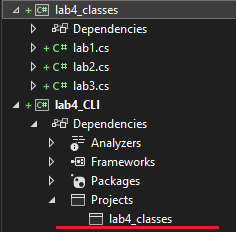
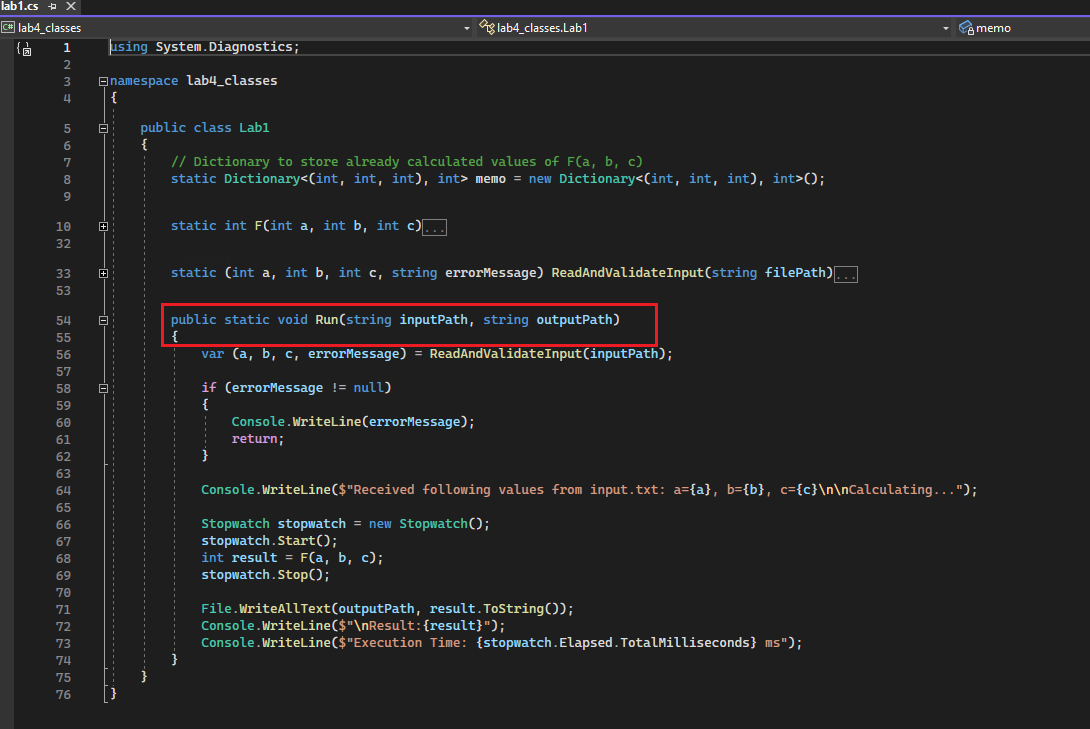
|  |  |  |  |
| --- | --- | --- | --- |
| **МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  **КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка**  **ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**  **Кафедра програмних систем і технологій**  **Дисципліна**  **«Кросплатформне програмування»**  **Лабораторна робота 4** | | | |
| **Виконав:** | Чушенко Ярослав Володимирович | **Перевірив**: | Петрівський В.Я |
| Група | ІПЗ-41 | Дата перевірки |  |
| Форма навчання | денна | Оцінка |  |
| Спеціальність | 121 |
| 2023 | | | |

**Загальний варіант – 54**Репозиторій на Github: [Посилання](https://github.com/PingvinAustr/CrossplatformKNU)  
 **Умова завдання:**Створити консольний застосунок, що задовольнятиме наступним вимогам.

