Для выполнения лабораторной работы Вам потребуется следующее:

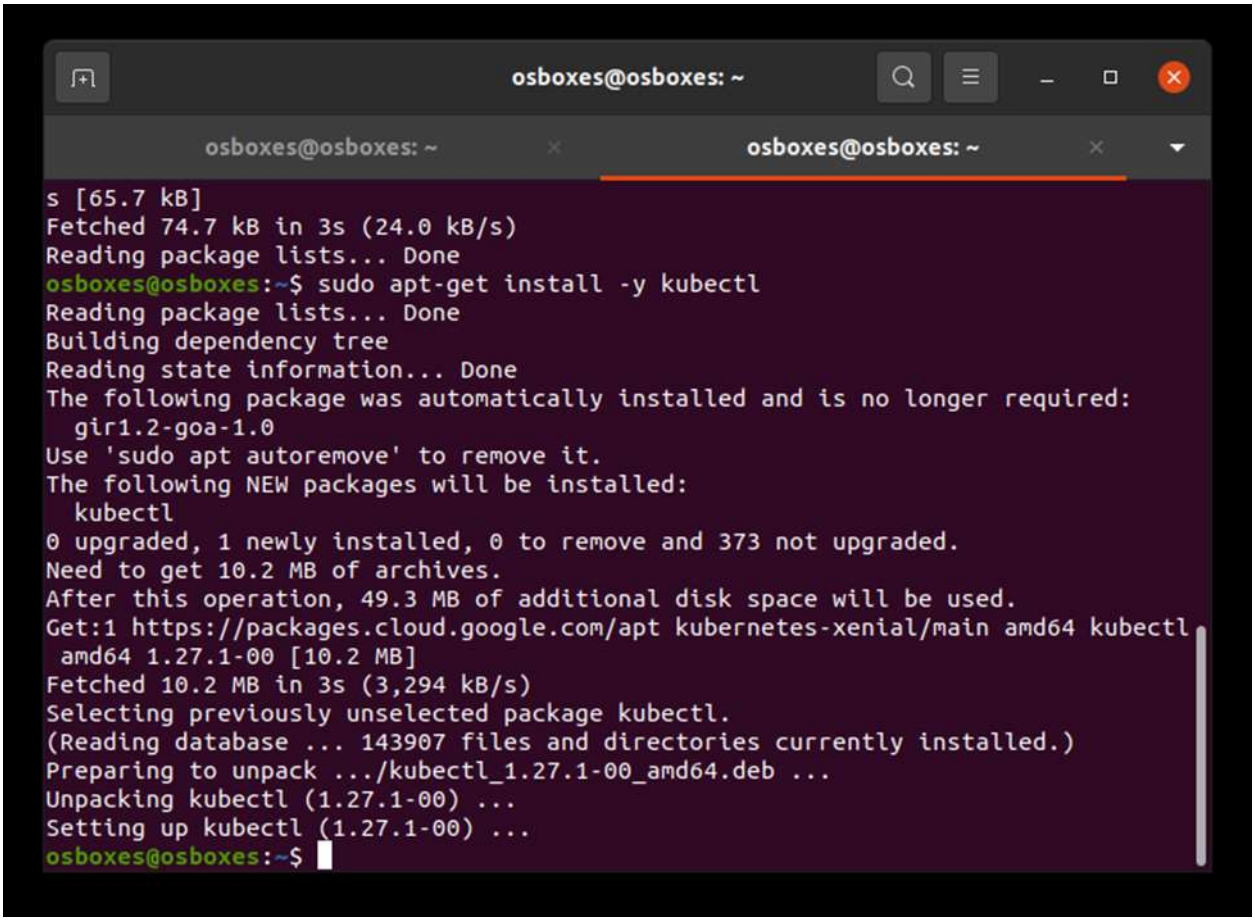
1. Установленный Docker, Docker Compose.
2. Учетная запись на сайте https://hub.docker.com.

Инструмент командной строки Kubernetes kubectl

Необходимо установить инструмент командной строки Kubernetes kubectl https://kubernetes.io/ru/docs/tasks/tools/install-kubectl/

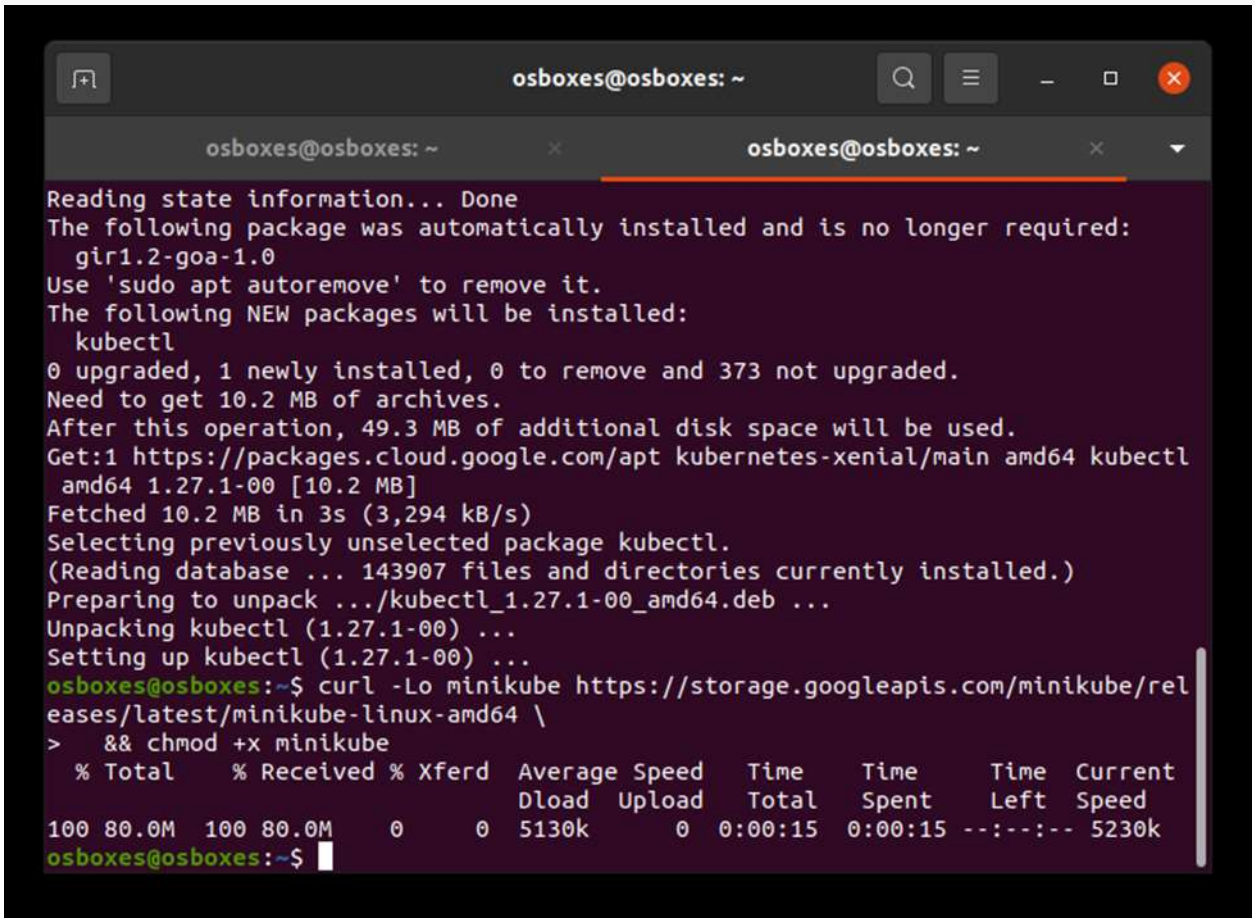
Установка с помощью встроенного пакетного менеджера

```
sudo apt-get update && sudo apt-get install -y apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubectl
```



