Автомаштабирование в K8s

(вспомогательный материал для выполнения 5 лабораторной работы)

Contents

[Требования к 5 лабораторной работе ”Автомаштабирование в K8s” 3](#_Toc87261865)

[Реализация 5 лабораторной работы 4](#_Toc87261866)

[Используем терминал CLOUD SHELL: 5](#_Toc87261867)

[Создание кластера K8s on GCP using KE DASHBOARD 6](#_Toc87261868)

[Google cloud SDK installer 6](#_Toc87261869)

[Configuring cluster access for kubectl using SDK 8](#_Toc87261870)

[Создание кластера K8s on CLOUD SHELL 9](#_Toc87261871)

[Создаем clusterul lab4 с помощью команды: gcloud container clusters create lab4 10](#_Toc87261872)

[Просматриваем информацию об узлах командой: gcloud compute instances list 10](#_Toc87261873)

[Выполнение команд kubectl для автомасштабирования в терминале Cloud Shell 10](#_Toc87261874)

[Cоздаем сервис LoadBalancer 11](#_Toc87261875)

[Примеры автоматического масштабирования 15](#_Toc87261876)

[Exemplul 1. Node js server 15](#_Toc87261877)

[Продолжение в соответствии с первым примером (node js) из темы 8, а именно: 17](#_Toc87261878)

[Exemplul 2. Run & expose php-apache server 17](#_Toc87261879)

[Выполнение 17](#_Toc87261880)

[Многопараметрический метод автомасштабирования 20](#_Toc87261881)

[Продолжение в соответствии со вторым примером (php) из темы 8 20](#_Toc87261882)

[ВЫВОДЫ 20](#_Toc87261883)

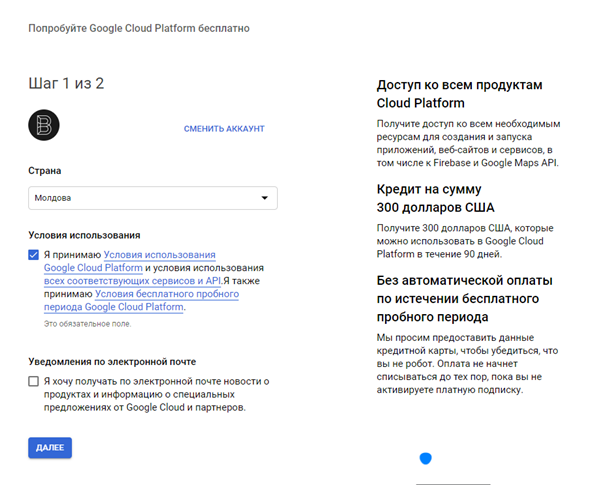
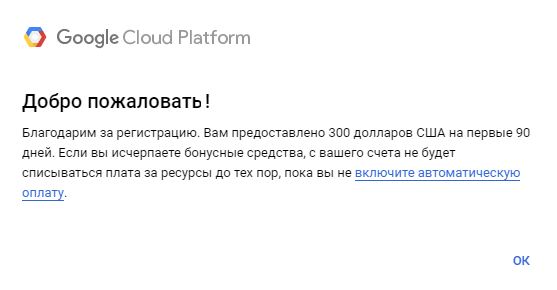
[БИБЛИОГРАФИЯ 21](#_Toc87261884)

