.NET CORE AZURE UBUNTU VM DEPLOY GUIDE

PETRO KOLOSOV

Abstract. Simple and easy way to deploy your .NET Core web application to the Azure Ubuntu-based virtual machine.

Contents

1. Virtual machine creation

Firstly, it is necessary to create a virtual machine (unexpectedly) where deployment to be hosted on. In this guide is considered free virtual machine of type `Standard B1ms (1 vcpu, 2 GiB memory)` with Ubuntu 20.04 operating system. Definitely it won't be considered step by step creation in this document, however required VM parameter are as follows:

- Size: `Standard B1ms (1 vcpu, 2 GiB memory)`
- OS: `Ubuntu Server 20.04 LTS - Gen2`
- Availability options: `No infrastructure required`
- Authentication type: `SSH public key`

Date: October 31, 2022.

2010 *Mathematics Subject Classification.* 26E70, 05A30.

Key words and phrases. Azure, DevOps, Virtual machine, Deploy, Nginx, SSH, CI/CD, Azure pipelines, Github actions .

- SSH public key source: `Use existing public key (create it before you created VM)`
- Public inbound ports: `HTTP(80), HTTPS(443), SSH(22)`
- OS disk type: `Standard SSD`
- Encryption type: `Default`
- Public IP: `Basic SKU, Static (be sure to create static IP)`
- Select inbound ports: `HTTP(80), HTTPS(443), SSH(22)`
- Boot diagnostics: `Disabled`

Chosen parameters of the virtual machine are collected in order to minimize vm's cost. If you are not sure, refer to the screenshots via the reference [Kol22c].

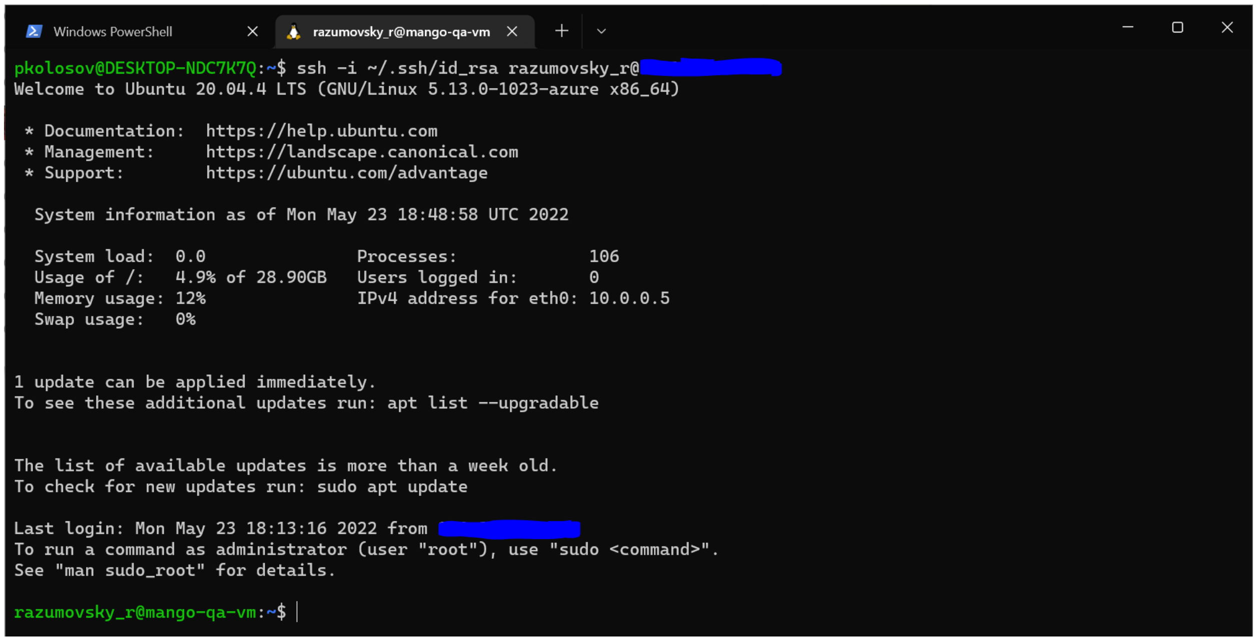
2. Connect to VM via SSH

In order to configure virtual machine manually (as this guide tends to describe), we have to connect to it via SSH using the specified RSA private and public key-pair. It is assumed that programmer uses `WSL2` under `Windows 10` in order to work with VM via the SSH. By default, SSH keys are stored under the path `c/Users/username/.ssh`. Assume that RSA key-pair is stored there and have the names `id_rsa` and `id_rsa.pub` for private and public keys respectively. In order to interact the VM via SSH it is necessary to copy RSA keypair to the WSL `username/.ssh` folder, we use the commands under WSL

- `cp /mnt/c/Users/pkolosov/.ssh/id_rsa /.ssh/`
- `cp /mnt/c/Users/pkolosov/.ssh/id_rsa.pub /.ssh/`

Then connection is available now using the command

- `ssh -i /.ssh/id_rsa razumovsky_r@MachineStaticIP`



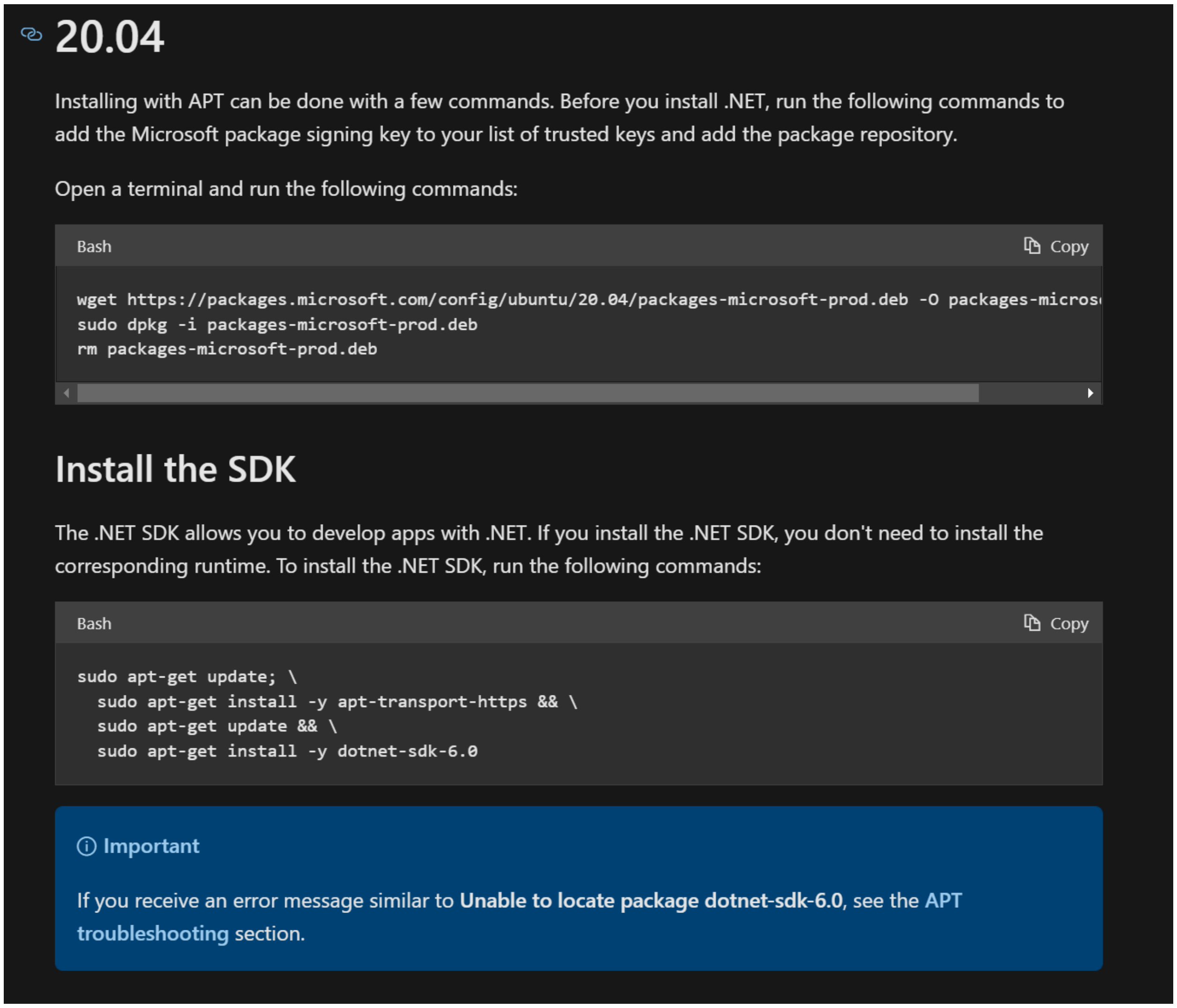
```
pkolosov@DESKTOP-NDC7K7Q:~$ ssh -i ~/.ssh/id_rsa razumovsky_r@[redacted]
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-1023-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon May 23 18:48:58 UTC 2022

  System load:  0.0               Processes:             106
  Usage of /:   4.9% of 28.90GB   Users logged in:       0
  Memory usage: 12%               IPv4 address for eth0: 10.0.0.5
  Swap usage:   0%


1 update can be applied immediately.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon May 23 18:13:16 2022 from [redacted]
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

razumovsky_r@mango-qa-vm:~$ |
```