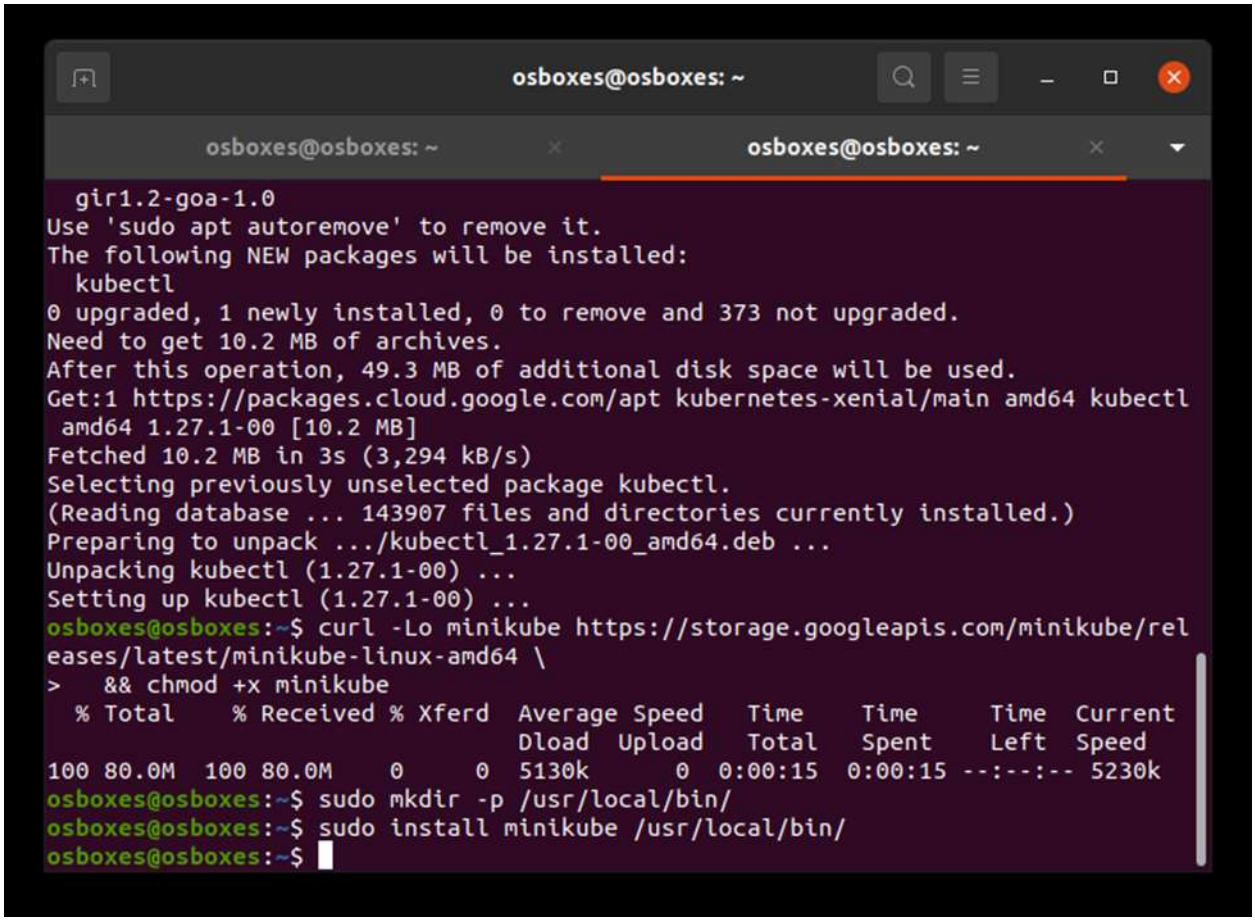
Установка Minikube

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \
  && chmod +x minikube
```



Чтобы исполняемый файл Minikube был доступен из любой директории выполните следующие команды:

```
sudo mkdir -p /usr/local/bin/
sudo install minikube /usr/local/bin/
```

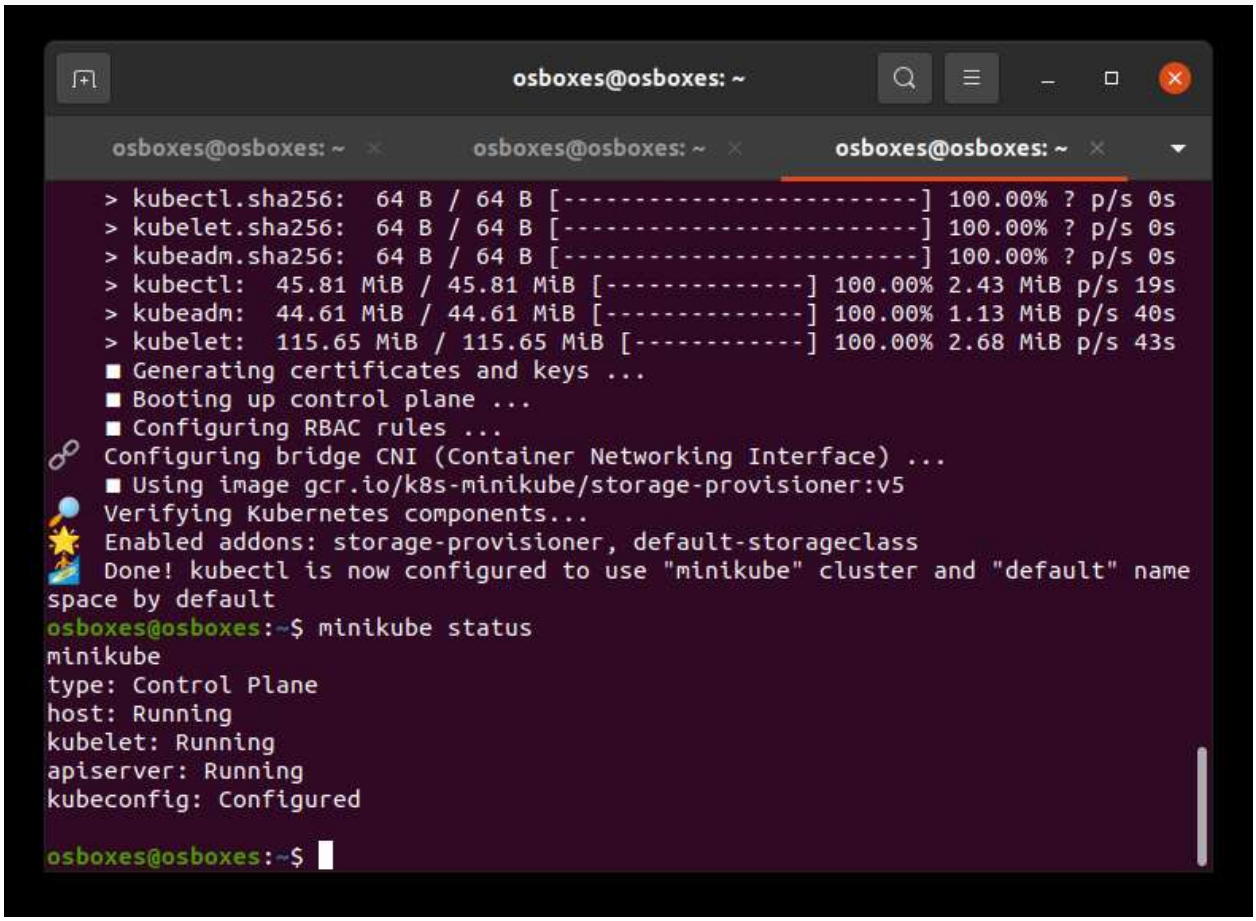


Запустите локальный кластер Kubernetes:

```
minikube start  --driver=docker
```

и проверьте состояния кластера:

```
minikube status
```



Создадим небольшое веб приложение app.py

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World! Simple web app'

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0',port='4000')
```

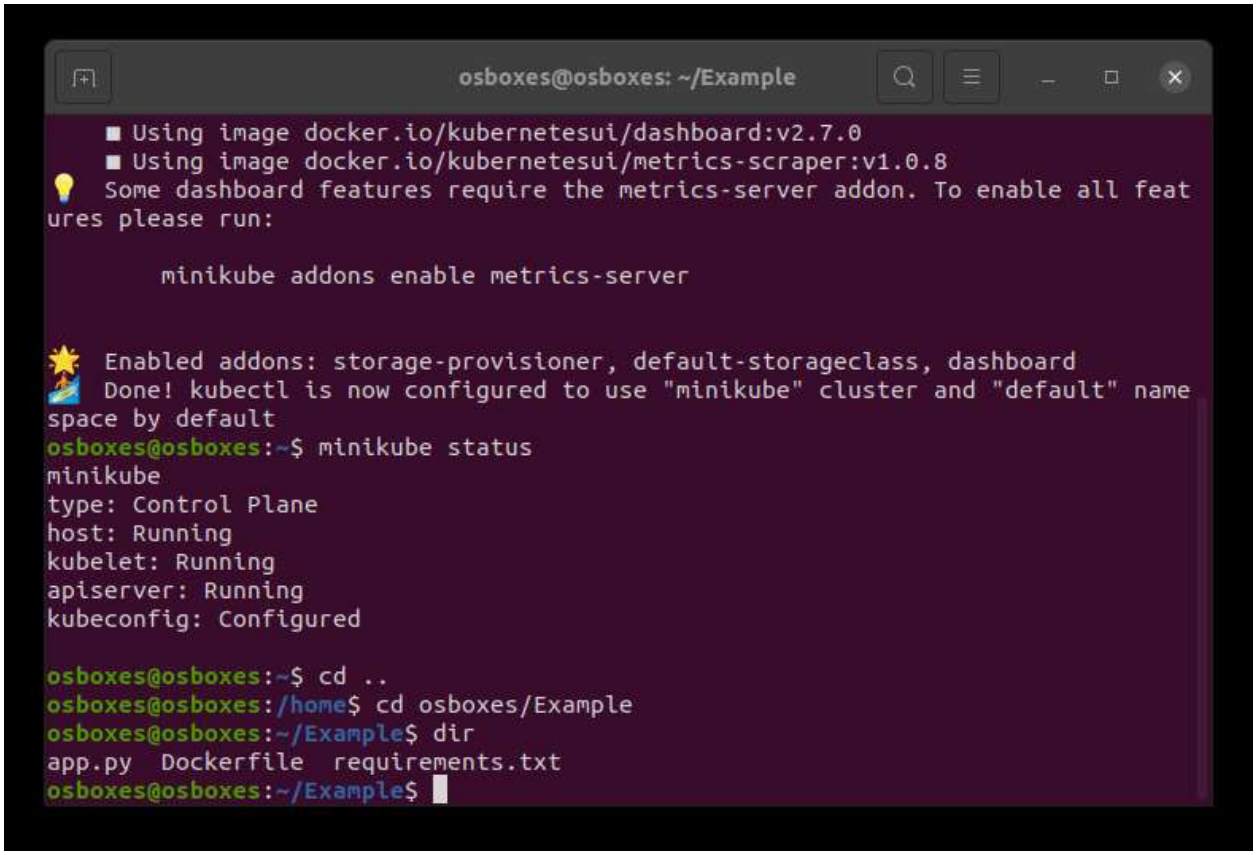
Dockerfile

```
# Use an official Python runtime as a parent image
FROM python:3.9-slim-buster

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in requirements.txt
RUN pip install --no-cache-dir -r requirements.txt

# Make port 4000 available to the world outside this container
EXPOSE 4000

# Define environment variables
ENV FLASK_APP=app.py
ENV FLASK_ENV=development
ENV FLASK_DEBUG=1

# Run app.py when the container launches
CMD ["flask", "run", "--host=0.0.0.0", "--port=4000"]
```