# Требования к 5 лабораторной работе ”Автомаштабирование в K8s”

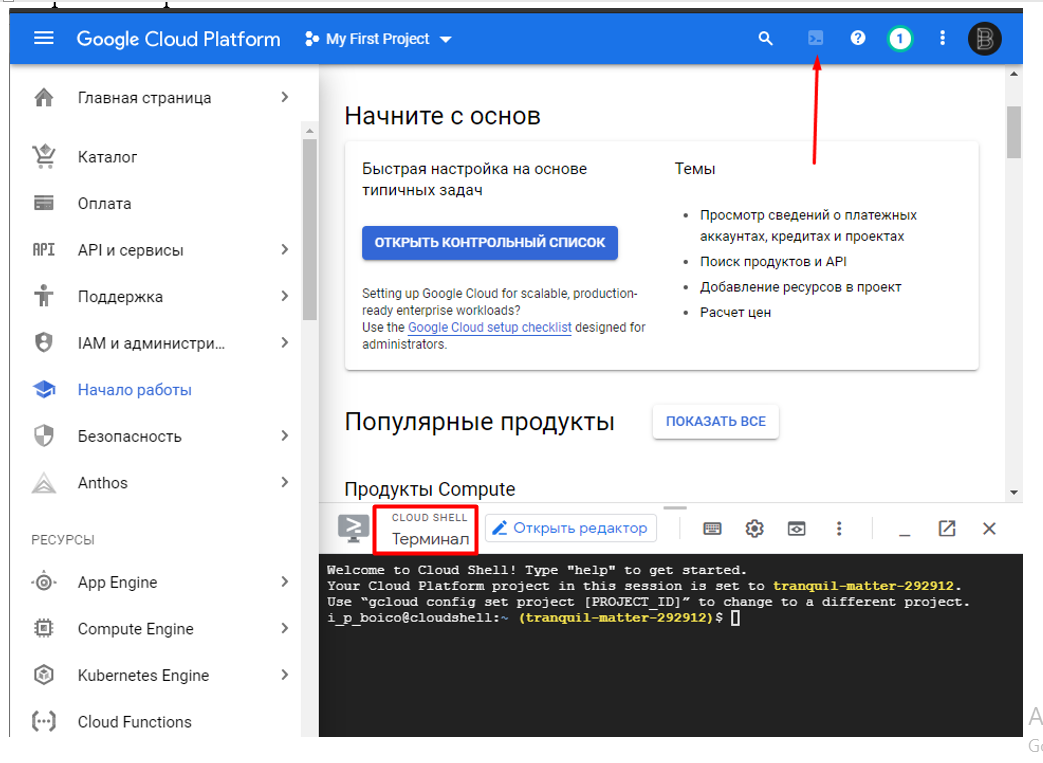
* Создание учетной записи GCP
* Установка Cloud SKD
* Создание кластера k8s в облаке Google 3 способами, используя:
  + Панель управления Kubernetes Engine
  + Cloud Shell
  + Cloud SDK
* Выполнение команд kubectl в кластере K8s и примеры автомасштабирования для 2х примеров серверов (node.js и php) из Темы 8 курса (в Cloud Shell и SDK)
* Выводы
* Библиография

# Реализация 5 лабораторной работы

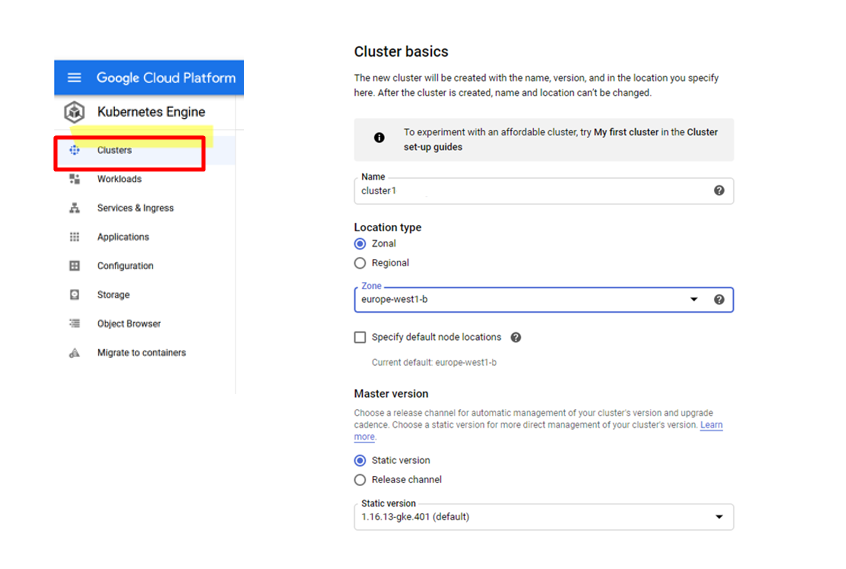
Создание кластера k8s в Google Cloud Platform при помощи Kubernetes Engine Dashboard

Регистрация в Google Cloud  
  
  


## Используем терминал CLOUD SHELL:



## Создание кластера K8s on GCP using KE DASHBOARD



O imagine care conține text

Descriere generată automat

# Google cloud SDK installer

O imagine care conține text

Descriere generată automat

<https://cloud.google.com/sdk/docs/quickstart>

O imagine care conține text

Descriere generată automat

[https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-windows](https://kubernetes.io/docs/tasks/tools/install-kubectl/)