Figure 1. SSH connected successfully.

Therefore, the SSH connection between WSL2 under Windows 10 is established so that we are able to configure our virtual machine as per our needs.

3. Install .NET SDK and Runtime to the Ubuntu 20.04

Next, we should install the .NET SDK (unexpectedly again) in order to run our application. Proceeding, we refer to the Microsoft documentation article named `Install the .NET SDK or the .NET Runtime on Ubuntu` [Cor22], precisely the version is 20.04. As per documentation, consider the following commands to install .NET 6.0 SDK to your Ubuntu VM



20.04

Installing with APT can be done with a few commands. Before you install .NET, run the following commands to add the Microsoft package signing key to your list of trusted keys and add the package repository.

Open a terminal and run the following commands:

```
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packages-micros
sudo dpkg -i packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
```

Install the SDK

The .NET SDK allows you to develop apps with .NET. If you install the .NET SDK, you don't need to install the corresponding runtime. To install the .NET SDK, run the following commands:

```
sudo apt-get update; \
  sudo apt-get install -y apt-transport-https && \
  sudo apt-get update && \
  sudo apt-get install -y dotnet-sdk-6.0
```

Important

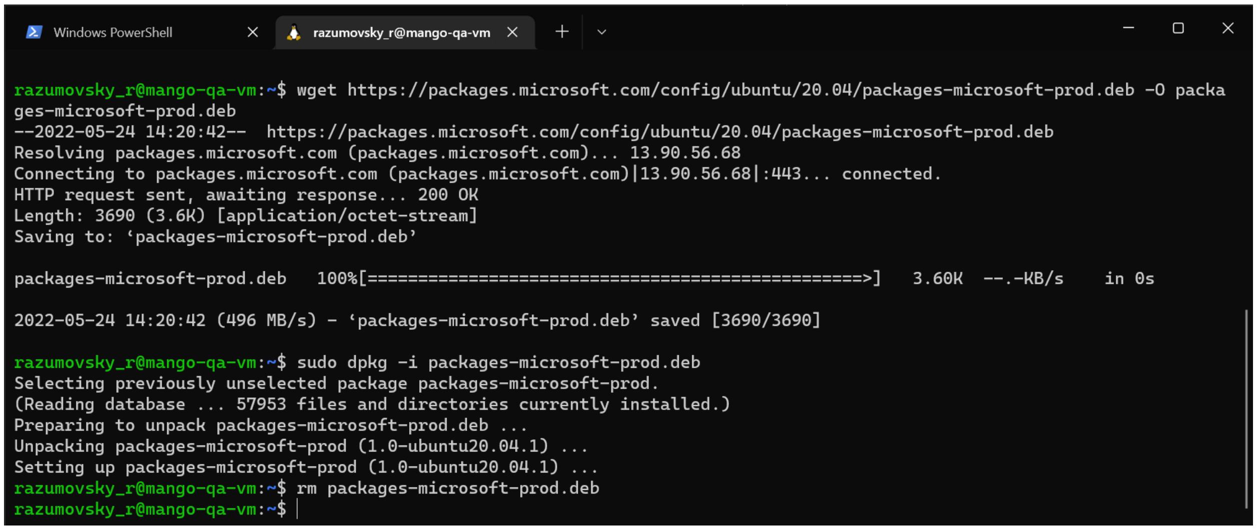
If you receive an error message similar to **Unable to locate package dotnet-sdk-6.0**, see the APT troubleshooting section.

Figure 2. Ubuntu 20.04 install .NET 6.0 SDK MSDN.

Prepare your virtual machine applying the commands

- `wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.de`
  `-O packages-microsoft-prod.deb`
- `sudo dpkg -i packages-microsoft-prod.deb`
- `rm packages-microsoft-prod.deb`