файл requirements.txt

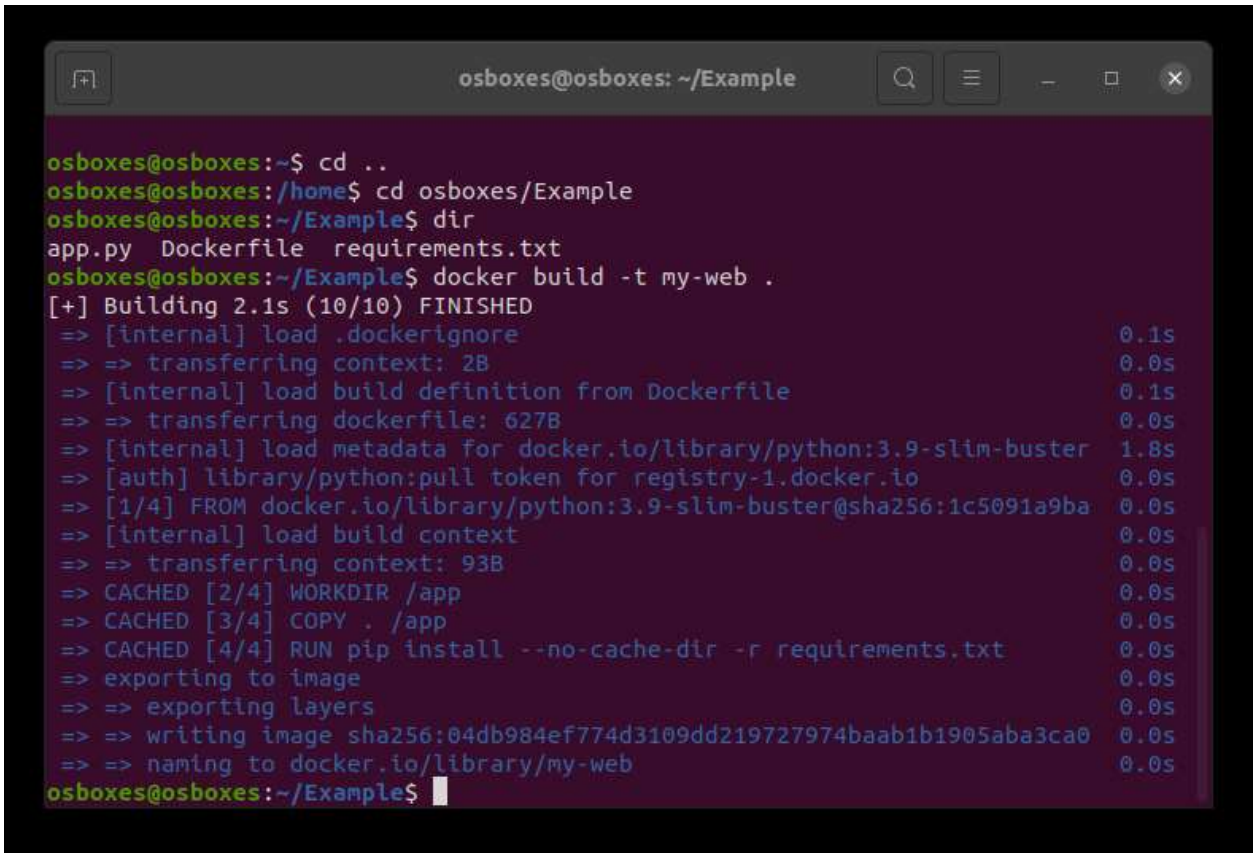
```
Flask==2.1.0
```

Перейдем в текущий каталог с созданными файлами



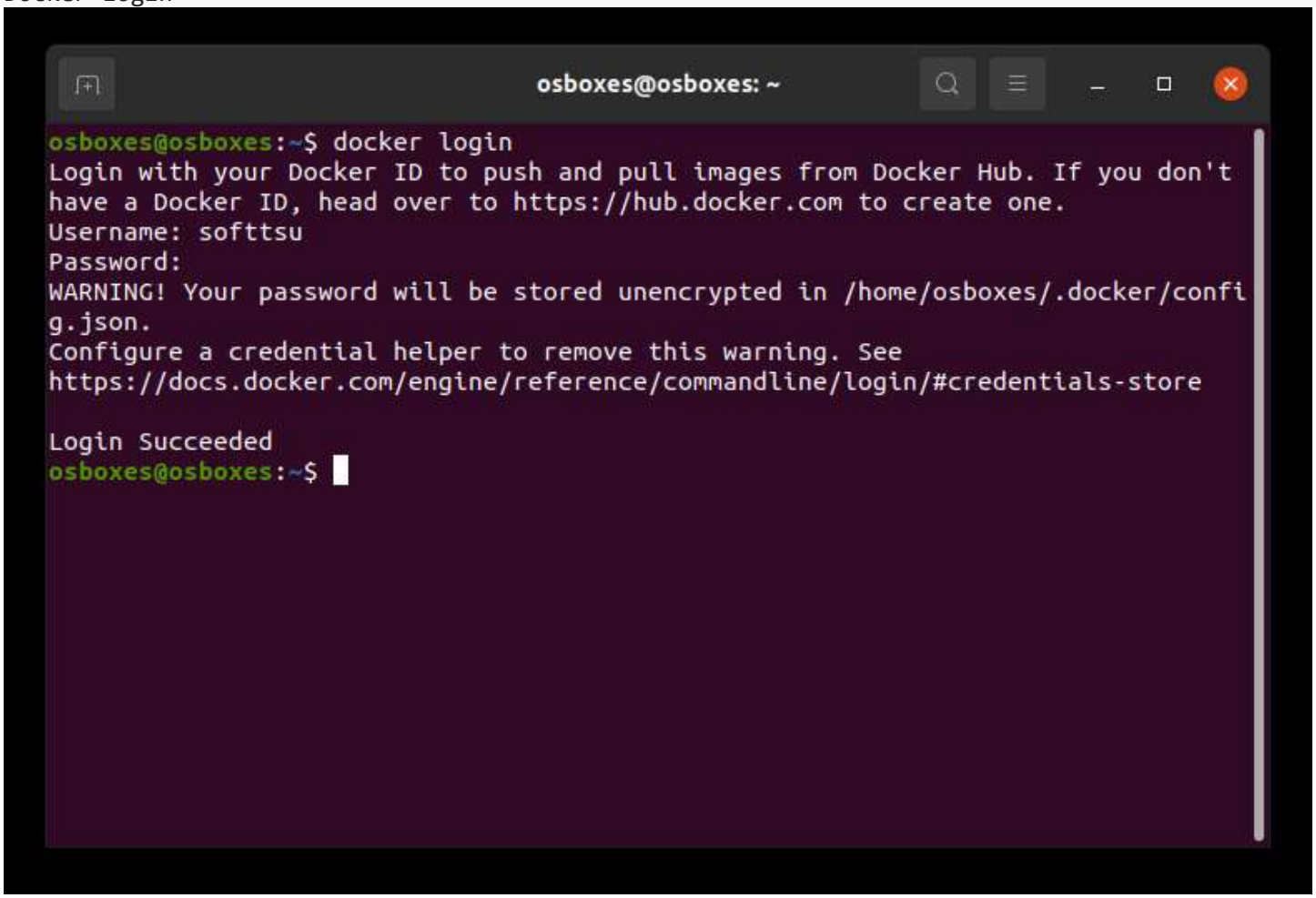
Соберем образ

```
docker build -t my-web .
```



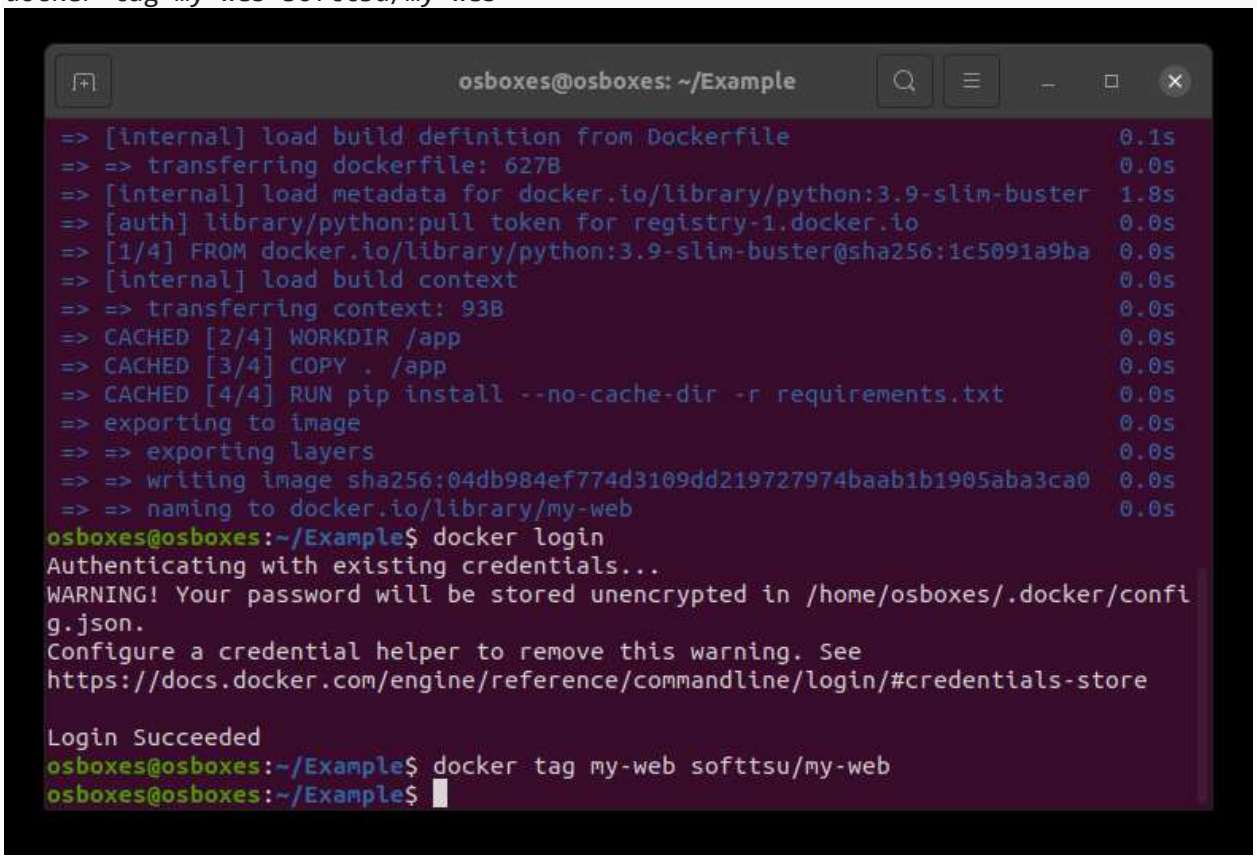
Авторизуемся в docker

```
Docker login
```