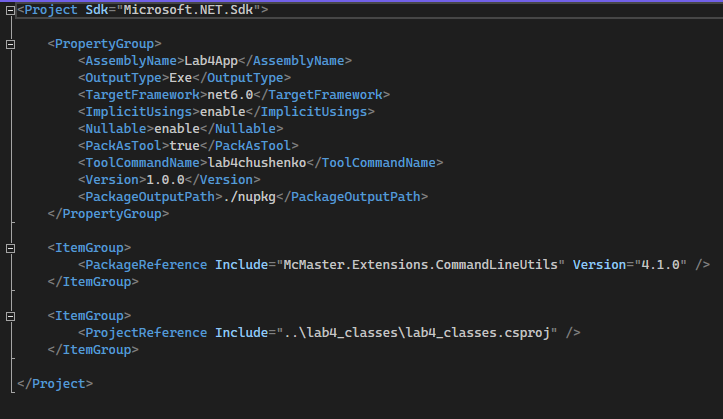
1. Складається з двох проектів
   1. Безпосередньо консольний додаток
   2. Бібліотека класів, що дає змогу запускати практичні 1, 2 або 3
2. Пакується як Nuget застосунок (dotnet tool) та публікується у приватному Nuget репозитарії
3. Має настпуний консольний інтерфейс
   1. При передачі команди “version” виводить інформацію про программу
      1. Автора
      2. Версію
   2. При передачі команди “run” очікує підкоманду “lab1”, “lab2” або “lab3” для запуску відповідної практичної
      1. Має необовязкові параметри “-I” або “--input” інпут файл та “-o” або “--output” аутпут файл.
   3. Пре передачі команди “set-path” очікує обовязковий параметр “-p” або “--path”, що задає шлях до папки з інпут та аутпут файлами. Отриманий шлях висталяється в змінну середовища на ім’я “LAB\_PATH”
4. Пріорітетність шляху
   1. Якщо шлях до файлу заданий параметрами консолі то саме його слід застосувати
   2. Якщо параметри консолі не задані то слід перевірити значення змінної “LAB\_PATH”
   3. Якщо a. та b. не задано то слід пошукати файли спочатку в домашній директорії користувача
   4. Якщо умови a. ,b. , с. не допомгли знайти інпут файл вивести помилку.
5. *Всі інші вводи в консольний застосунок мають бути проігноровані і вивдена підказка, щодо правильних команд и синтаксису.*
6. Застосунок має бути розгорнутий на 3-х операційних системах Linux, Mac, Windows за допомогою віртуальних машин побудованих за допомогою Vagrant файлів. Для інсталяції слід використати можливості vagrant provision. Весь програмний код для створення віртуальних машин та запуску процесу інсталяції слід додати в репозиторій.
7. Для приватного Nuget репозитарію використати BaGet. (його завантажувати не потрібно). Vagrant provision має сконфігурувати доступ до приватного репозитарія і виконати необхідні команди для інсталяції пакету.
8. Імя пакету має бути ваше імя перша літера та прізвище латинською.
9. **Під час захисту потрібно буде показати розгорнуті віртуаьлні машини і те що вони працюють з Vagrant.**

**Виконання:  
  
Перший крок**

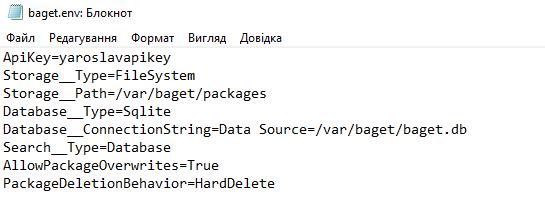
Створимо 2 нових проекта – lab4\_classes, lab4\_CLI й додамо в консольний застосунок референс на бібліотеку класів:  


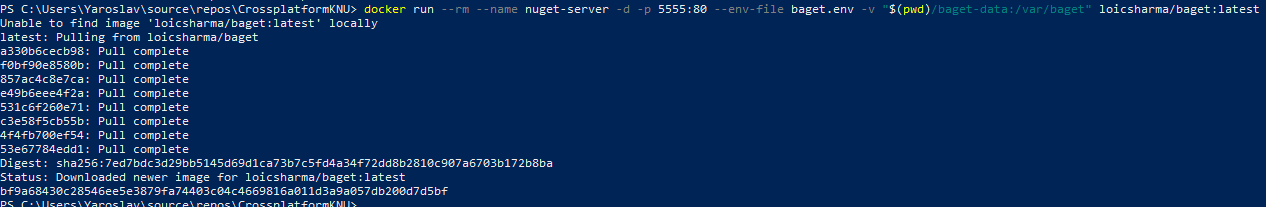
В код lab1,lab2,lab3 було внесено незначні зміни (метод Main перетворено на статичний Run й додано вхідні параметри для input/output файлів).  
  