The terminal output is as follows



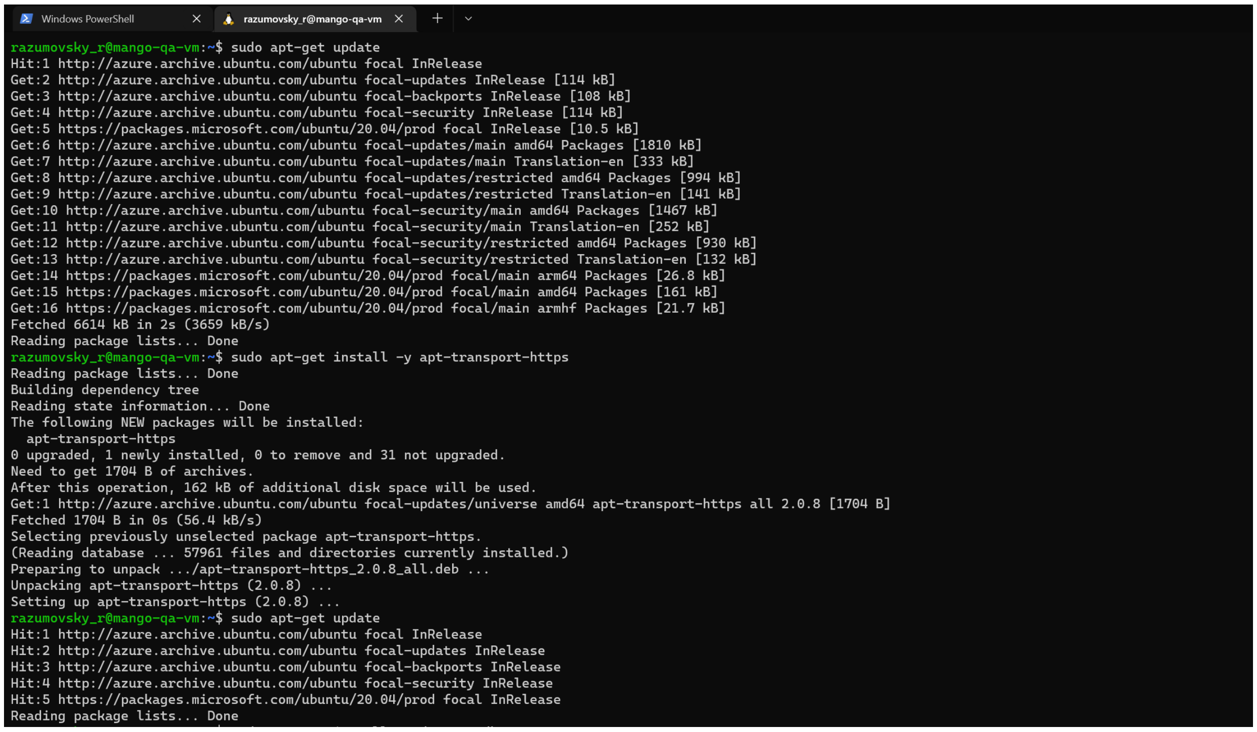
```
razumovsky_r@mango-qa-vm:~$ wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb -O packa
ges-microsoft-prod.deb
--2022-05-24 14:20:42--  https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb
Resolving packages.microsoft.com (packages.microsoft.com)... 13.90.56.68
Connecting to packages.microsoft.com (packages.microsoft.com)|13.90.56.68|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3690 (3.6K) [application/octet-stream]
Saving to: ‘packages-microsoft-prod.deb’

packages-microsoft-prod.deb   100%[=============================================>]   3.60K  --.-KB/s    in 0s

2022-05-24 14:20:42 (496 MB/s) - ‘packages-microsoft-prod.deb’ saved [3690/3690]

razumovsky_r@mango-qa-vm:~$ sudo dpkg -i packages-microsoft-prod.deb
Selecting previously unselected package packages-microsoft-prod.
(Reading database ... 57953 files and directories currently installed.)
Preparing to unpack packages-microsoft-prod.deb ...
Unpacking packages-microsoft-prod (1.0-ubuntu20.04.1) ...
Setting up packages-microsoft-prod (1.0-ubuntu20.04.1) ...
razumovsky_r@mango-qa-vm:~$ rm packages-microsoft-prod.deb
razumovsky_r@mango-qa-vm:~$ 
```

Figure 3. Virtual machine preparation..

Apply the following commands in order to install the SDK

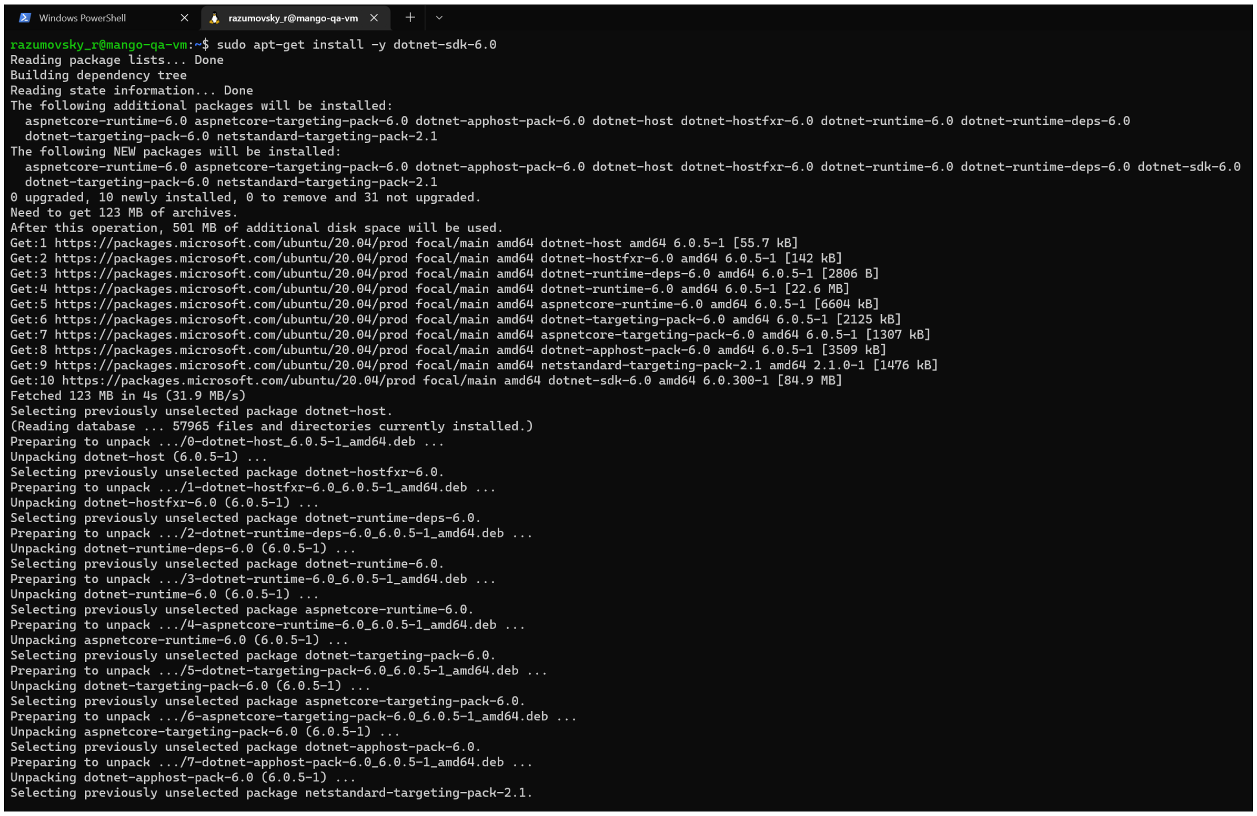
- `sudo apt-get update`
- `sudo apt-get install -y apt-transport-https`
- `sudo apt-get update`
- `sudo apt-get install -y dotnet-sdk-6.0`