Создадим тег образа

```
docker tag my-web softtsu/my-web
```



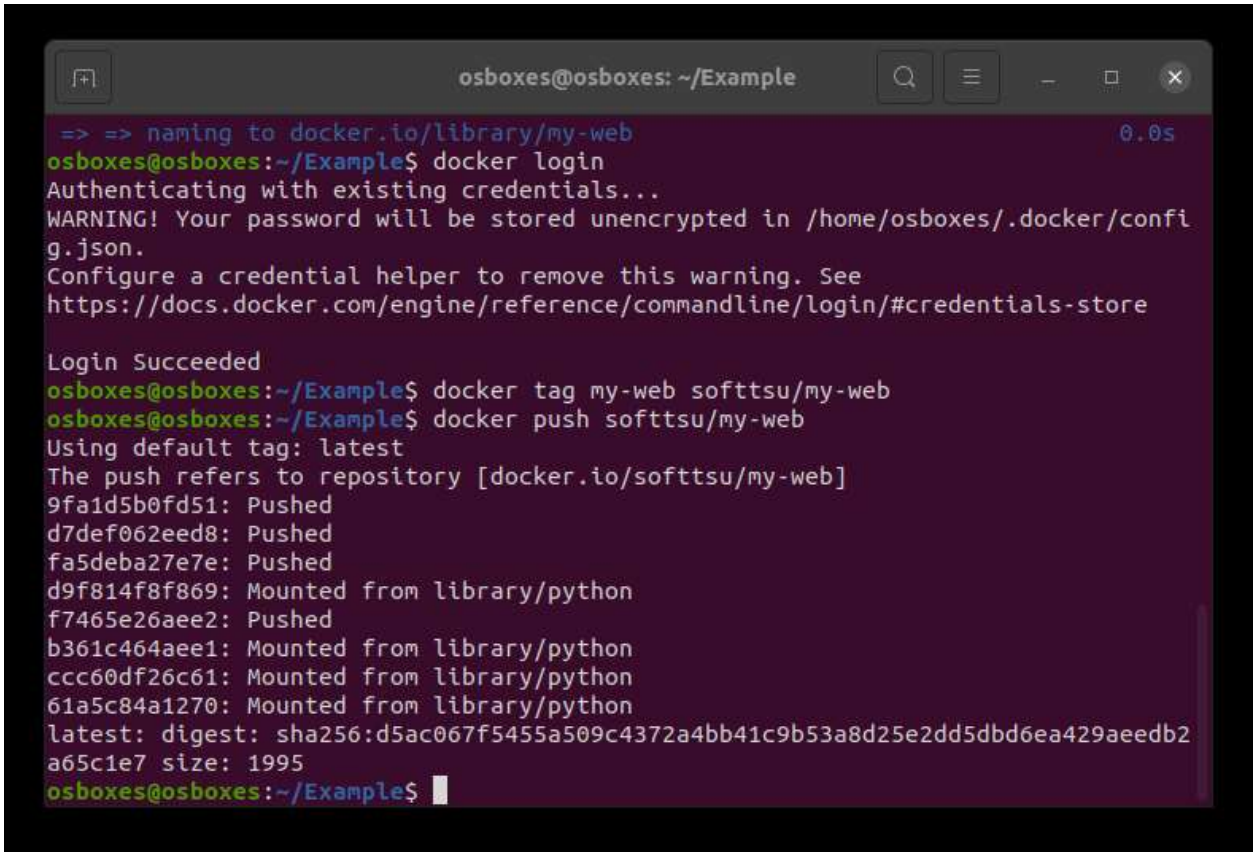
И разместим его в репозитории используя следующий формат