Приклад:  
   
  
**Повний код проекту можна переглянути на GitHub (**[Посилання](https://github.com/PingvinAustr/CrossplatformKNU))**, щоб не перевантажувати звіт зайвим текстом.**

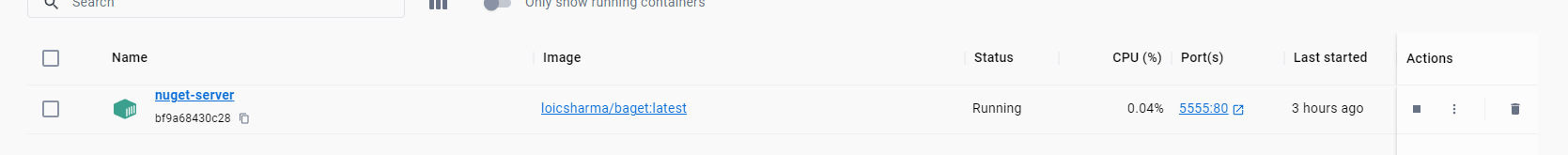
**Другий крок**

Змінимо lab4\_CLI.csproj щоб пакувати застосунок як dotnet tool:  


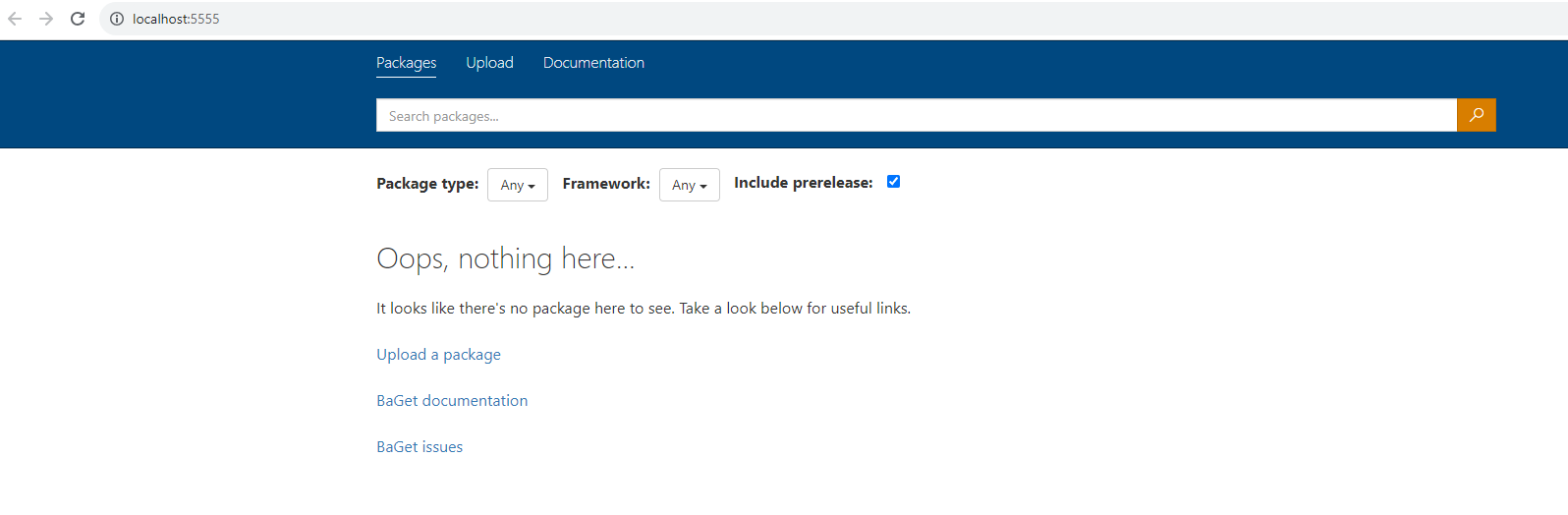
**Третій крок:**

Створимо приватний локальний репозиторій за допомогою Baget.  
  