The terminal output after .NET 6.0 SDK installation is as follows



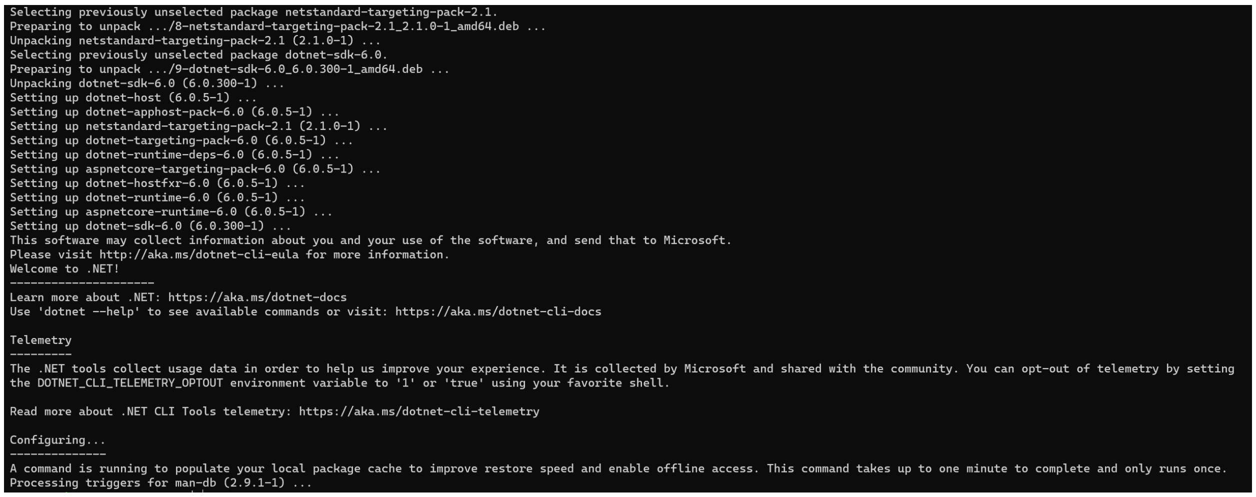
```
razumovsky_r@mango-qa-vm:~$ sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:5 https://packages.microsoft.com/ubuntu/20.04/prod focal InRelease [10.5 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1810 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu focal-updates/main Translation-en [333 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu focal-updates/restricted amd64 Packages [994 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu focal-updates/restricted Translation-en [141 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu focal-security/main amd64 Packages [1467 kB]
Get:11 http://azure.archive.ubuntu.com/ubuntu focal-security/main Translation-en [252 kB]
Get:12 http://azure.archive.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [930 kB]
Get:13 http://azure.archive.ubuntu.com/ubuntu focal-security/restricted Translation-en [132 kB]
Get:14 https://packages.microsoft.com/ubuntu/20.04/prod focal/main arm64 Packages [26.8 kB]
Get:15 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 Packages [161 kB]
Get:16 https://packages.microsoft.com/ubuntu/20.04/prod focal/main armhf Packages [21.7 kB]
Fetched 6614 kB in 2s (3659 kB/s)
Reading package lists... Done
razumovsky_r@mango-qa-vm:~$ sudo apt-get install -y apt-transport-https
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  apt-transport-https
0 upgraded, 1 newly installed, 0 to remove and 31 not upgraded.
Need to get 1704 B of archives.
After this operation, 162 kB of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe amd64 apt-transport-https all 2.0.8 [1704 B]
Fetched 1704 B in 0s (56.4 kB/s)
Selecting previously unselected package apt-transport-https.
(Reading database ... 57961 files and directories currently installed.)
Preparing to unpack .../apt-transport-https_2.0.8_all.deb ...
Unpacking apt-transport-https (2.0.8) ...
Setting up apt-transport-https (2.0.8) ...
razumovsky_r@mango-qa-vm:~$ sudo apt-get update
Hit:1 http://azure.archive.ubuntu.com/ubuntu focal InRelease
Hit:2 http://azure.archive.ubuntu.com/ubuntu focal-updates InRelease
Hit:3 http://azure.archive.ubuntu.com/ubuntu focal-backports InRelease
Hit:4 http://azure.archive.ubuntu.com/ubuntu focal-security InRelease
Hit:5 https://packages.microsoft.com/ubuntu/20.04/prod focal InRelease
Reading package lists... Done
```

Figure 4. Ubuntu 20.04 install .NET 6.0 SDK terminal output.



```
razumovsky_r@mango-qa-vm:~$ sudo apt-get install -y dotnet-sdk-6.0
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  aspnetcore-runtime-6.0 aspnetcore-targeting-pack-6.0 dotnet-apphost-pack-6.0 dotnet-host dotnet-hostfxr-6.0 dotnet-runtime-6.0 dotnet-runtime-deps-6.0
  dotnet-targeting-pack-6.0 netstandard-targeting-pack-2.1
The following NEW packages will be installed:
  aspnetcore-runtime-6.0 aspnetcore-targeting-pack-6.0 dotnet-apphost-pack-6.0 dotnet-host dotnet-hostfxr-6.0 dotnet-runtime-6.0 dotnet-runtime-deps-6.0 dotnet-sdk-6.0
  dotnet-targeting-pack-6.0 netstandard-targeting-pack-2.1
0 upgraded, 10 newly installed, 0 to remove and 31 not upgraded.
Need to get 123 MB of archives.
After this operation, 501 MB of additional disk space will be used.
Get:1 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 dotnet-host amd64 6.0.5-1 [55.7 kB]
Get:2 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 dotnet-hostfxr-6.0 amd64 6.0.5-1 [142 kB]
Get:3 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 dotnet-runtime-deps-6.0 amd64 6.0.5-1 [2806 B]
Get:4 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 dotnet-runtime-6.0 amd64 6.0.5-1 [22.6 MB]
Get:5 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 aspnetcore-runtime-6.0 amd64 6.0.5-1 [6604 kB]
Get:6 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 dotnet-targeting-pack-6.0 amd64 6.0.5-1 [2125 kB]
Get:7 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 aspnetcore-targeting-pack-6.0 amd64 6.0.5-1 [1307 kB]
Get:8 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 dotnet-apphost-pack-6.0 amd64 6.0.5-1 [3509 kB]
Get:9 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 netstandard-targeting-pack-2.1 amd64 2.1.0-1 [1476 kB]
Get:10 https://packages.microsoft.com/ubuntu/20.04/prod focal/main amd64 dotnet-sdk-6.0 amd64 6.0.300-1 [84.9 MB]
Fetched 123 MB in 4s (31.9 MB/s)
Selecting previously unselected package dotnet-host.
(Reading database ... 57965 files and directories currently installed.)
Preparing to unpack .../0-dotnet-host_6.0.5-1_amd64.deb ...
Unpacking dotnet-host (6.0.5-1) ...
Selecting previously unselected package dotnet-hostfxr-6.0.
Preparing to unpack .../1-dotnet-hostfxr-6.0_6.0.5-1_amd64.deb ...
Unpacking dotnet-hostfxr-6.0 (6.0.5-1) ...
Selecting previously unselected package dotnet-runtime-deps-6.0.
Preparing to unpack .../2-dotnet-runtime-deps-6.0_6.0.5-1_amd64.deb ...
Unpacking dotnet-runtime-deps-6.0 (6.0.5-1) ...
Selecting previously unselected package dotnet-runtime-6.0.
Preparing to unpack .../3-dotnet-runtime-6.0_6.0.5-1_amd64.deb ...
Unpacking dotnet-runtime-6.0 (6.0.5-1) ...
Selecting previously unselected package aspnetcore-runtime-6.0.
Preparing to unpack .../4-aspnetcore-runtime-6.0_6.0.5-1_amd64.deb ...
Unpacking aspnetcore-runtime-6.0 (6.0.5-1) ...
Selecting previously unselected package dotnet-targeting-pack-6.0.
Preparing to unpack .../5-dotnet-targeting-pack-6.0_6.0.5-1_amd64.deb ...
Unpacking dotnet-targeting-pack-6.0 (6.0.5-1) ...
Selecting previously unselected package aspnetcore-targeting-pack-6.0.
Preparing to unpack .../6-aspnetcore-targeting-pack-6.0_6.0.5-1_amd64.deb ...
Unpacking aspnetcore-targeting-pack-6.0 (6.0.5-1) ...
Selecting previously unselected package dotnet-apphost-pack-6.0.
Preparing to unpack .../7-dotnet-apphost-pack-6.0_6.0.5-1_amd64.deb ...
Unpacking dotnet-apphost-pack-6.0 (6.0.5-1) ...
Selecting previously unselected package netstandard-targeting-pack-2.1.
```

