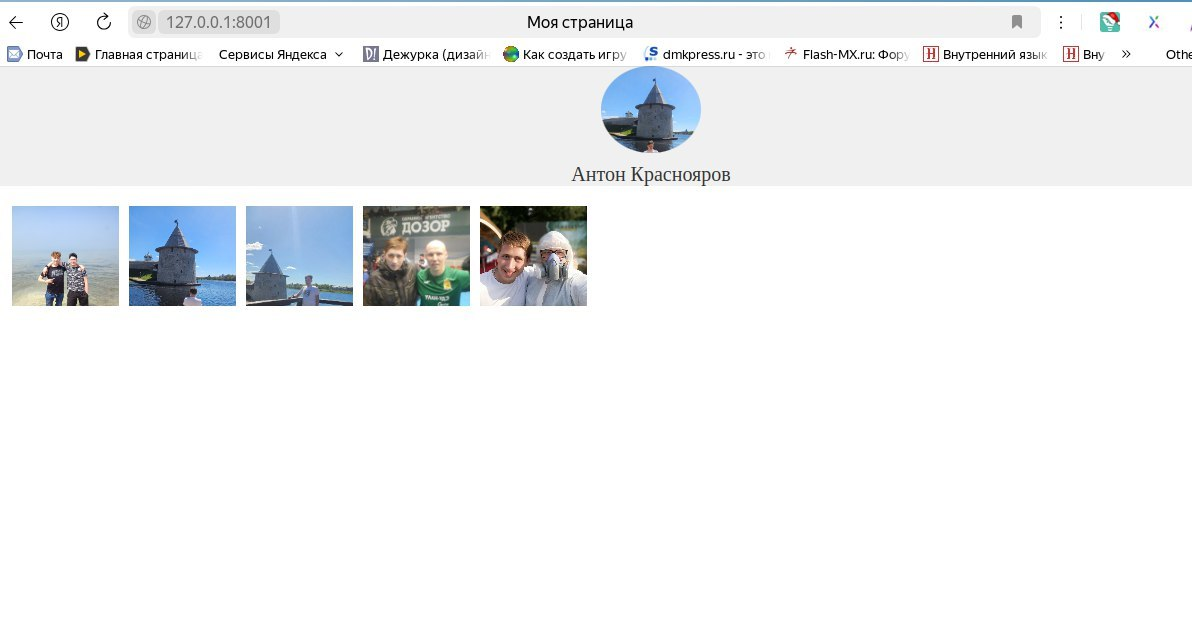


Рисунок 2. Web-страница

1. Загрузка Image на Docker Hub

На рисунке представлена авторизация на Docker Hub.

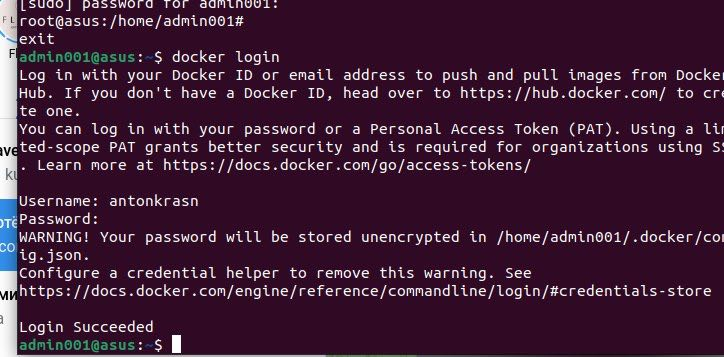


Рисунок 3. Авторизация на Docker Hub

Далее была выполнена загрузка образа. Команды представлены ниже.

| docker tag course-work:1.0.0 antonkrasn/course-work:1.0.0  docker push antonkrasn/course-work:1.0.0 |
| --- |

2. Работа с Kubernetes

Ввиду технических ограничений вместо полноценного kubernetes был использован minicube. Процесс выполнения задания представлен ниже.