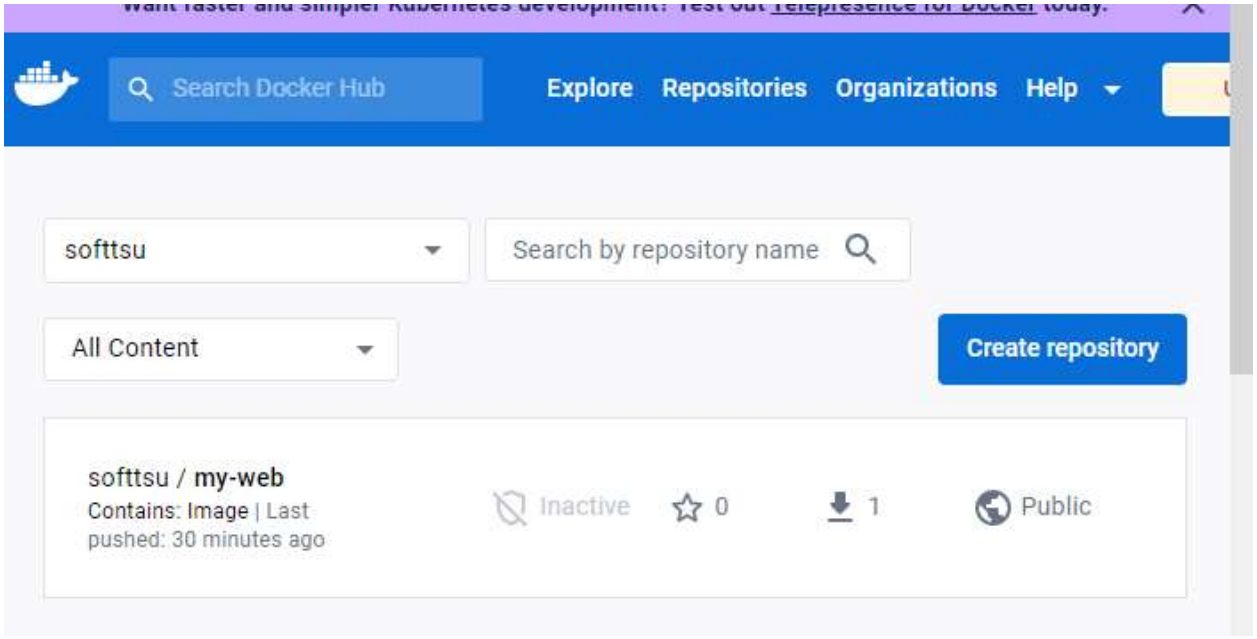
docker push user_name/repository_name,

где user_name имя пользователя на сайте https://hub.docker.com

repository_name имя репозитория

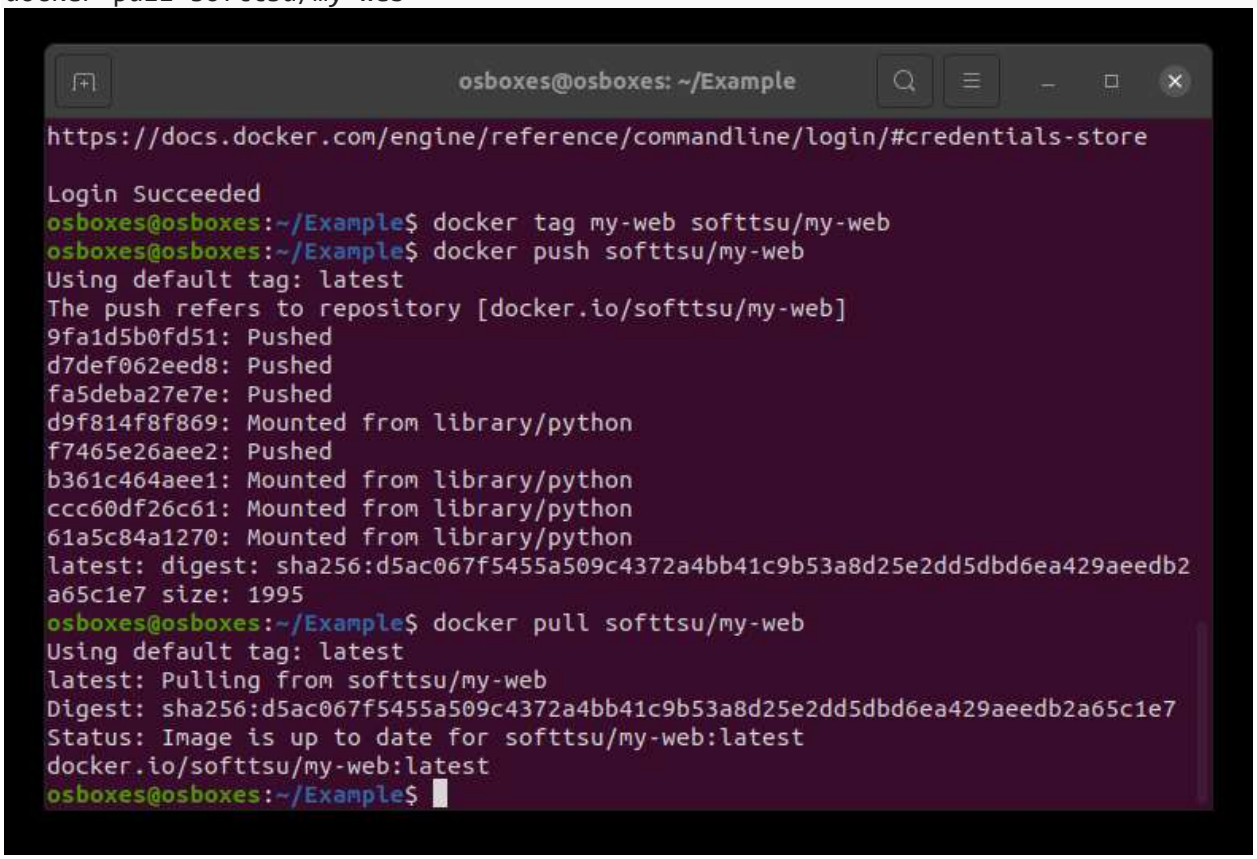
```
docker push softtsu/my-web
```





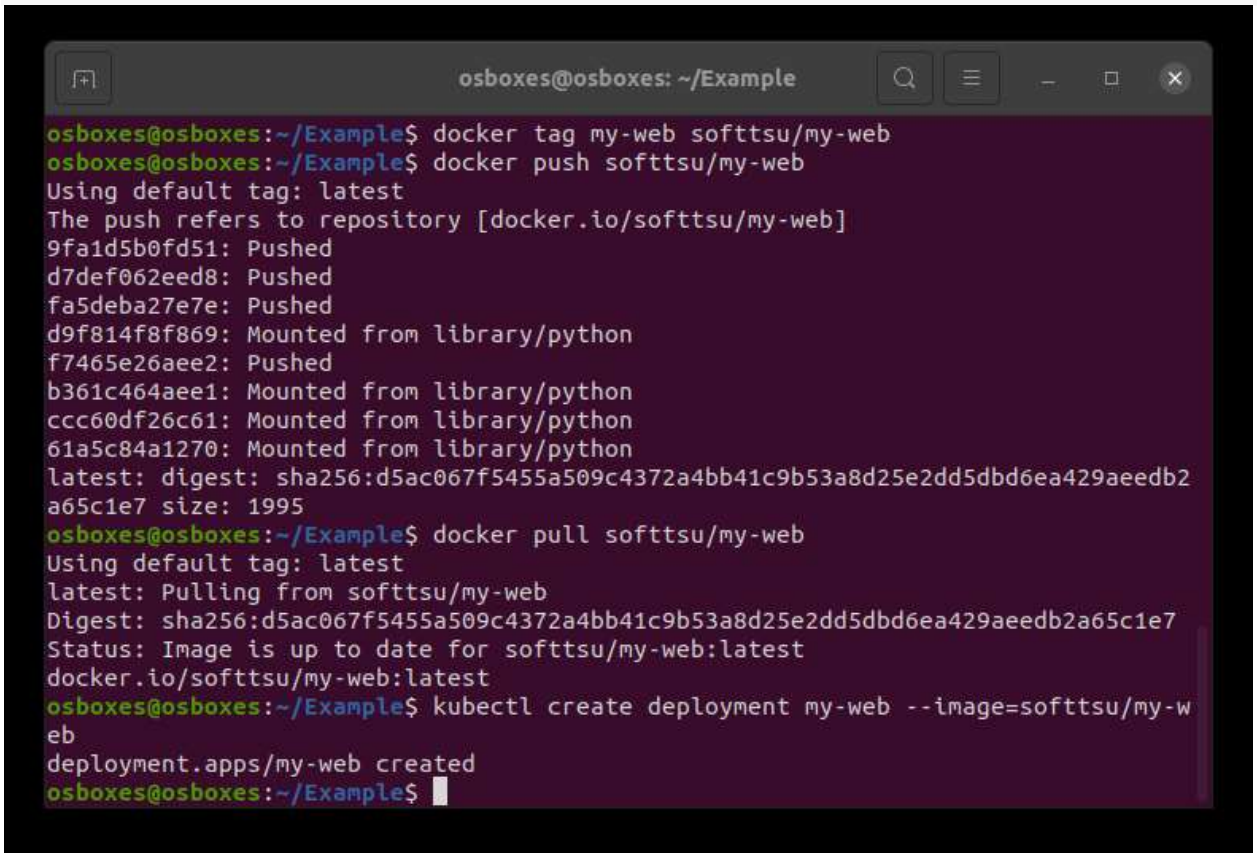
Загрузим образ из Docker Hub с помощью команды pull

```
docker pull softtsu/my-web
```