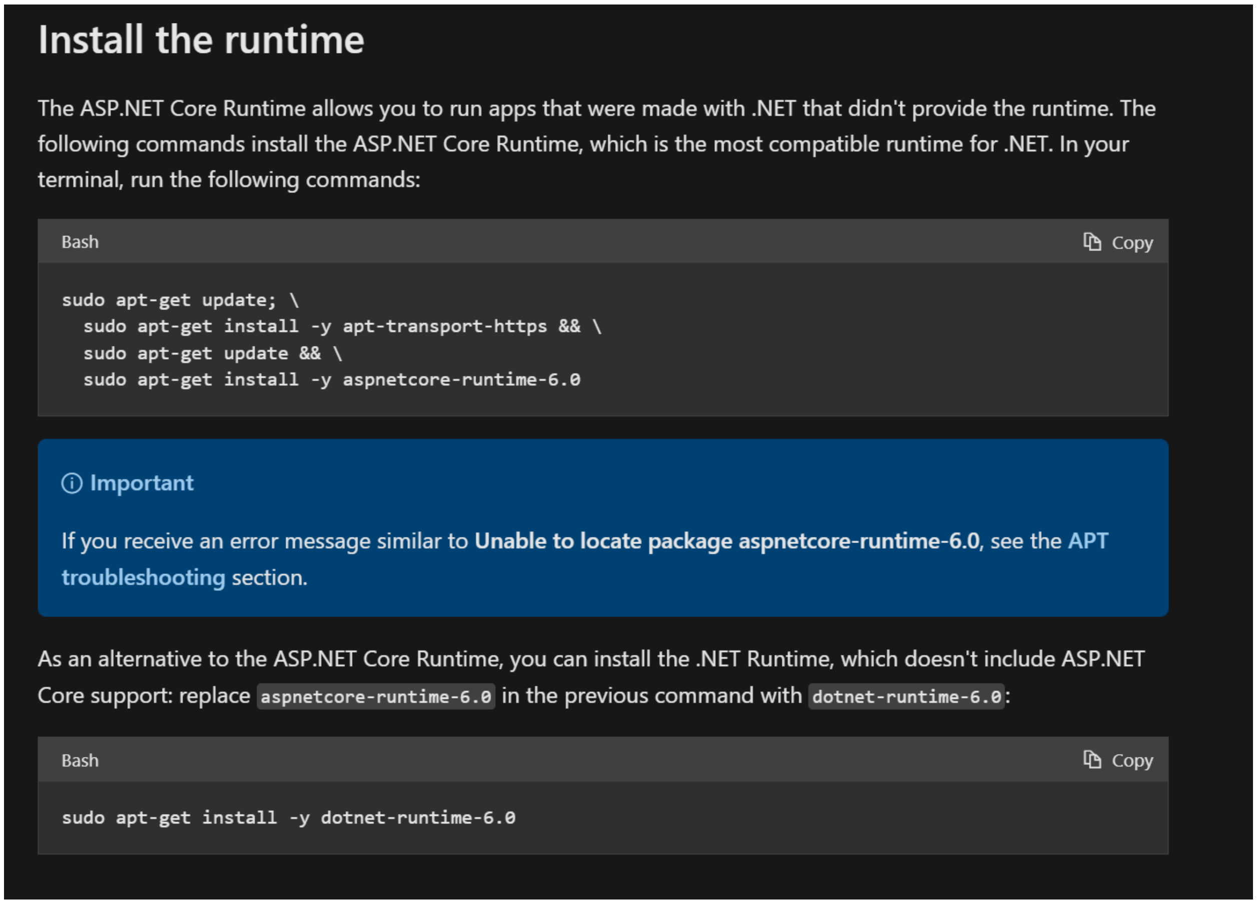
Figure 5. Ubuntu 20.04 install .NET 6.0 SDK terminal output.



```
Selecting previously unselected package netstandard-targeting-pack-2.1.
Preparing to unpack .../8-netstandard-targeting-pack-2.1_2.1.0-1_amd64.deb ...
Unpacking netstandard-targeting-pack-2.1 (2.1.0-1) ...
Selecting previously unselected package dotnet-sdk-6.0.
Preparing to unpack .../9-dotnet-sdk-6.0_6.0.300-1_amd64.deb ...
Unpacking dotnet-sdk-6.0 (6.0.300-1) ...
Setting up dotnet-host (6.0.5-1) ...
Setting up dotnet-apphost-pack-6.0 (6.0.5-1) ...
Setting up netstandard-targeting-pack-2.1 (2.1.0-1) ...
Setting up dotnet-targeting-pack-6.0 (6.0.5-1) ...
Setting up dotnet-runtime-deps-6.0 (6.0.5-1) ...
Setting up aspnetcore-targeting-pack-6.0 (6.0.5-1) ...
Setting up dotnet-hostfxr-6.0 (6.0.5-1) ...
Setting up dotnet-runtime-6.0 (6.0.5-1) ...
Setting up aspnetcore-runtime-6.0 (6.0.5-1) ...
Setting up dotnet-sdk-6.0 (6.0.300-1) ...
This software may collect information about you and your use of the software, and send that to Microsoft.
Please visit http://aka.ms/dotnet-cli-eula for more information.
Welcome to .NET!
---------------------
Learn more about .NET: https://aka.ms/dotnet-docs
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli-docs

Telemetry
---------
The .NET tools collect usage data in order to help us improve your experience. It is collected by Microsoft and shared with the community. You can opt-out of telemetry by setting
the DOTNET_CLI_TELEMETRY_OPTOUT environment variable to '1' or 'true' using your favorite shell.

Read more about .NET CLI Tools telemetry: https://aka.ms/dotnet-cli-telemetry

Configuring...
--------------
A command is running to populate your local package cache to improve restore speed and enable offline access. This command takes up to one minute to complete and only runs once.
Processing triggers for man-db (2.9.1-1) ...
```

Figure 6. Ubuntu 20.04 install .NET 6.0 SDK terminal output.

In order to install the .NET Runtime we refer again to the Microsoft documentation, that is



Install the runtime

The ASP.NET Core Runtime allows you to run apps that were made with .NET that didn't provide the runtime. The following commands install the ASP.NET Core Runtime, which is the most compatible runtime for .NET. In your terminal, run the following commands:

```bash
sudo apt-get update; \
  sudo apt-get install -y apt-transport-https && \
  sudo apt-get update && \
  sudo apt-get install -y aspnetcore-runtime-6.0
```

Important

If you receive an error message similar to **Unable to locate package aspnetcore-runtime-6.0**, see the APT troubleshooting section.

As an alternative to the ASP.NET Core Runtime, you can install the .NET Runtime, which doesn't include ASP.NET Core support: replace `aspnetcore-runtime-6.0` in the previous command with `dotnet-runtime-6.0`:

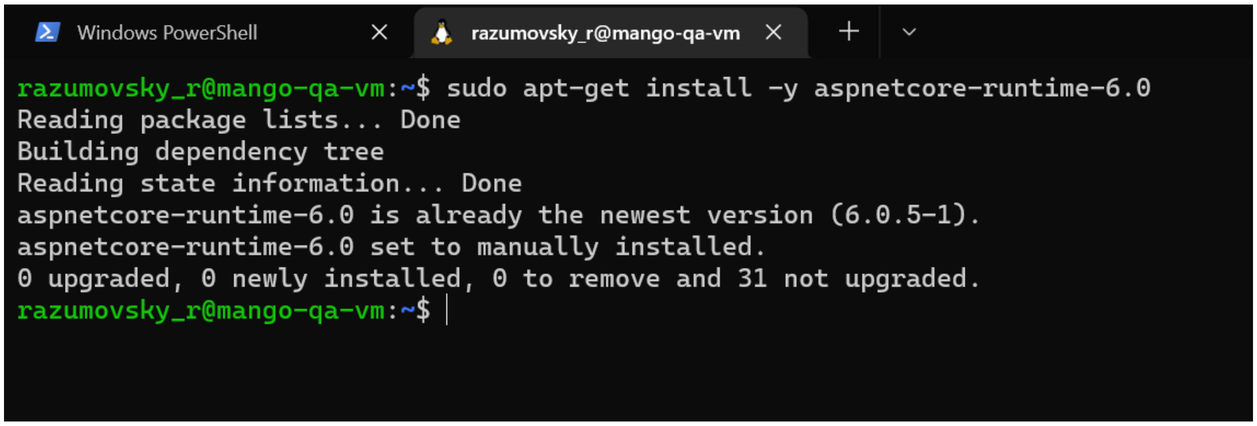
```bash
sudo apt-get install -y dotnet-runtime-6.0
```

Figure 7. Install the .NET SDK or the .NET Runtime on Ubuntu MSDN.