1. Установка minicube

Для установки minicube была использована следующая команда.

| curl https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64  sudo install minikube-linux-amd64 /usr/local/bin/minikube- |
| --- |

1. Запуск minicube

Далее был запущен minicube с двумя нодами (рисунки 4-5).

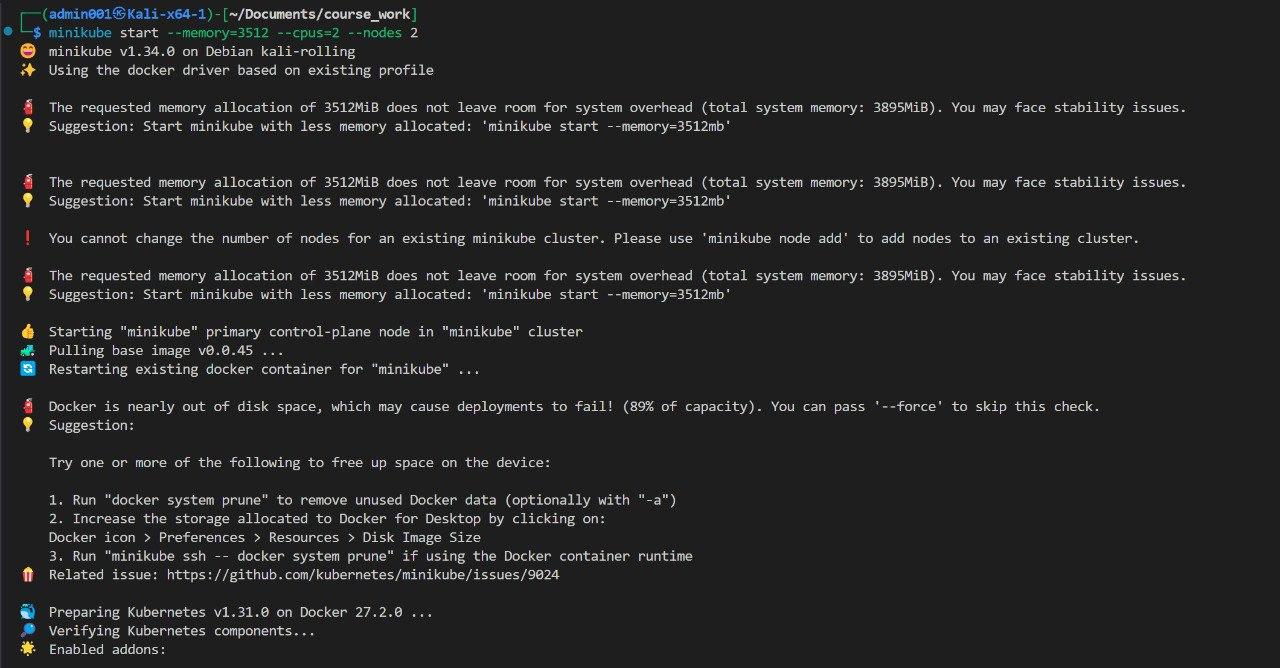


Рисунок 4. Запуск minicube

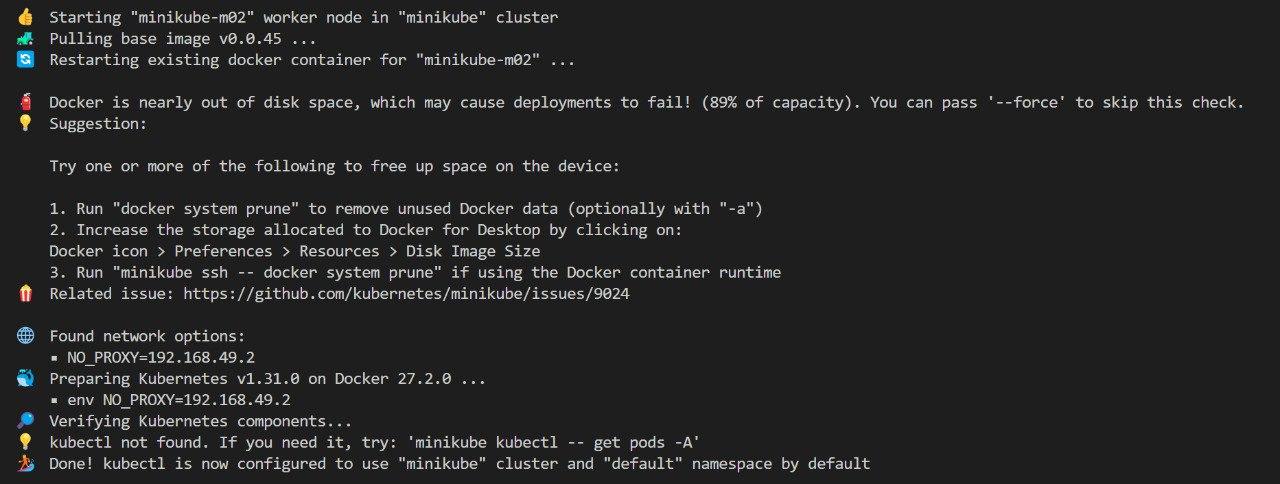


Рисунок 5. Запуск minicube продолжение

1. Создание Kubernetes Deployment

Был создан Deployment manifest для Kubernetes с именем web и двумя репликами контейнера. Использован следующий код:

| apiVersion: apps/v1  kind: Deployment  metadata:  name: web  spec:  replicas: 2  selector:  matchLabels:  app: web  template:  metadata:  labels:  app: web  spec:  containers:  - name: web  image: antonkrasn/course-work:1.0.0  ports:  - containerPort: 8000 |
| --- |