## Configuring cluster access for kubectl using SDK

O imagine care conține text

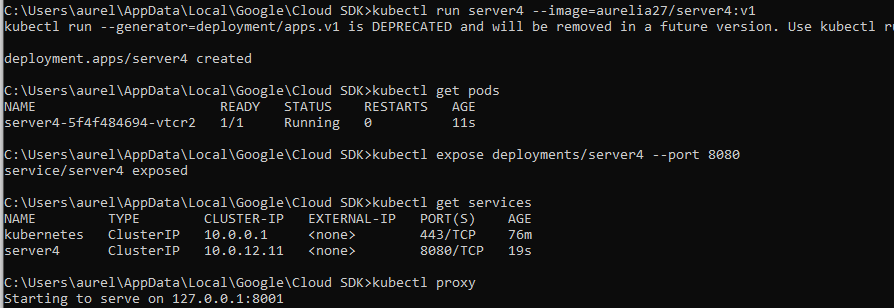
Descriere generată automat

O imagine care conține text

Descriere generată automat

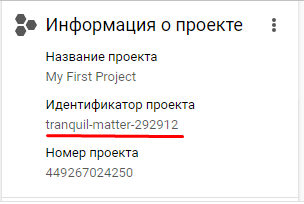
O imagine care conține text

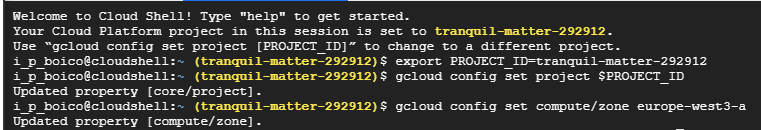
Descriere generată automat



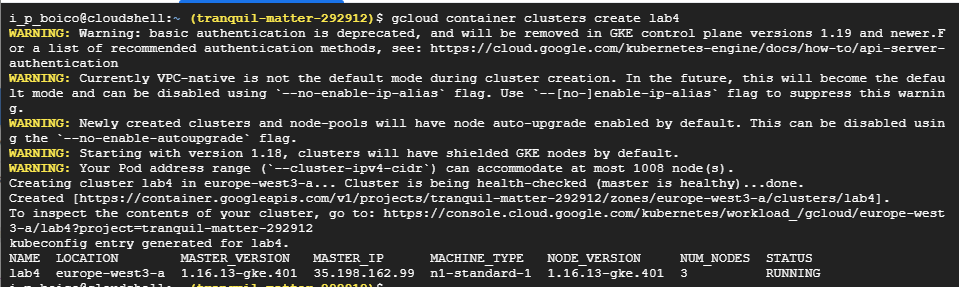
# Создание кластера K8s on CLOUD SHELL

Вводим id нашего проекта в переменную **PROJECT\_ID**

  
  
**export PROJECT\_ID=tranquil-matter-292912**

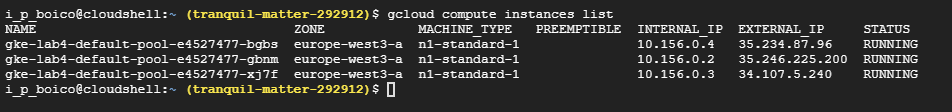
Вводим идентификатор проекта через команду: **gcloud config set project $PROJECT\_ID**Устанавливаем зону (регион) проекта с помощью команды: **gcloud config set compute/zone europe-west3-a**

## Создаем clusterul lab4 с помощью команды: gcloud container clusters create lab4



По умолчанию будут созданы 3 узла.

## Просматриваем информацию об узлах командой: gcloud compute instances list



## Выполнение команд kubectl для автомасштабирования в терминале Cloud Shell

Устанавливаем автоматическое масштабирование под в контроллере репликации с помощью команды:

**$ kubectl autoscale deployment hello-app --cpu-percent = 80 --min = 1 --max = 5** ,

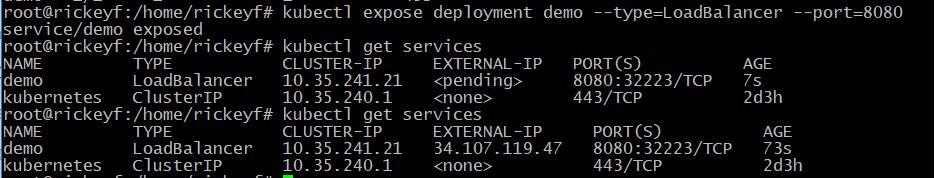
где параметр **--cpu-percent** отвечает за средний процент загрузки процессора, **--min** и **--max** - минимальное и максимальное количество реплик.