We install .NET runtime using the commands

- `sudo apt-get update`
- `sudo apt-get install -y apt-transport-https`
- `sudo apt-get update`
- `sudo apt-get install -y aspnetcore-runtime-6.0`

Terminal output as follows



```
razumovsky_r@mango-qa-vm:~$ sudo apt-get install -y aspnetcore-runtime-6.0
Reading package lists... Done
Building dependency tree
Reading state information... Done
aspnetcore-runtime-6.0 is already the newest version (6.0.5-1).
aspnetcore-runtime-6.0 set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
razumovsky_r@mango-qa-vm:~$
```

Figure 8. Ubuntu 20.04 install .NET 6.0 Runtime terminal output.

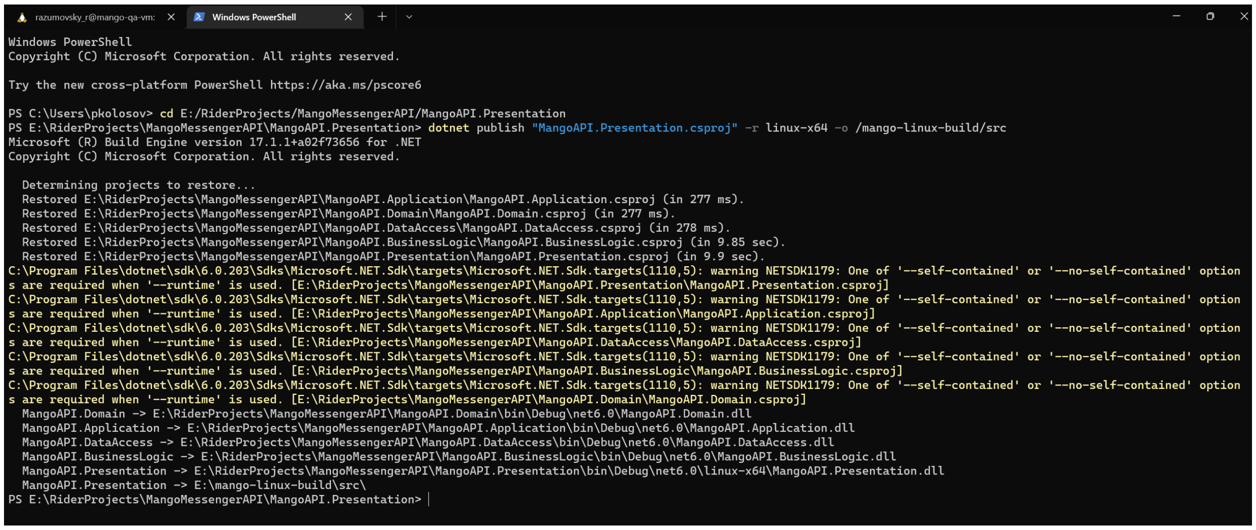
Therefore, the .NET SDK and Runtime are installed so that we are able to run specified .NET app on behalf of our Ubuntu virtual machine.

4. Copy build files to the VM via SSH

Now we have to build our .NET Core Web Application to the specified folder, say `/mango-linux-build/src`. Note that it is much better to build it on behalf of Windows 10 main machine, not WSL 2.0 one. We use the following commands to build .NET Core Web App with `Release` configuration

- `cd E:/RiderProjects/MangoMessengerAPI/MangoAPI.Presentation`
- `dotnet publish "MangoAPI.Presentation.csproj" -r linux-x64 -o /mango-linux-build/src`