Створимо baget.env: ****  
Запустимо його за допомогою docker:  
docker run --rm --name nuget-server -d -p 5555:80 --env-file baget.env -v "$(pwd)/baget-data:/var/baget" loicsharma/baget:latest

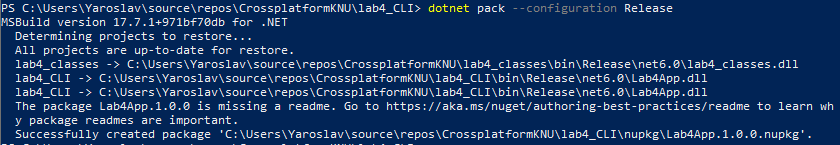
****

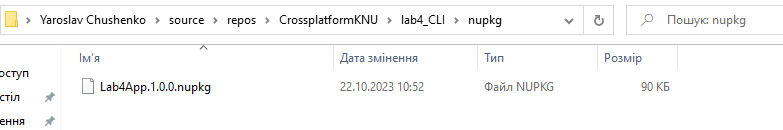
Вдало створився відповідний контейнер:  
  


Перейдемо за адресою <http://localhost:5555/> або ж <http://192.168.0.191:5555/> й можемо побачити, що репозиторій створено.

**  
  
Третій крок – додати пакет до приватного репозиторію**Запакуємо створений застосунок командою:

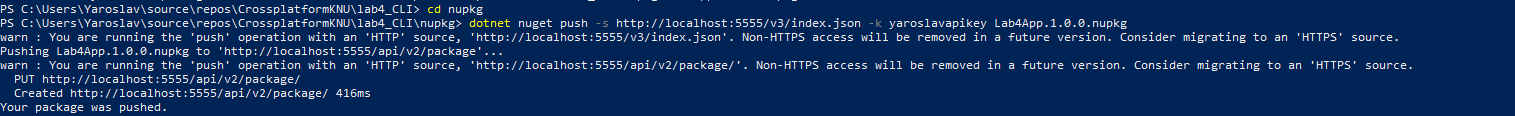
dotnet pack --configuration Release

****В файловій системі створився файл пакету:

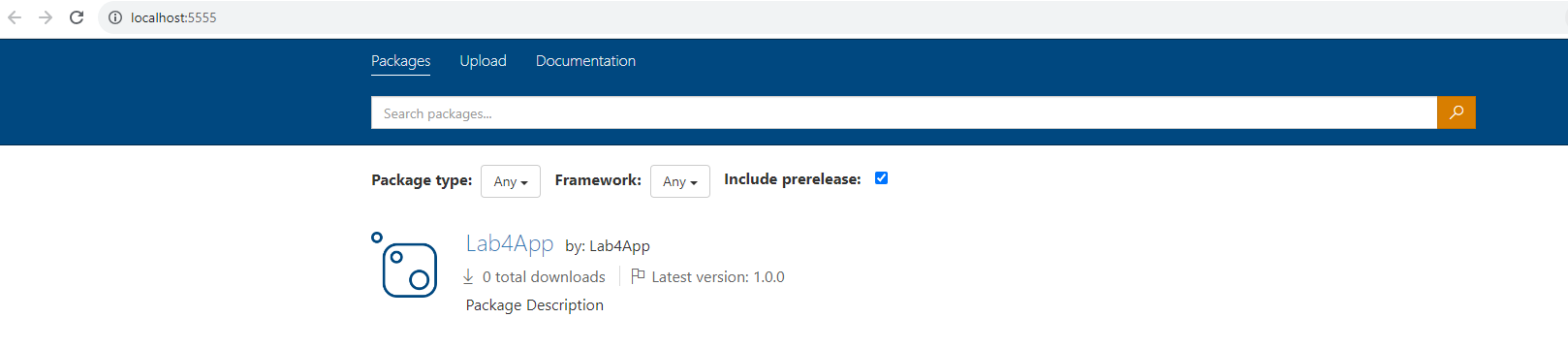
****

Виконаємо команду

dotnet nuget push -s [http://localhost:5555/v3/index.json -k yaroslavapikey Lab4App.1.0.0.nupkg](http://localhost:5555/v3/index.json%20-k%20yaroslavapikey%20Lab4App.1.0.0.nupkg) для відправлення пакету в репозиторій

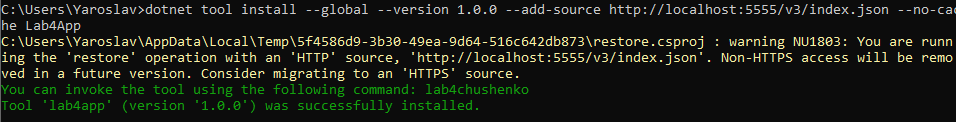
****

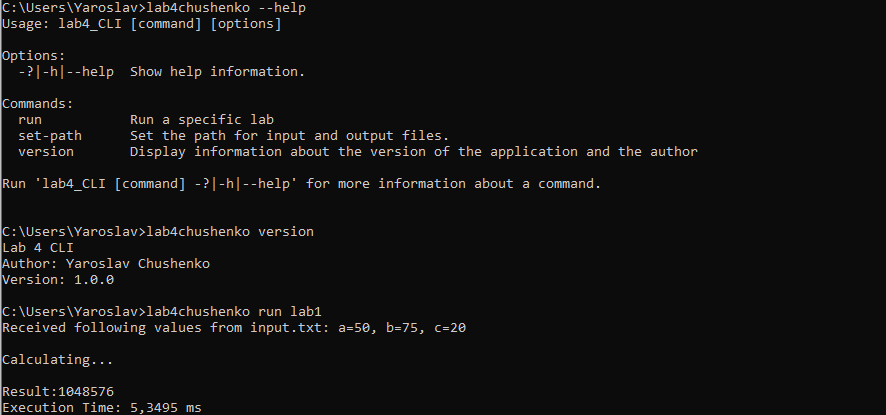
Перейдемо на локальну адресу й бачимо, що пакет відправлено вдало.

****

**Четвертий крок – встановити dotnet tool на локальну машину для тестування.**Виконаємо команду  
dotnet tool install --global --version 1.0.0 --add-source http://localhost:5555/v3/index.json --no-cache Lab4App