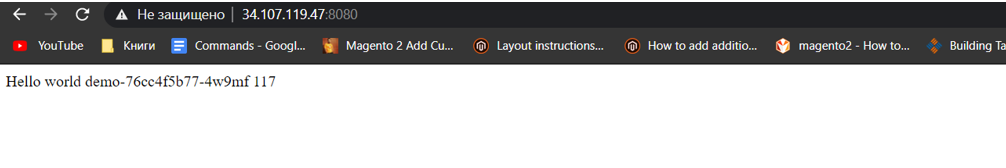


## Cоздаем сервис LoadBalancer

**kubectl expose deployment demo --type=LoadBalancer --port=8080**

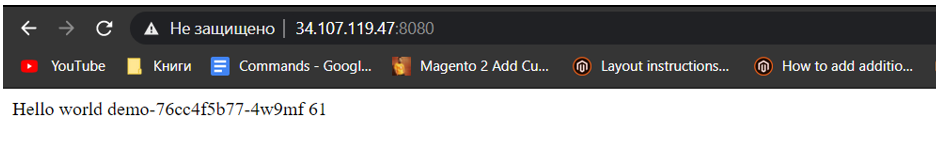
**kubectl get service**

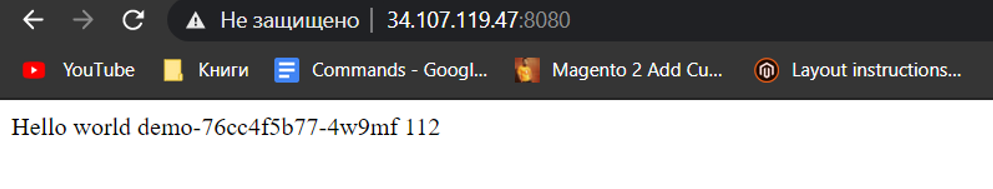
  
  
Включаем kubectl proxy, проверяем приложение:

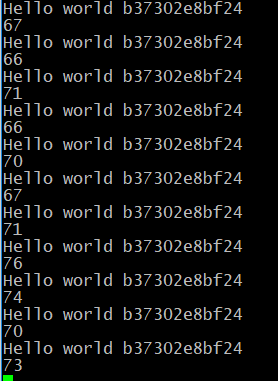
 Запускаем автоматическое масштабирование **kubectl autoscale deployment demo --cpu-percent=15 --min=1 --max=20**

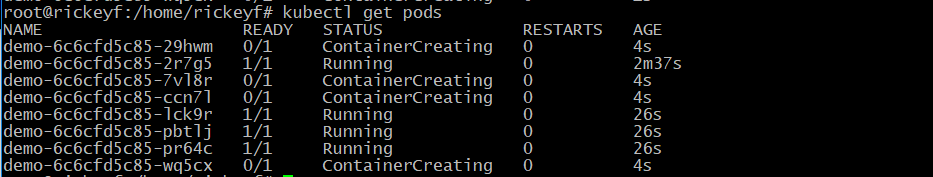
Пере

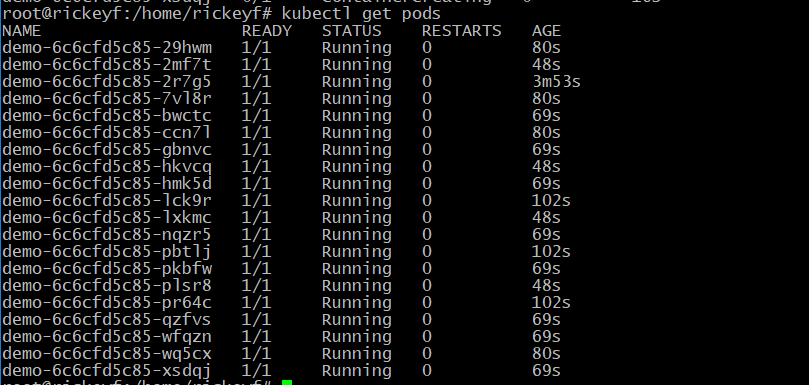
Перегружаем сервис **while true; do curl 127.0.0.1:8080; sleep .009; done**