7. Установка Deployment в Kubernetes

Применение манифеста (рисунок 6):

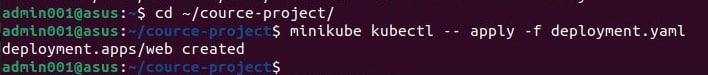


Рисунок 6. Применение манифеста

8. Доступ к приложению через Kubernetes

Для проверки работы приложения выполнили port-forward (рисунок 7):

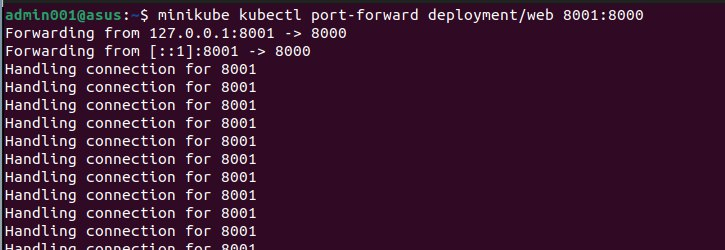


Рисунок 7. Выполнение port-forward

Проверили доступ, перейдя по адресу [http://127.0.0.1:800](http://127.0.0.1:8001/%D0%9C%D0%BE%D1%8F)1.

9. Описание способа создания Service

1.Создание файла (рисунок 8)

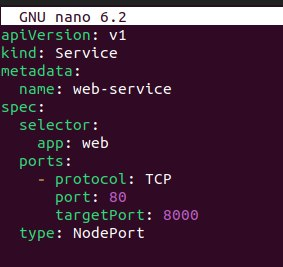


Рисунок 8. Выполнение port-forward

2.Применение манифеста web-service.yaml (рисунок 9)

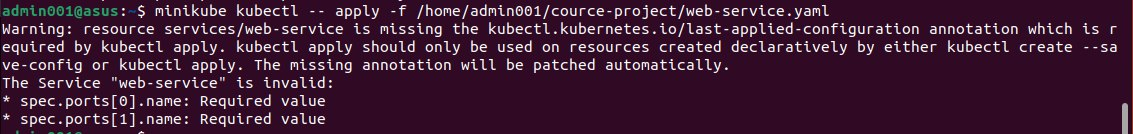


Рисунок 9. Применение манифеста web-service.yaml

3.Проверка результата (рисунок 10)

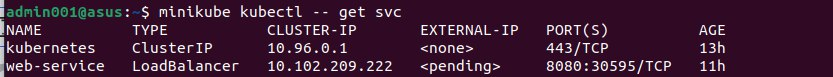


Рисунок 10. Проверка результата

Результаты

Приложение успешно упаковано в Docker.