**Застосунок встановлено:**

****

**Виконаємо декілька тестових команд:  
**

**П’ятий крок – розгортання віртуальних машин за допомогою Vagrant.**

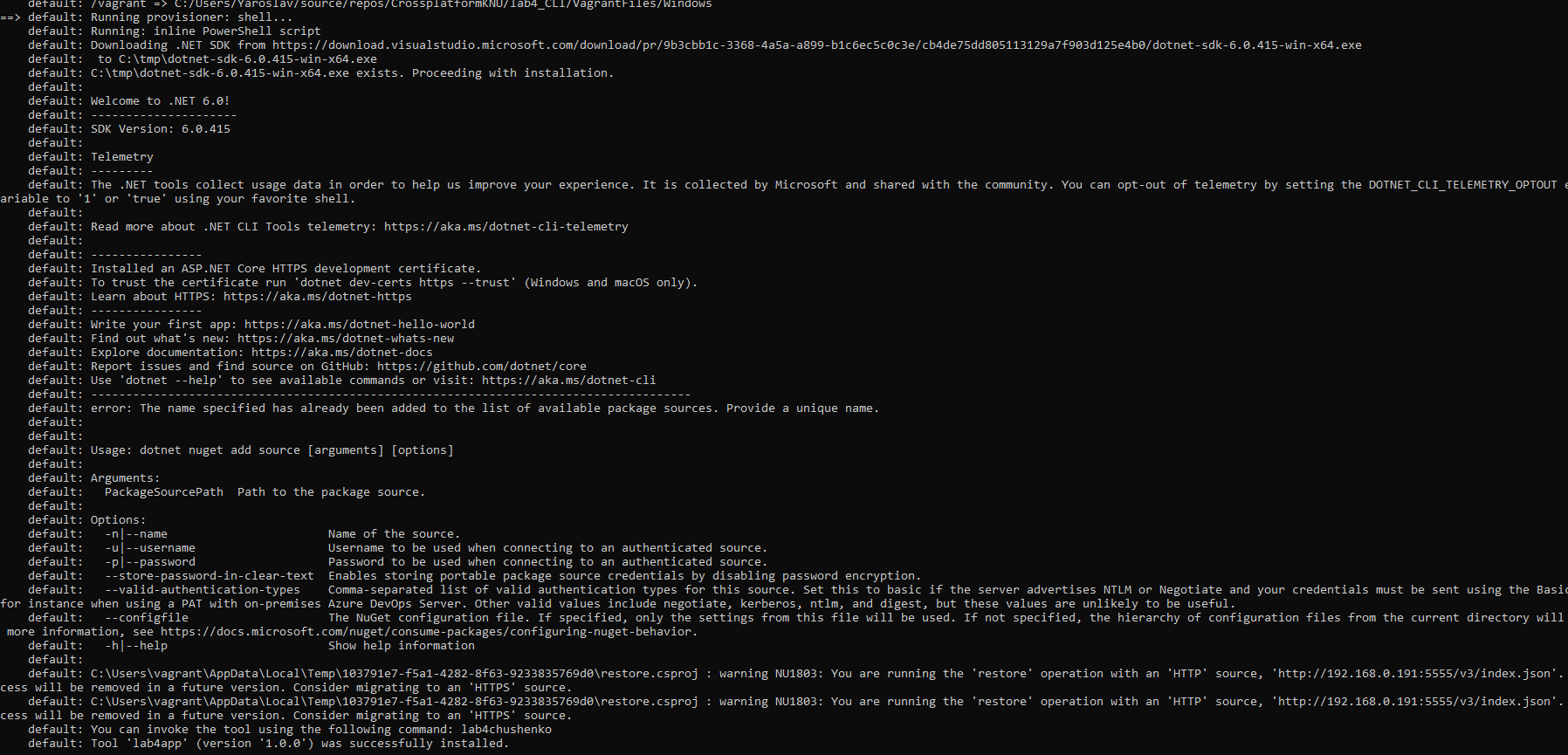
**Створимо Windows Vagrantfile:**

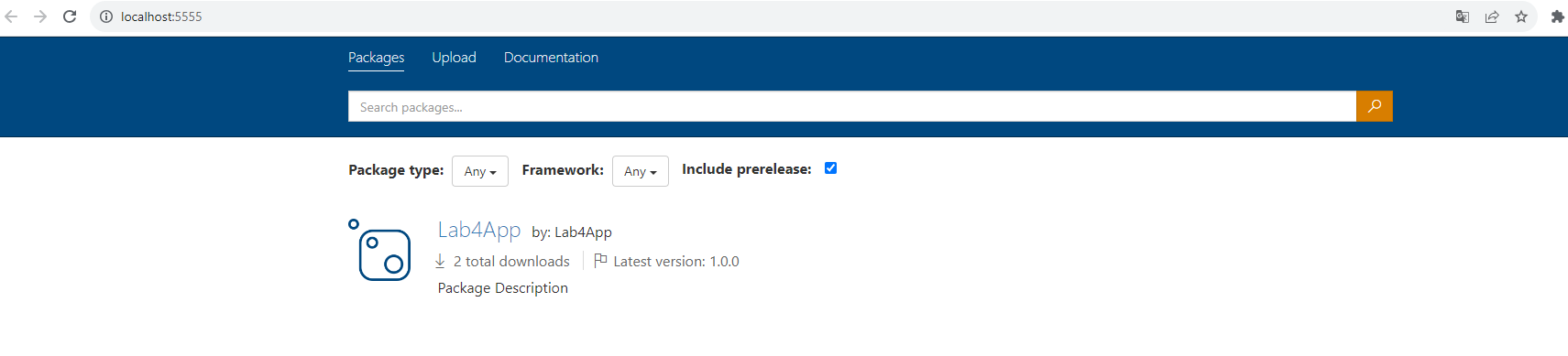
1. Vagrant.configure("2") do |config|
2. config.vm.box = "gusztavvargadr/windows-10"
3. config.vm.provider "virtualbox" do |vb|
   1. vb.gui = true
   2. vb.memory = "1024"
4. end
5. config.vm.network "public\_network"
6. config.vm.provision "shell", privileged: true, inline: <<-SHELL
7. # Ensure PowerShell is set to use TLS1.2
8. [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
9. # Specify the URL for the .NET Core SDK
10. $dotnetSdkUrl = "https://download.visualstudio.microsoft.com/download/pr/9b3cbb1c-3368-4a5a-a899-b1c6ec5c0c3e/cb4de75dd805113129a7f903d125e4b0/dotnet-sdk-6.0.415-win-x64.exe
11. "
12. $installerPath = "C:\\tmp\\dotnet-sdk-6.0.415-win-x64.exe"
13. Write-Host "Downloading .NET SDK from $dotnetSdkUrl to $installerPath"
14. # Download the installer
15. $webClient = New-Object System.Net.WebClient
16. $webClient.DownloadFile($dotnetSdkUrl, $installerPath)