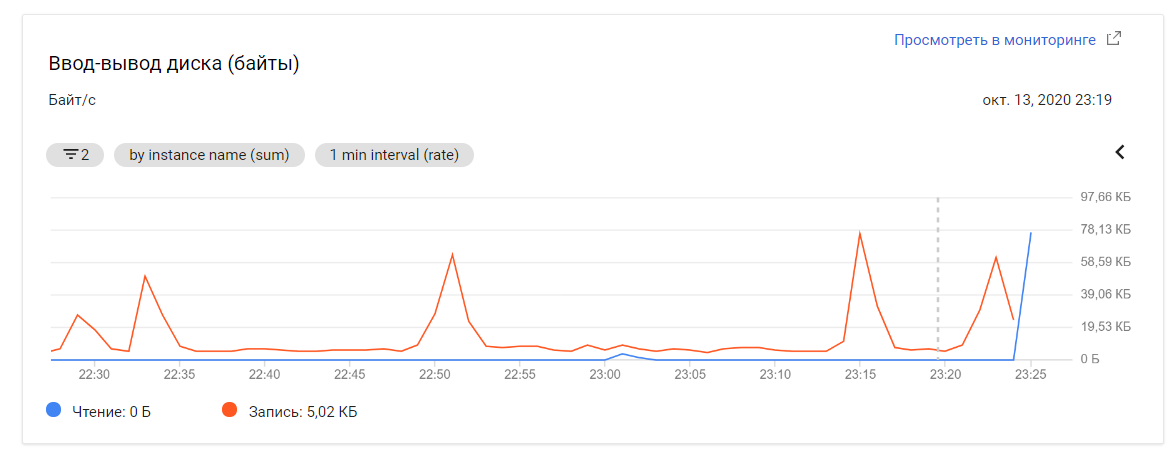
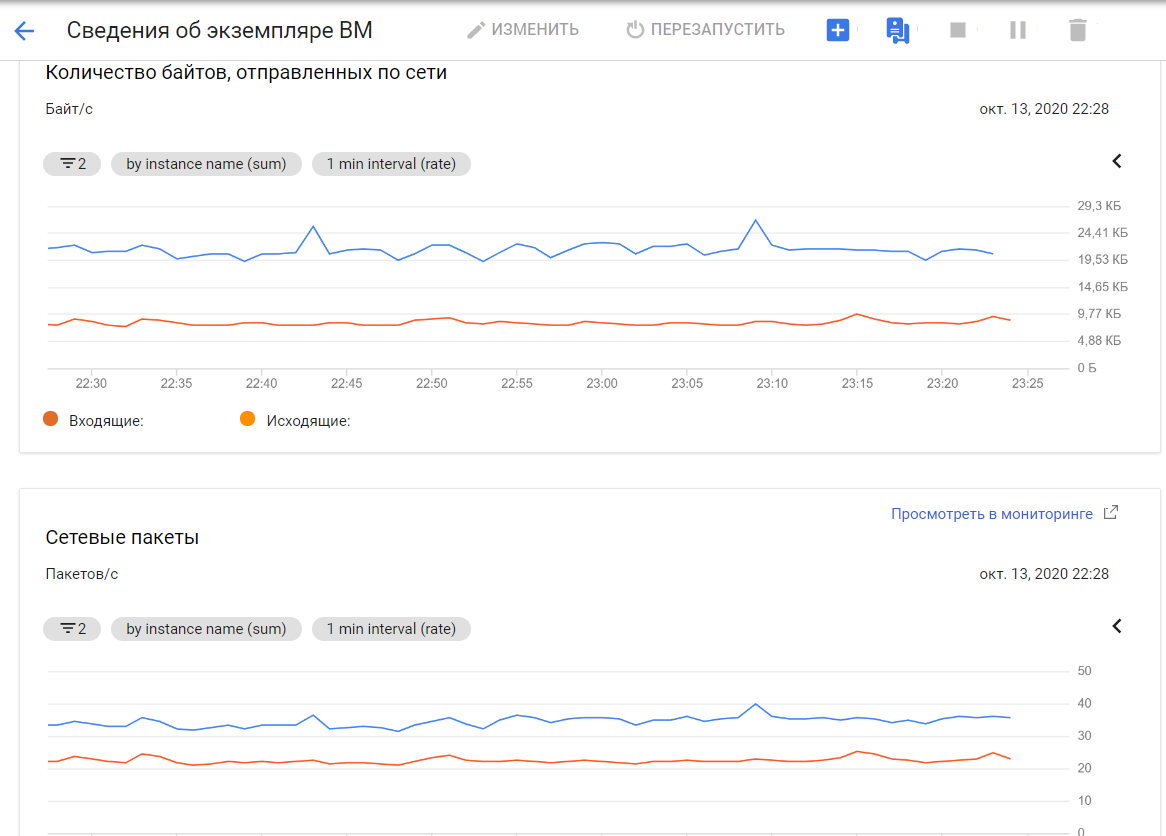
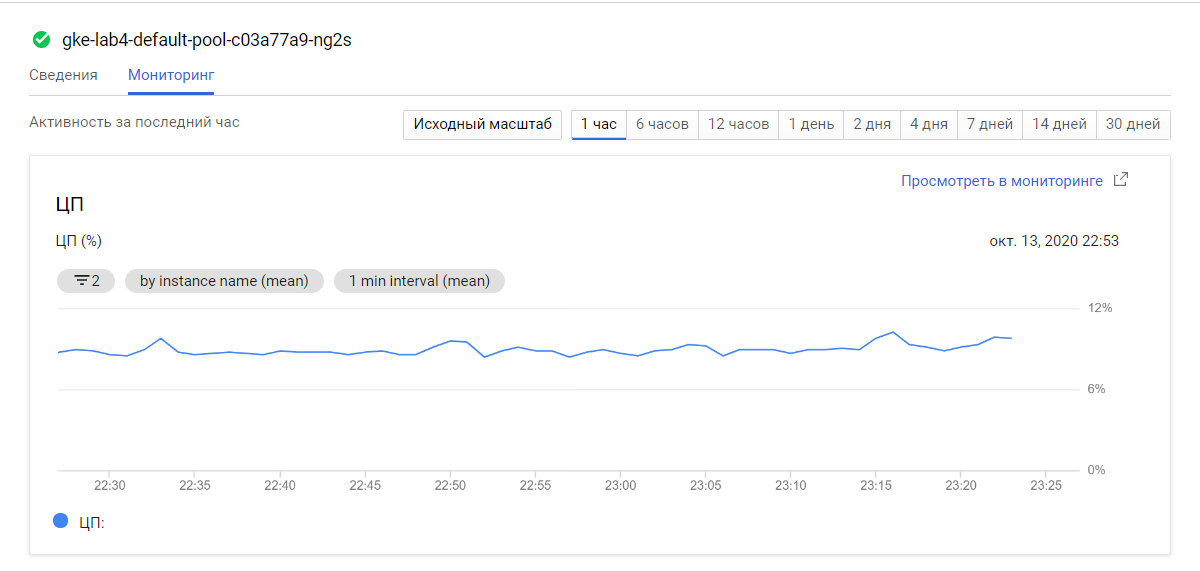