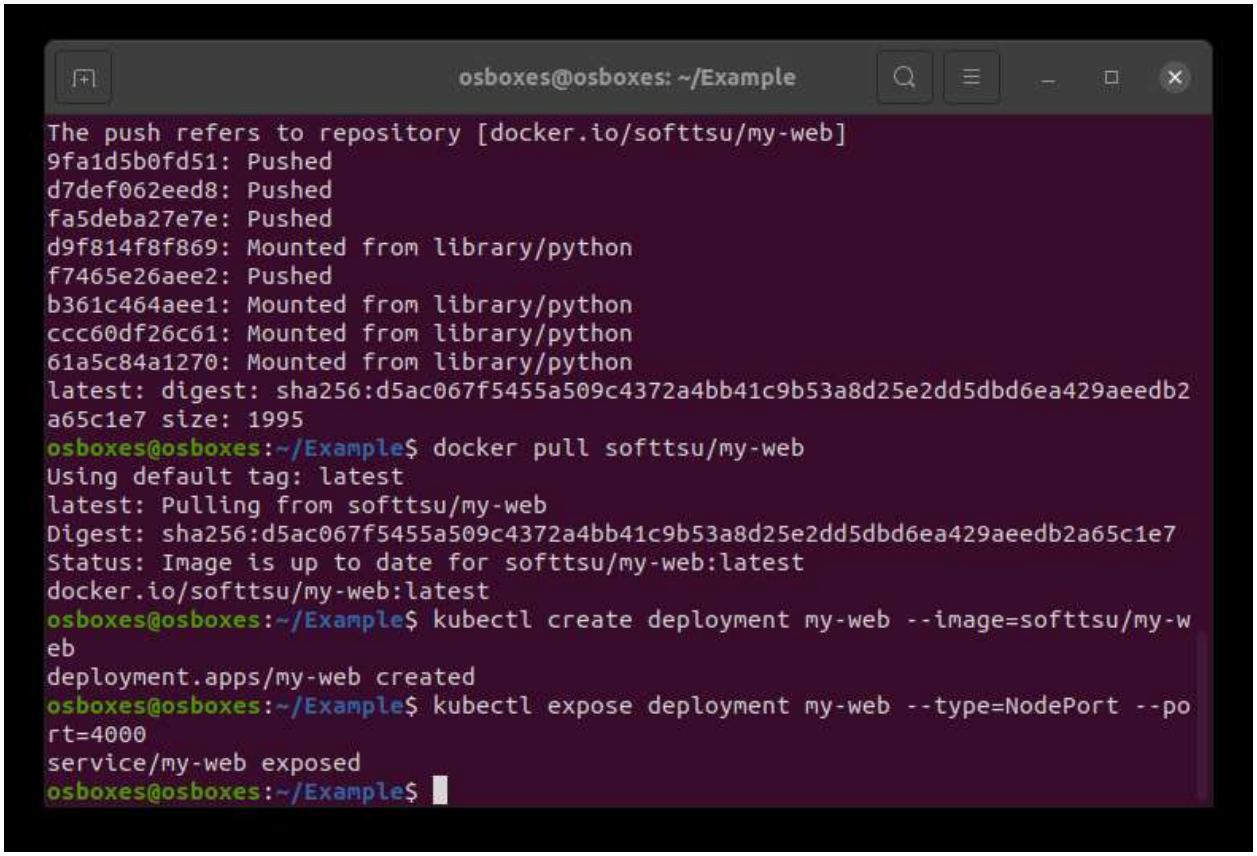
Создадим развёртывание (Deployment) в Kubernetes

```
kubectl create deployment my-web --image=softtsu/my-web
```

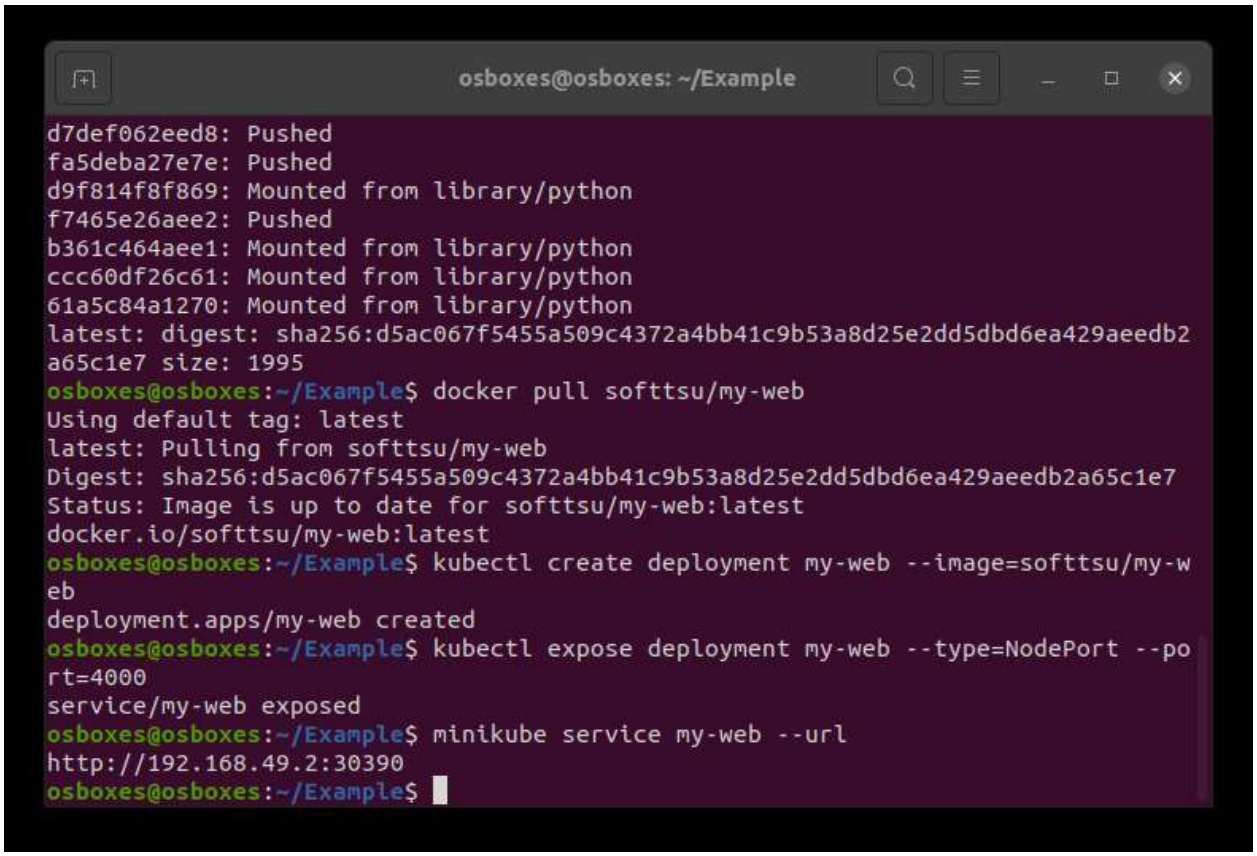


Чтобы получить доступ к объекту Deployment извне, создадим объект сервиса (Service):

```
kubectl expose deployment my-web --type=NodePort --port=4000
```

