Государственный Университет Молдовы

Факультет Математики и Информатики

Департамент Информатики

“Inițiere în Cloud Computing și Sisteme de Operare Distribuite”

Лабораторная работа 2   
“Git. Docker.”

Проверил: Препелица Аурелия

Выполнил: Чобану Артём

Кишинев 2021

# Требования к лабораторной работе:

* Создание учётной записи, репозитория на докер-хабе и установка Docker
* Создание учётной записи на GitHub, установка Git.
* Выполнения примеров создания образов, контейнеров и их запуска в K8s на основе примеров Темы 3 курса.

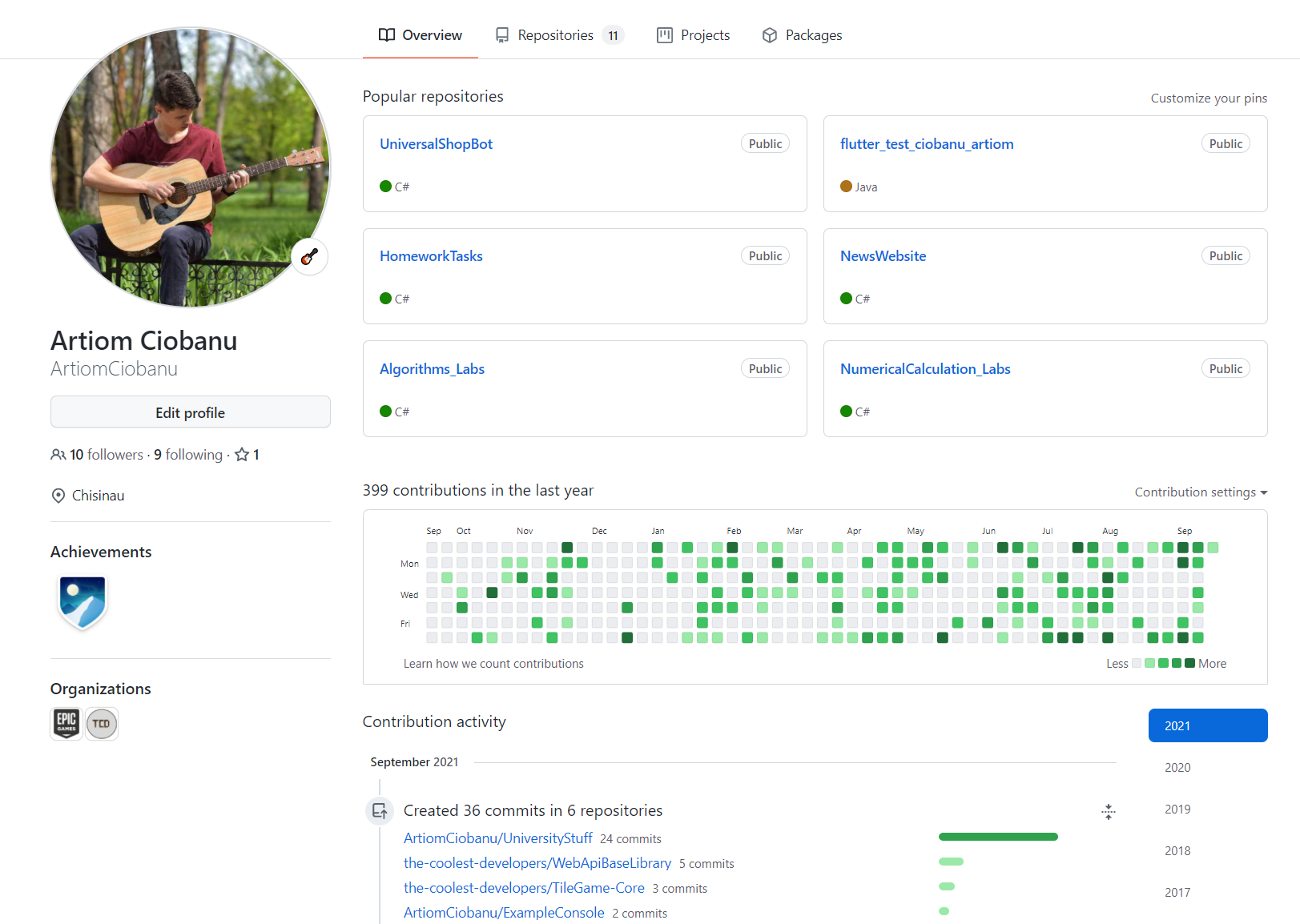
# Выполнение лабораторной работы

Моя учётная запись на GitHub существует уже давно:

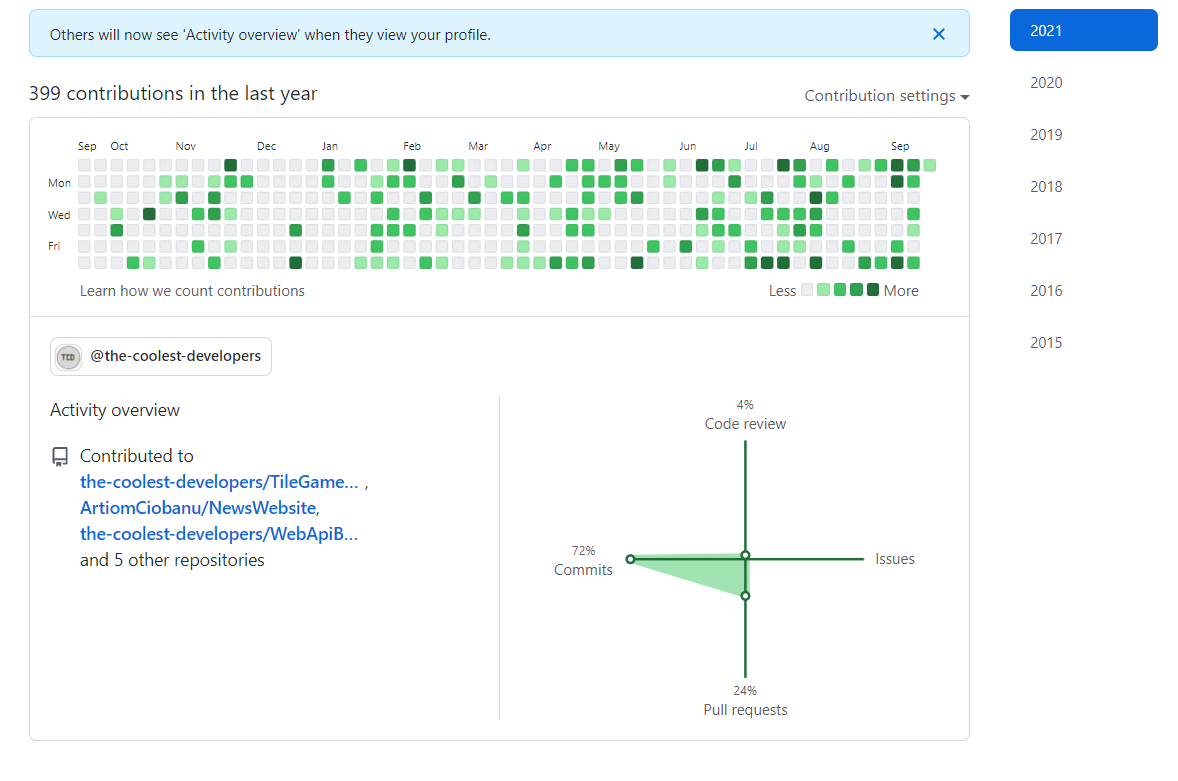
<https://github.com/ArtiomCiobanu>

Также есть созданная организация для совместных проектов с друзьями:

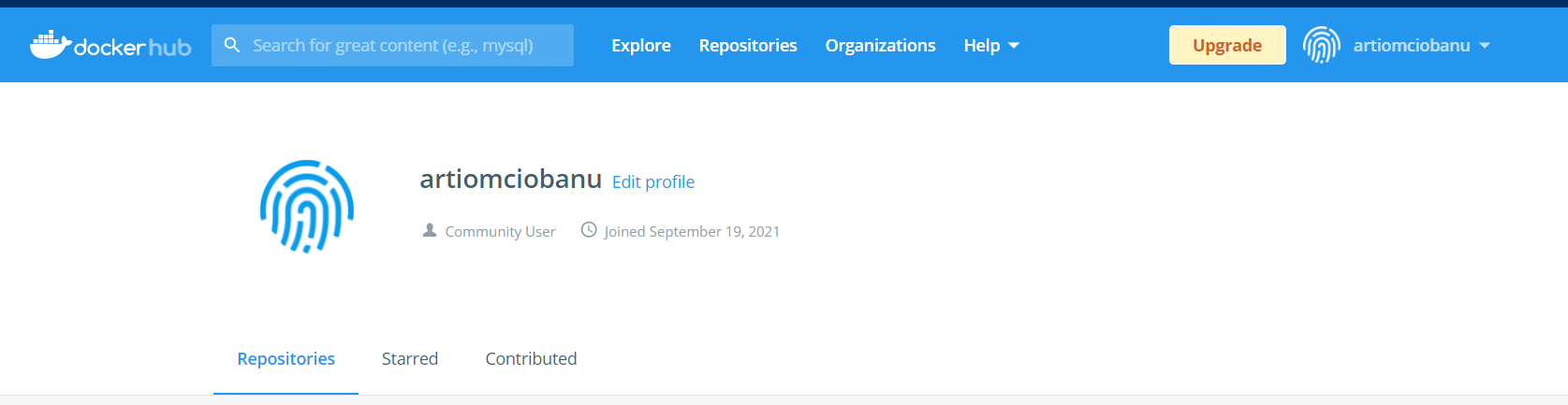
<https://github.com/the-coolest-developers>



Моя активность на GitHub за 2021 год:



Созданная учётная запись Docker Hub:



# Создание контейнеров, образов и их запуск в Kubernetes.

Для контейнеризации я буду использовать простой проект ASP.NET Web API приложения.

Dockerfile для созданного приложения:

#Облегнённый образ ASP.NET

FROM mcr.microsoft.com/dotnet/aspnet:5.0-alpine3.12 AS base

#Установим рабочую директорию

WORKDIR /app

EXPOSE 80

EXPOSE 12345

#Облегчённый образ .NET 5.0 SDK

FROM mcr.microsoft.com/dotnet/sdk:5.0-alpine3.13 AS publish

WORKDIR /src

#Скопируем файлы в контейнер

COPY . .

#Команда dotnet publish – сборка и копирование всех зависимостей в папку проекта

RUN dotnet publish "Cloud\_Lab2.csproj" -c Release -o /app/publish

FROM base AS final

WORKDIR /app

COPY --from=publish /app/publish .

#Устанавливаем .dll проекта как точку входа

ENTRYPOINT dotnet Cloud\_Lab2.dll

Запустим сборку образа командой docker build с тегом “test:test”

Text

Description automatically generated

Запустим приложение в контейнере. Порт контейнера 12345 будет перенаправляться на порт приложения – 12346.