## Примеры автоматического масштабирования

## Exemplul 1. Node js server

В качестве примера мы создадим докер образ для сервера node js

**server2.js**:

'use strict'

const express = require('express');

// Constants

const PORT = 8080;

const HOST = '0.0.0.0';

// App

const app = express();

app.get('/', (req, res) => {

let text = '';

let i;

let date1 = Date.now();

for (let i = 0; i < 5000000; i++) {

Math.sqrt(Math.random());

}

let date2 = Date.now();

let diff = date2 - date1;

var os = require('os');

res.send('Hello world ' + os.hostname() + '\n' + diff + '\n');

});

app.listen(PORT, HOST);

console.log(`Running on http://${HOST}:${PORT}`);

Файл **dockerfile**:

FROM node:10

# Create app directory

WORKDIR /usr/src/app

# Install app dependencies

# A wildcard is used to ensure package.json AND package-lock.json are copied

# where available (npm@5+)

COPY package\*.json ./

RUN npm install

# If you are building your code for production

# RUN npm ci --only=production

# Bundle app source

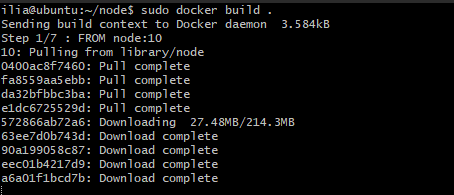
COPY . .

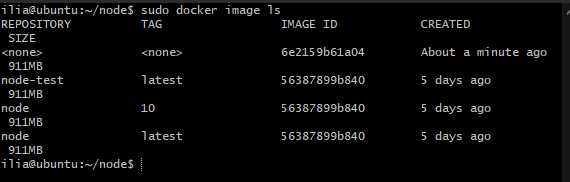
# exposing port 8080

EXPOSE 8080

CMD ["node", "server2.js"]

Создаем образ:





## Продолжение в соответствии с первым примером (node js) из темы 8, а именно:

- Сохранить докер образ в репозиторий Docker Hub

- Создать и проверить работу контейнера на основе вновь созданного образа

- Создание под / деплоймент на кластере K8s на основе созданного образа

- Создание сервиса LoadBalanser

- Вызов созданного приложения из браузера и curl

- Создайте контроллер горизонтального автомасштабирования с помощью *kubectl autoscale*

- Продемонстрировать перегрузку под и увеличить / уменьшить количество реплик, чтобы средняя загрузка ЦП на всех подах оставалась ниже порогового значения

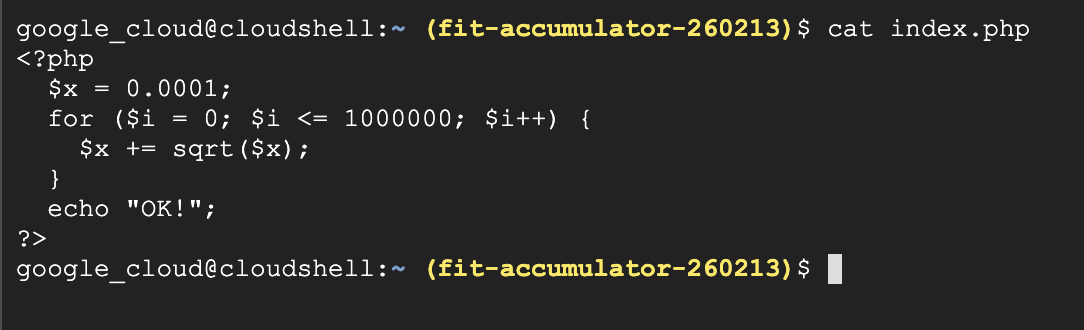
- Анализ полученных результатов

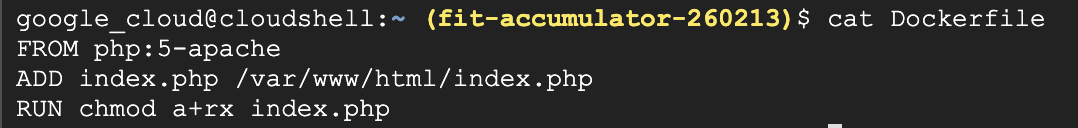
## Exemplul 2. Run & expose php-apache server

Создать котроллер автомасштабирования, используя образ приложения php на сервере Apache.