Образ загружен на Docker Hub.

Web-приложение развернуто в Kubernetes и проверена его доступность через port-forward.

Результаты работы команд:

1. Результат работы команды kubectl describe deployment web (рисунок 11)

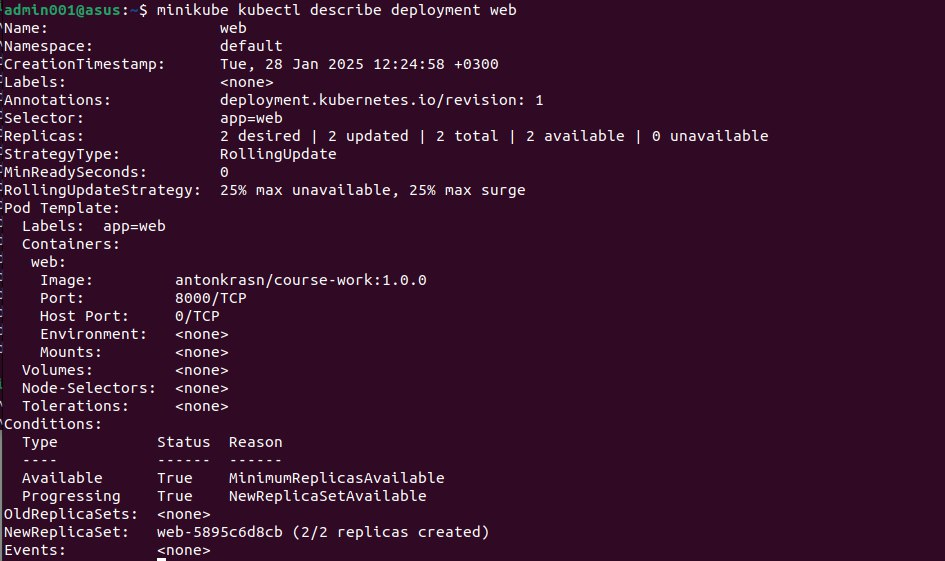


Рисунок 11. Выполнение kubectl describe deployment web

1. Результат работы команды kubectl describe service web (рисунок 12)

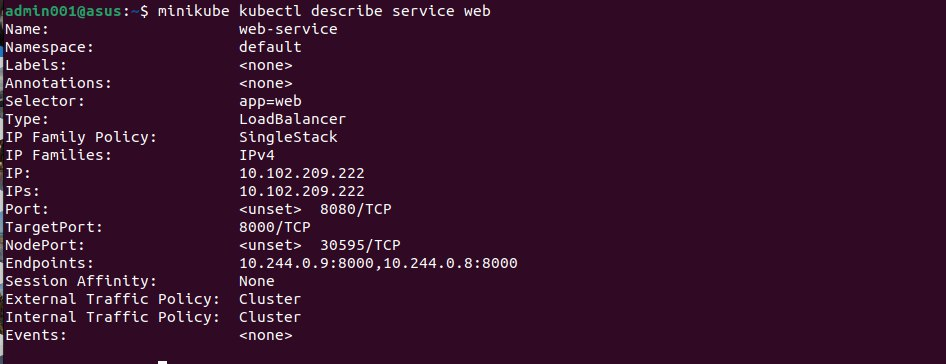


Рисунок 12. Выполнение kubectl describe service web

1. Результат работы команды curl (рисунки 13 и 14)