17. # Check if the installer was downloaded
18. if (Test-Path $installerPath) {
    * 1. Write-Host "$installerPath exists. Proceeding with installation."
      2. Start-Process -Wait -FilePath $installerPath -ArgumentList "/quiet /install"
      3. Remove-Item $installerPath
19. } else {
    * 1. Write-Host "Failed to download .NET SDK from $dotnetSdkUrl"
      2. exit 1
20. }
21. $dotnetExePath = "C:\\Program Files\\dotnet\\dotnet.exe"
22. $dotnetVersion = & $dotnetExePath --version
23. # Add NuGet source for the tool
24. & $dotnetExePath nuget add source http://192.168.0.191:5555/v3/index.json -n "LocalNuGet"
25. # Install the tool
26. & $dotnetExePath tool install --global --version 1.0.0 --add-source http://192.168.0.191:5555/v3/index.json --no-cache Lab4App
27. SHELL

End

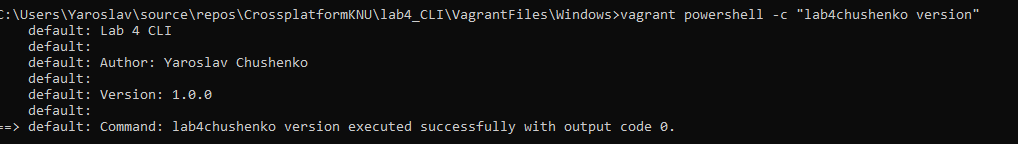
Запустимо машину через vagrant up й використаємо provision.

Бачимо повідомлення про вдалу встановку застосунку на віртуальну машину:

****  
  
Також бачимо, що на локальній машині змінилась кількість завантажень пакету:

****

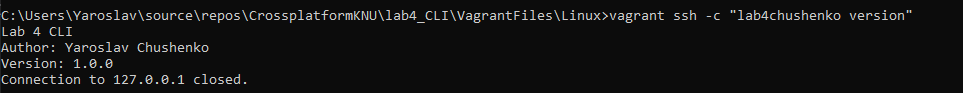
Перевіримо чи працює застосунок на віртуальній машині через команду vagrant powershell.  
  