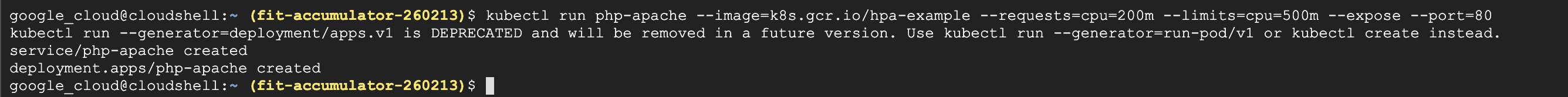
### Выполнение

Файл index.php:

Файл Dockerfile:



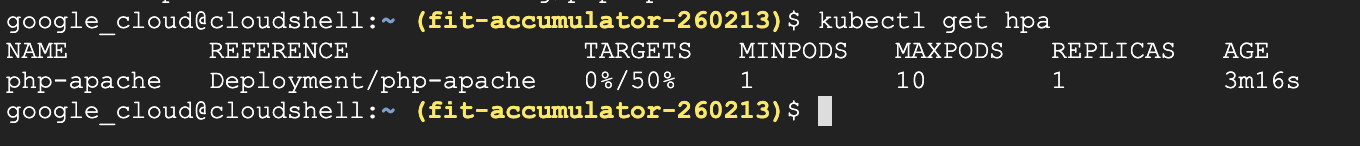
Создание деплойментта на основе созданного докер образа, сохраненного в репозитории DockerHub



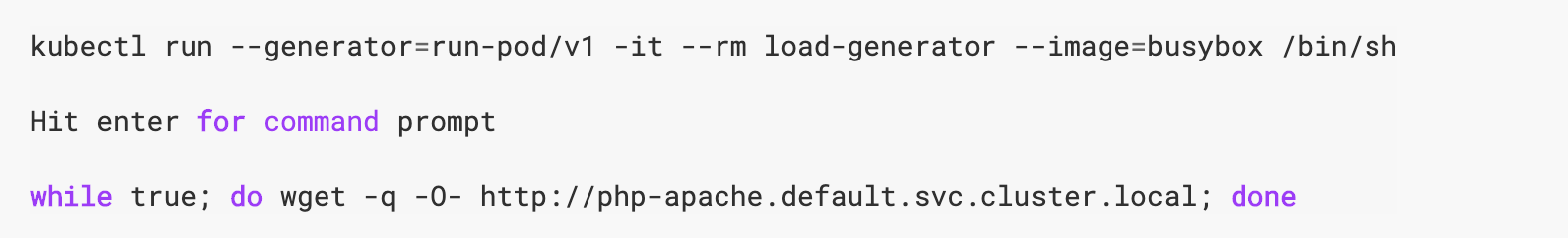
Автоматическое масштабирование под с использованием kubectl autoscale. Следующая команда создаст горизонтальноt автоматическое масштабирование под, которое поддерживает от 1 до 10 реплик под, контролируемых деплойментом php-apache, созданного на первом шаге.

HPA будет увеличивать и уменьшать количество реплик, чтобы поддерживать среднее использование ЦП на всех подах на уровне 50% (поскольку каждый под требует 200 миллиядер на запуск kubectl run, это означает, что среднее использование ЦП составляет 100 m).