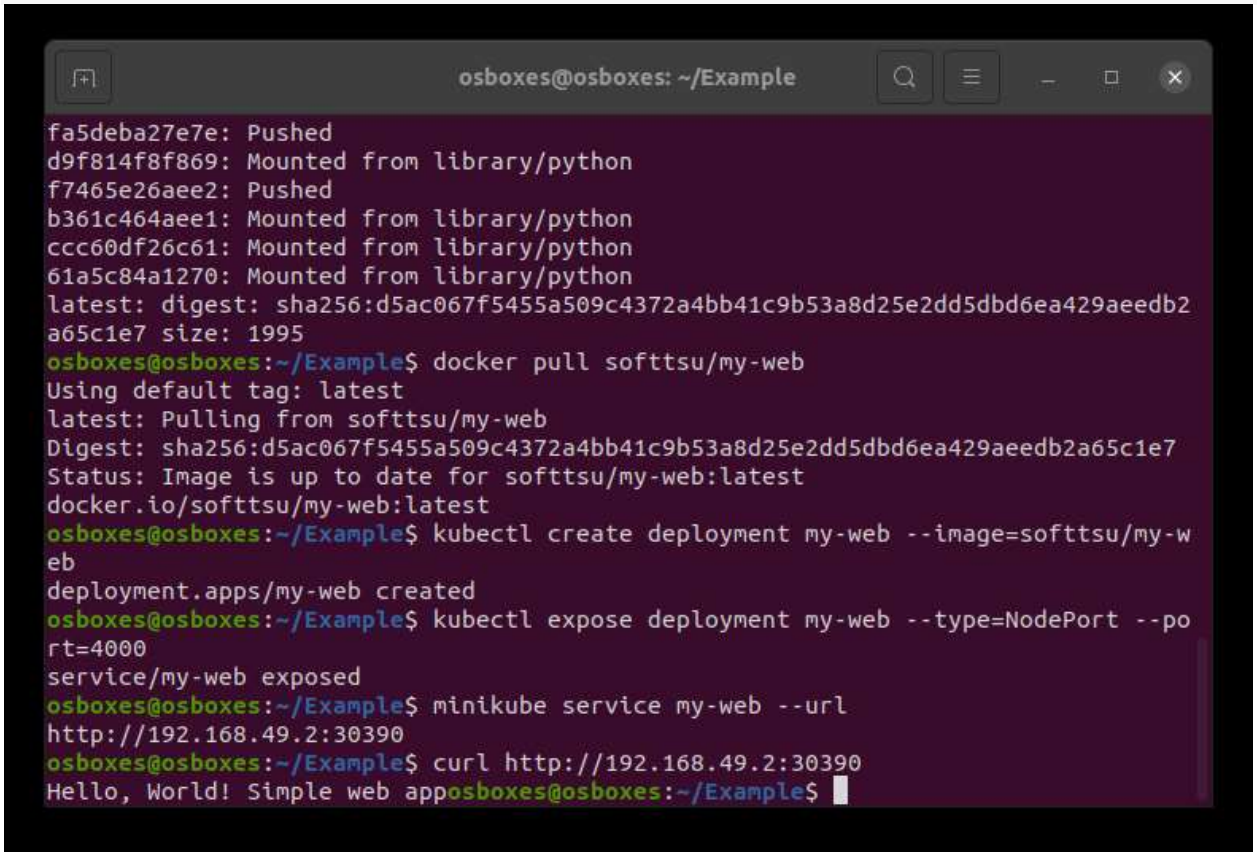


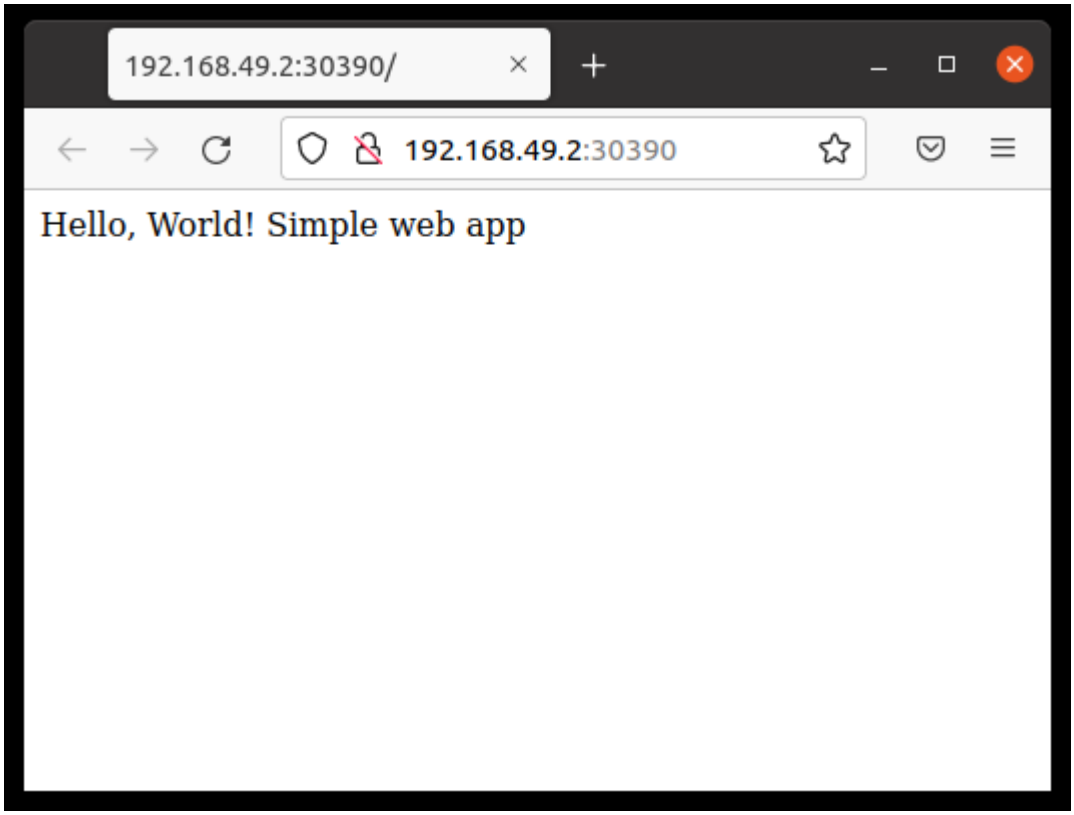
Узнайте URL-адрес открытого (exposed) сервиса, чтобы просмотреть подробные сведения о сервисе:



Результат работы веб приложения

```
curl http://192.168.49.2:30390
```





Просмотреть графический интерфейс kubernetes можно с помощью команды

```
minikube dashboard
```