Бачимо, що програма вдало працює:

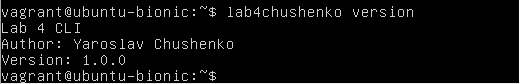
****

**Створимо Linux Vagrantfile.**

1. Vagrant.configure("2") do |config|
2. config.vm.box = "generic/ubuntu1804"
3. config.vm.network "forwarded\_port", guest: 80, host: 5000
4. config.vm.box\_download\_insecure = true
5. config.vm.provider "virtualbox" do |vb|
6. vb.gui = false
7. vb.memory = 2048
8. end
9. config.vm.provision "shell", inline: <<-SHELL
10. sudo apt-get update
11. sudo apt-get install -y apt-transport-https
12. # Install the .NET SDK
13. # Update and install necessary components
14. sudo apt update
15. sudo apt install -y apt-transport-https wget curl
16. wget https://packages.microsoft.com/config/ubuntu/18.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
17. sudo dpkg -i packages-microsoft-prod.deb
18. sudo apt update
19. sudo apt install -y dotnet-sdk-6.0
20. echo 'export PATH=$PATH:$HOME/.dotnet/tools' >> /home/vagrant/.bash\_profile
21. # Add your local Windows machine's IP address and port where NuGet packages are hosted
22. su - vagrant -c "dotnet nuget add source -n YChushenko http://192.168.0.191:5555/nuget/v3/index.json"
23. # Install your custom NuGet package
24. su - vagrant -c "dotnet tool install --global --version 1.0.0 --add-source http://192.168.0.191:5555/v3/index.json --no-cache Lab4App"
25. SHELL

End  
  
Запустимо віртуальну машину, застосуємо provisions й перевіримо чи працює застосунок:

****

****

**Mac Vagrantfile:**

1. Vagrant.configure("2") do |config|
2. config.vm.box = "jhcook/osx-elcapitan-10.11"
3. config.vm.network "forwarded\_port", guest: 80, host: 5000
4. config.vm.synced\_folder ".", "/vagrant", :disabled => true
5. config.ssh.insert\_key = false
6. config.vm.provider "virtualbox" do |vb|
7. vb.memory = 2048
8. vb.customize ["modifyvm", :id, "--cpuidset", "1","000206a7","02100800","1fbae3bf","bfebfbff"]
9. vb.customize ["setextradata", :id, "VBoxInternal/Devices/efi/0/Config/DmiSystemProduct", "MacBookPro11,3"]
10. vb.customize ["setextradata", :id, "VBoxInternal/Devices/efi/0/Config/DmiSystemVersion", "1.0"]
11. vb.customize ["setextradata", :id, "VBoxInternal/Devices/efi/0/Config/DmiBoardProduct", "Iloveapple"]
12. vb.customize ["setextradata", :id, "VBoxInternal/Devices/smc/0/Config/DeviceKey", "ourhardworkbythesewordsguardedpleasedontsteal(c)AppleComputerInc"]
13. vb.customize ["setextradata", :id, "VBoxInternal/Devices/smc/0/Config/GetKeyFromRealSMC", "1"]
14. end
15. config.vm.provision "shell", inline: <<-SHELL
16. brew update
17. brew install wget
18. # Install the .NET SDK
19. wget https://packages.microsoft.com/config/homebrew/20.04/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
20. dpkg -i packages-microsoft-prod.deb
21. brew update
22. brew install --cask dotnet-sdk
23. echo 'export PATH=$PATH:$HOME/.dotnet/tools' >> /Users/vagrant/.bash\_profile
24. # Add your NuGet package source
25. su - vagrant -c "dotnet nuget add source -n YChushenko http://192.168.0.191:5555/nuget/v3/index.json"
26. # Install your custom NuGet package
27. su - vagrant -c "dotnet tool install --global --version 1.0.0 --add-source http://192.168.0.191:5555/v3/index.json --no-cache Lab4App"
28. SHELL
29. end