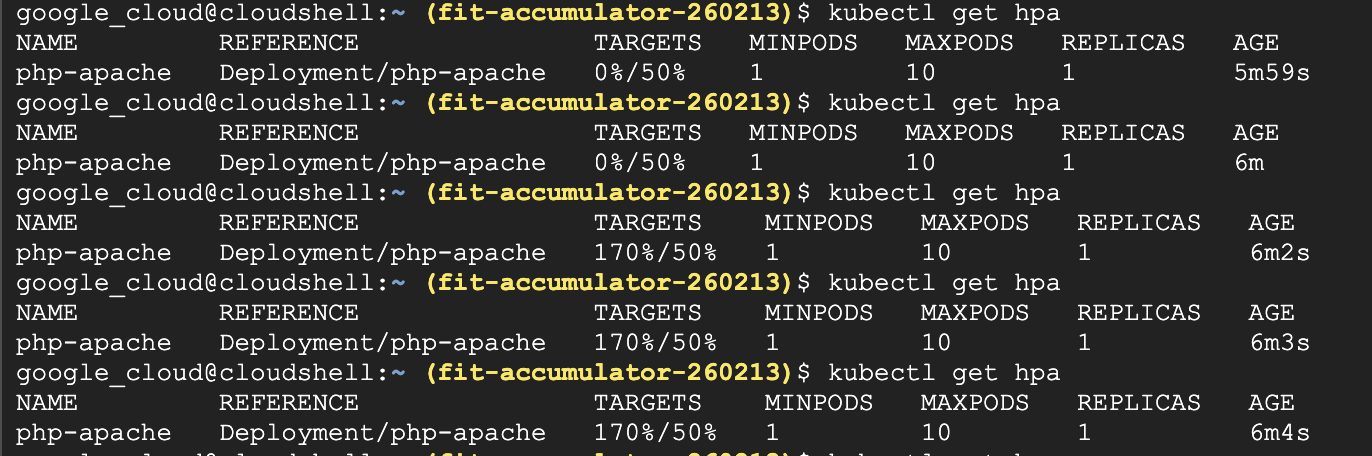
Проверяем текущий статус автомасштабирования командой: **kubectl get hpa**

Видим ответ в терминале:

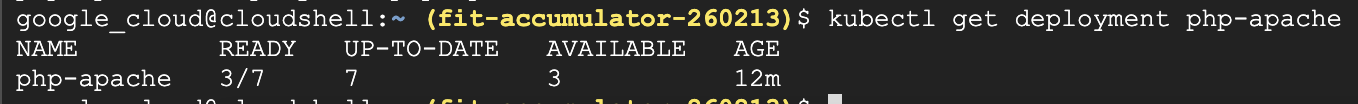
В другом окне оболочки запускаем команды:

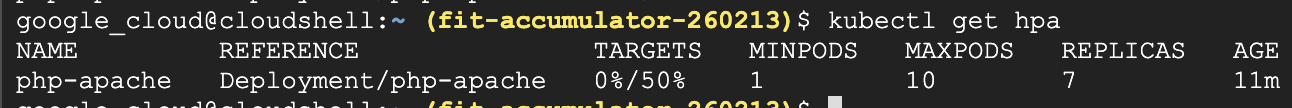


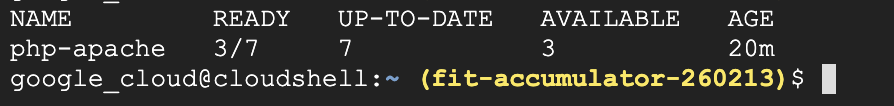
В первом терминале последовательно вводим команду kubectl get hpa, чтобы наблюдать рост нагрузки на процессор.

i

При проверке количества реплик мы замечаем, что развертывание масштабировано до 7 реплик.

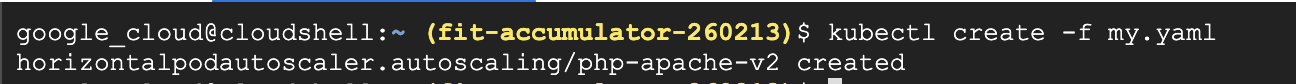


После остановки загрузки со второго окна терминала проверяем состояние автоскалера:

**Кластеру нужно время, чтобы уменьшить количество реплик до 1

### Многопараметрический метод автомасштабирования

Мы создаем файл конфигурации yaml:

** Создаем автомасштабирование через *kubectl create -f my.yaml*

## Продолжение в соответствии со вторым примером (php) из темы 8

- Сохранить докер образ в репозиторий Docker Hub

- Создать и проверить работу контейнера на основе вновь созданного образа

- Создание под / деплоймент на кластере K8s на основе созданного образа

- Создание сервиса LoadBalanser

- Вызов созданного приложения из браузера и curl

- Создайте контроллер горизонтального автомасштабирования с помощью *kubectl autoscale*

- Продемонстрировать перегрузку под и увеличить / уменьшить количество реплик, чтобы средняя загрузка ЦП на всех подах оставалась ниже порогового значения

- Анализ полученных результатов

# ВЫВОДЫ

# БИБЛИОГРАФИЯ

1. [Quickstart: Getting started with Cloud SDK | Cloud SDK Documentation](https://cloud.google.com/sdk/docs/quickstart)
2. [Google Cloud Platform (part III): Развертывание Kubernetes кластера в облаке](https://vk.com/@progmeme-google-cloud-platform-part-iii-razvertyvanie-kubernetes-klas)