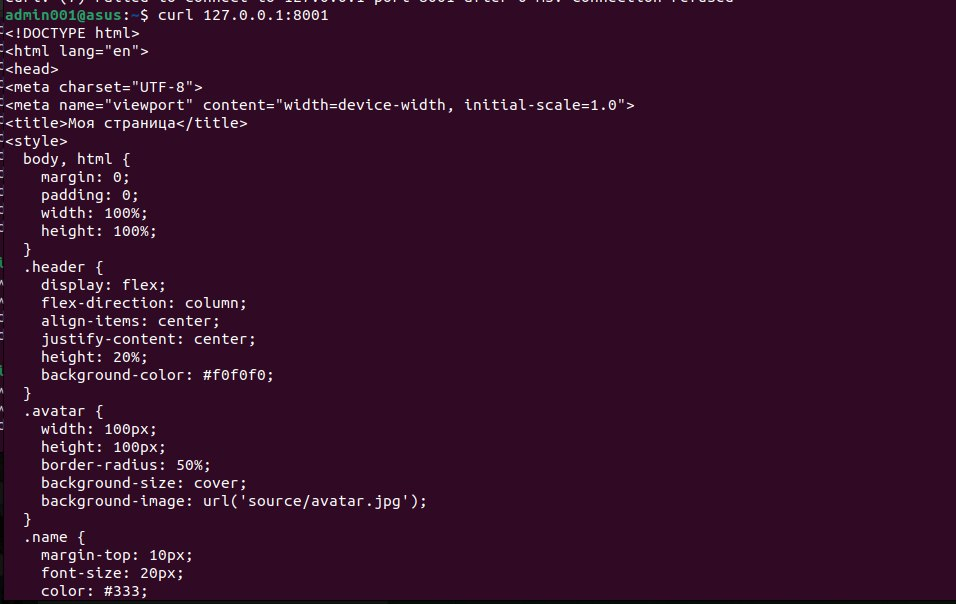


Рисунок 13. Выполнение curl

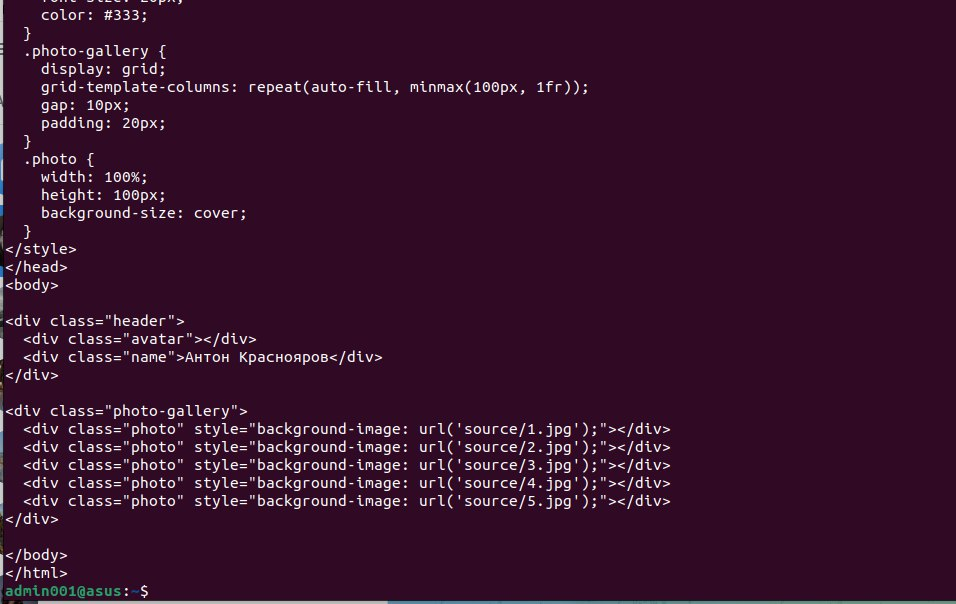


Рисунок 14. Выполнение curl