Text

Description automatically generated

Работа приложения:

Graphical user interface, text, application

Description automatically generated

Контейнер:

Graphical user interface, application

Description automatically generated

Text

Description automatically generated

Получим список образов:

Text

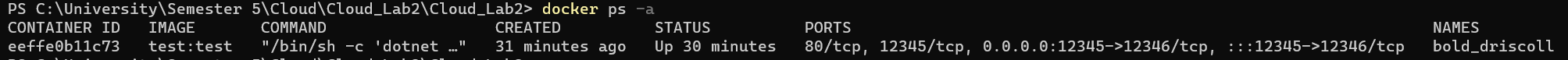
Description automatically generated

Дадим новый тег образу test:test

Text

Description automatically generated

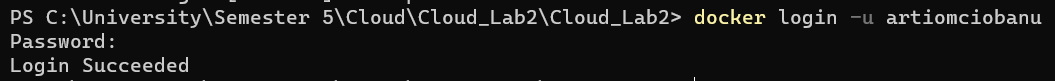
Выведем список контейнеров:



Text

Description automatically generated

Войдём в аккаунт с помощью docker login:



Созданный репозиторий на Docker Hub:

Graphical user interface, text, application, email

Description automatically generated

Изменим тег образа и отправим на Docker Hub:

Text

Description automatically generated

Как видим, образ появился на Docker Hub:

Graphical user interface, text, application, email, website

Description automatically generated

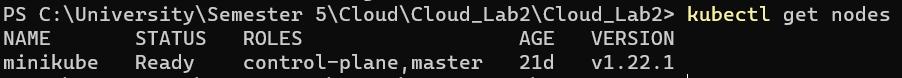
# Развёртывание в Minikube:

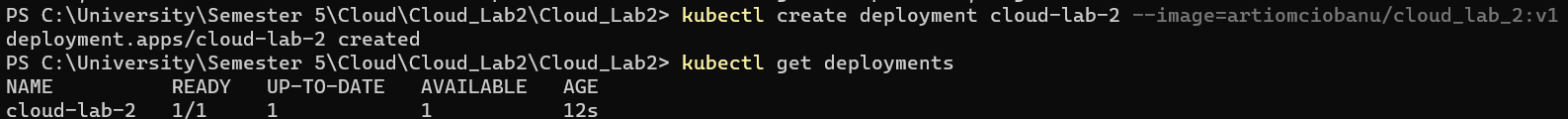
Запустим minukube:

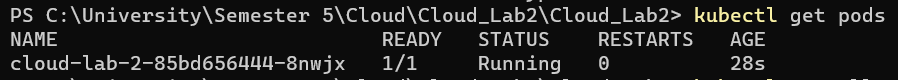
Text

Description automatically generated

Узлы:

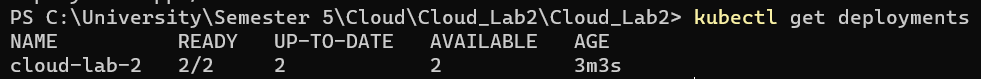




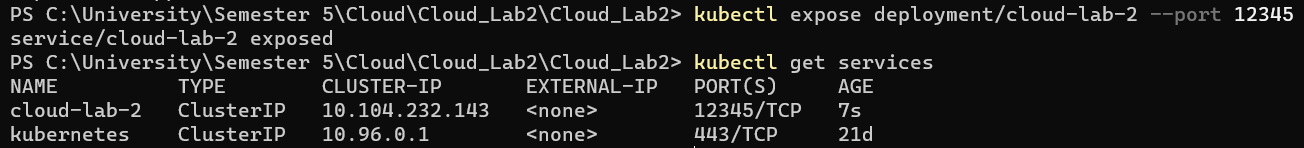


Масштабирование





Откроем приложение для порта 12345



Graphical user interface, text, website

Description automatically generated

# Вывод

Во время работы над данной лабораторной работой я научился работать с докером, создавать образы и запускать контейнеры. В частности, делал это для приложений ASP.NET Core. Кроме того, я научился создавать кластер на основе образов докера, и развёртывать приложение в нём.

# Библиография

<https://docs.docker.com/engine/reference/commandline/ps/>

<https://docs.docker.com/engine/reference/run/>

<https://docs.docker.com/get-started/overview/>

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/configuration/?view=aspnetcore-5.0>

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/servers/kestrel/endpoints?view=aspnetcore-5.0>