Terminal output is as follows



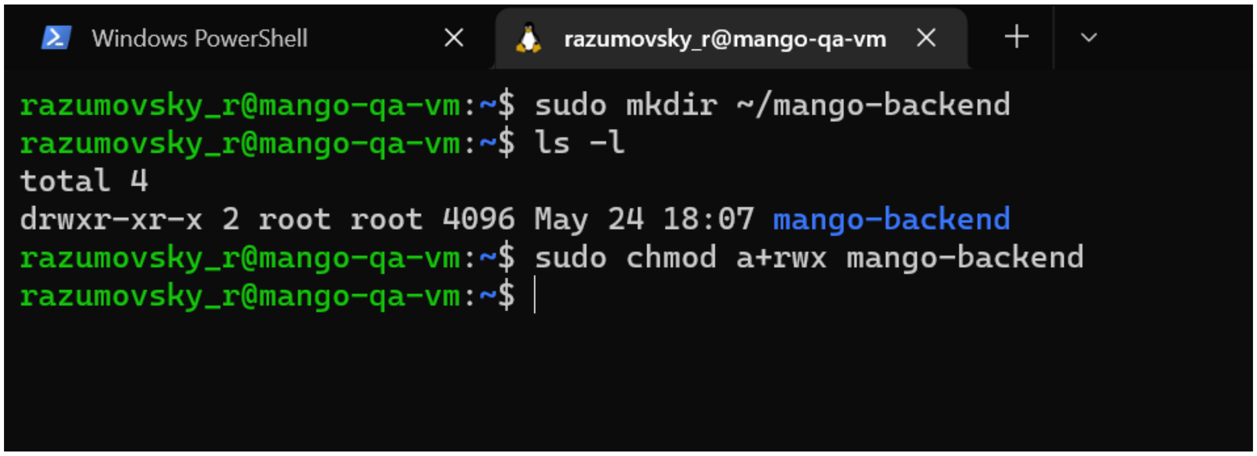
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\pkolosov> cd E:/RiderProjects/MangoMessengerAPI/MangoAPI.Presentation
PS E:\RiderProjects\MangoMessengerAPI\MangoAPI.Presentation> dotnet publish "MangoAPI.Presentation.csproj" -r linux-x64 -o /mango-linux-build/src
Microsoft (R) Build Engine version 17.1.1+a02f73656 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  Restored E:\RiderProjects\MangoMessengerAPI\MangoAPI.Application\MangoAPI.Application.csproj (in 277 ms).
  Restored E:\RiderProjects\MangoMessengerAPI\MangoAPI.Domain\MangoAPI.Domain.csproj (in 277 ms).
  Restored E:\RiderProjects\MangoMessengerAPI\MangoAPI.DataAccess\MangoAPI.DataAccess.csproj (in 278 ms).
  Restored E:\RiderProjects\MangoMessengerAPI\MangoAPI.BusinessLogic\MangoAPI.BusinessLogic.csproj (in 9.85 sec).
  Restored E:\RiderProjects\MangoMessengerAPI\MangoAPI.Presentation\MangoAPI.Presentation.csproj (in 9.9 sec).
C:\Program Files\dotnet\sdk\6.0.203\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Sdk.targets(1110,5): warning NETSDK1179: One of '--self-contained' or '--no-self-contained' option
s are required when '--runtime' is used. [E:\RiderProjects\MangoMessengerAPI\MangoAPI.Presentation\MangoAPI.Presentation.csproj]
C:\Program Files\dotnet\sdk\6.0.203\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Sdk.targets(1110,5): warning NETSDK1179: One of '--self-contained' or '--no-self-contained' option
s are required when '--runtime' is used. [E:\RiderProjects\MangoMessengerAPI\MangoAPI.Application\MangoAPI.Application.csproj]
C:\Program Files\dotnet\sdk\6.0.203\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Sdk.targets(1110,5): warning NETSDK1179: One of '--self-contained' or '--no-self-contained' option
s are required when '--runtime' is used. [E:\RiderProjects\MangoMessengerAPI\MangoAPI.DataAccess\MangoAPI.DataAccess.csproj]
C:\Program Files\dotnet\sdk\6.0.203\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Sdk.targets(1110,5): warning NETSDK1179: One of '--self-contained' or '--no-self-contained' option
s are required when '--runtime' is used. [E:\RiderProjects\MangoMessengerAPI\MangoAPI.BusinessLogic\MangoAPI.BusinessLogic.csproj]
C:\Program Files\dotnet\sdk\6.0.203\Sdks\Microsoft.NET.Sdk\targets\Microsoft.NET.Sdk.targets(1110,5): warning NETSDK1179: One of '--self-contained' or '--no-self-contained' option
s are required when '--runtime' is used. [E:\RiderProjects\MangoMessengerAPI\MangoAPI.Domain\MangoAPI.Domain.csproj]
  MangoAPI.Domain -> E:\RiderProjects\MangoMessengerAPI\MangoAPI.Domain\bin\Debug\net6.0\MangoAPI.Domain.dll
  MangoAPI.Application -> E:\RiderProjects\MangoMessengerAPI\MangoAPI.Application\bin\Debug\net6.0\MangoAPI.Application.dll
  MangoAPI.DataAccess -> E:\RiderProjects\MangoMessengerAPI\MangoAPI.DataAccess\bin\Debug\net6.0\MangoAPI.DataAccess.dll
  MangoAPI.BusinessLogic -> E:\RiderProjects\MangoMessengerAPI\MangoAPI.BusinessLogic\bin\Debug\net6.0\MangoAPI.BusinessLogic.dll
  MangoAPI.Presentation -> E:\RiderProjects\MangoMessengerAPI\MangoAPI.Presentation\bin\Debug\net6.0\linux-x64\MangoAPI.Presentation.dll
  MangoAPI.Presentation -> E:\mango-linux-build\src\
PS E:\RiderProjects\MangoMessengerAPI\MangoAPI.Presentation>
```

Figure 9. Publish .NET Web app terminal output.

Let's create the folder `mango-backend` where build files to be stored. Do not forget to connect to your Azure VM via SSH. Do not also forget to assign read-write privileges to the folder, using the commands

- `sudo mkdir /mango-backend`
- `sudo chmod a+rwx /mango-backend`

Terminal output:



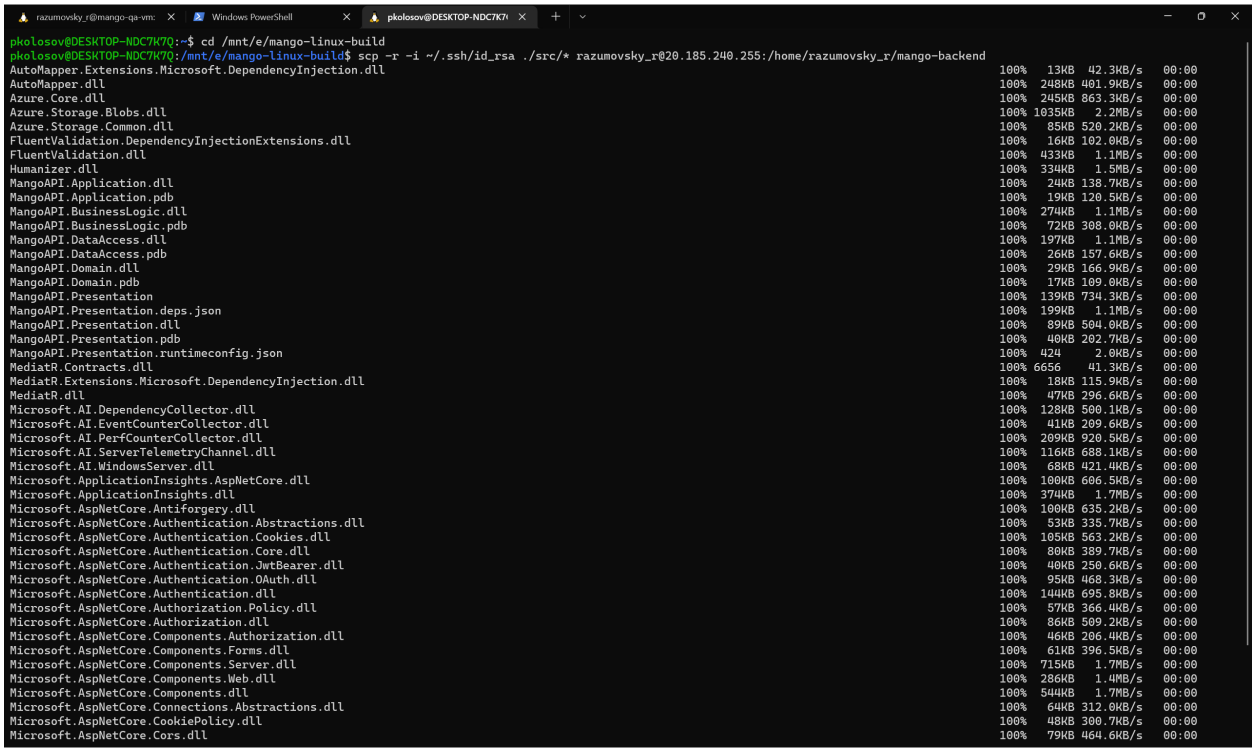
```
razumovsky_r@mango-qa-vm:~$ sudo mkdir ~/mango-backend
razumovsky_r@mango-qa-vm:~$ ls -l
total 4
drwxr-xr-x 2 root root 4096 May 24 18:07 mango-backend
razumovsky_r@mango-qa-vm:~$ sudo chmod a+rwx mango-backend
razumovsky_r@mango-qa-vm:~$
```

Figure 10. Create folder at remote VM.

As next step consider to copy build files to the remote folder on your Azure VM so that we execute our program after. We copy the build files on behalf of WSL2 this time. In order to copy the build files we use following commands

- `cd /mnt/e/mango-linux-build`
- `scp -r -i /.ssh/id_rsa ./src/* razumovsky_r@VM_IP_ADDRESS:/home/razumovsky_r/mango-backend`

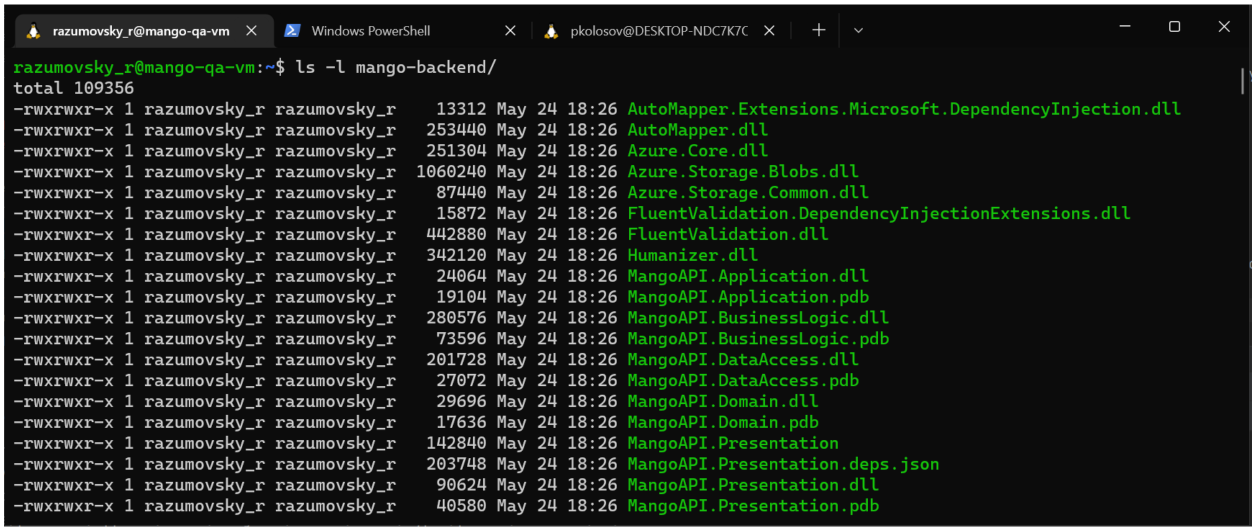
where `id_rsa` is the private key. Terminal output:



```
pkolosov@DESKTOP-NDC7K7Q:~$ cd /mnt/e/mango-linux-build
pkolosov@DESKTOP-NDC7K7Q:/mnt/e/mango-linux-build$ scp -r -i ~/.ssh/id_rsa ./src/* razumovsky_r@20.185.240.255:/home/razumovsky_r/mango-backend
AutoMapper.Extensions.Microsoft.DependencyInjection.dll        100%   13KB  42.3KB/s   00:00
AutoMapper.dll                                                 100%  248KB 401.9KB/s   00:00
Azure.Core.dll                                                 100%  245KB 863.3KB/s   00:00
Azure.Storage.Blobs.dll                                        100% 1035KB   2.2MB/s   00:00
Azure.Storage.Common.dll                                       100%   85KB 520.2KB/s   00:00
FluentValidation.DependencyInjectionExtensions.dll             100%   16KB 102.0KB/s   00:00
FluentValidation.dll                                           100%  433KB   1.1MB/s   00:00
Humanizer.dll                                                  100%  334KB   1.5MB/s   00:00
MangoAPI.Application.dll                                       100%   24KB 138.7KB/s   00:00
MangoAPI.Application.pdb                                       100%   19KB 120.5KB/s   00:00
MangoAPI.BusinessLogic.dll                                     100%  274KB   1.1MB/s   00:00
MangoAPI.BusinessLogic.pdb                                     100%   72KB 308.0KB/s   00:00
MangoAPI.DataAccess.dll                                        100%  197KB   1.1MB/s   00:00
MangoAPI.DataAccess.pdb                                        100%   26KB 157.6KB/s   00:00
MangoAPI.Domain.dll                                            100%   29KB 166.9KB/s   00:00
MangoAPI.Domain.pdb                                            100%   17KB 109.0KB/s   00:00
MangoAPI.Presentation                                          100%  139KB 734.3KB/s   00:00
MangoAPI.Presentation.deps.json                                100%  199KB   1.1MB/s   00:00
MangoAPI.Presentation.dll                                      100%   89KB 504.0KB/s   00:00
MangoAPI.Presentation.pdb                                      100%   40KB 202.7KB/s   00:00
MangoAPI.Presentation.runtimeconfig.json                       100%  424      2.0KB/s   00:00
MediatR.Contracts.dll                                          100% 6656     41.3KB/s   00:00
MediatR.Extensions.Microsoft.DependencyInjection.dll           100%   18KB 115.9KB/s   00:00
MediatR.dll                                                    100%   47KB 296.6KB/s   00:00
Microsoft.AI.DependencyCollector.dll                           100%  128KB 500.1KB/s   00:00
Microsoft.AI.EventCounterCollector.dll                         100%   41KB 209.6KB/s   00:00
Microsoft.AI.PerfCounterCollector.dll                          100%  209KB 920.5KB/s   00:00
Microsoft.AI.ServerTelemetryChannel.dll                        100%  116KB 688.1KB/s   00:00
Microsoft.AI.WindowsServer.dll                                 100%   68KB 421.4KB/s   00:00
Microsoft.ApplicationInsights.AspNetCore.dll                   100%  100KB 606.5KB/s   00:00
Microsoft.ApplicationInsights.dll                              100%  374KB   1.7MB/s   00:00
Microsoft.AspNetCore.Antiforgery.dll                           100%  100KB 635.2KB/s   00:00
Microsoft.AspNetCore.Authentication.Abstractions.dll           100%   53KB 335.7KB/s   00:00
Microsoft.AspNetCore.Authentication.Cookies.dll                100%  105KB 563.2KB/s   00:00
Microsoft.AspNetCore.Authentication.Core.dll                   100%   80KB 389.7KB/s   00:00
Microsoft.AspNetCore.Authentication.JwtBearer.dll              100%   40KB 250.6KB/s   00:00
Microsoft.AspNetCore.Authentication.OAuth.dll                  100%   95KB 468.3KB/s   00:00
Microsoft.AspNetCore.Authentication.dll                        100%  144KB 695.8KB/s   00:00
Microsoft.AspNetCore.Authorization.Policy.dll                  100%   57KB 366.4KB/s   00:00
Microsoft.AspNetCore.Authorization.dll                         100%   86KB 509.2KB/s   00:00
Microsoft.AspNetCore.Components.Authorization.dll              100%   46KB 206.4KB/s   00:00
Microsoft.AspNetCore.Components.Forms.dll                      100%   61KB 396.5KB/s   00:00
Microsoft.AspNetCore.Components.Server.dll                     100%  715KB   1.7MB/s   00:00
Microsoft.AspNetCore.Components.Web.dll                        100%  286KB   1.4MB/s   00:00
Microsoft.AspNetCore.Components.dll                            100%  544KB   1.7MB/s   00:00
Microsoft.AspNetCore.Connections.Abstractions.dll              100%   64KB 312.0KB/s   00:00
Microsoft.AspNetCore.CookiePolicy.dll                          100%   48KB 300.7KB/s   00:00
Microsoft.AspNetCore.Cors.dll                                  100%   79KB 464.6KB/s   00:00
```

Figure 11. Copy build files via SSH.

Ensure build files are copied successfully to the remote VM, use the command `ls -l mango-backend`. Terminal output:



```
razumovsky_r@mango-qa-vm:~$ ls -l mango-backend/
total 109356
-rwxrwxr-x 1 razumovsky_r razumovsky_r   13312 May 24 18:26 AutoMapper.Extensions.Microsoft.DependencyInjection.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r  253440 May 24 18:26 AutoMapper.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r  251304 May 24 18:26 Azure.Core.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r 1060240 May 24 18:26 Azure.Storage.Blobs.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r   87440 May 24 18:26 Azure.Storage.Common.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r   15872 May 24 18:26 FluentValidation.DependencyInjectionExtensions.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r  442880 May 24 18:26 FluentValidation.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r  342120 May 24 18:26 Humanizer.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r   24064 May 24 18:26 MangoAPI.Application.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r   19104 May 24 18:26 MangoAPI.Application.pdb
-rwxrwxr-x 1 razumovsky_r razumovsky_r  280576 May 24 18:26 MangoAPI.BusinessLogic.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r   73596 May 24 18:26 MangoAPI.BusinessLogic.pdb
-rwxrwxr-x 1 razumovsky_r razumovsky_r  201728 May 24 18:26 MangoAPI.DataAccess.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r   27072 May 24 18:26 MangoAPI.DataAccess.pdb
-rwxrwxr-x 1 razumovsky_r razumovsky_r   29696 May 24 18:26 MangoAPI.Domain.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r   17636 May 24 18:26 MangoAPI.Domain.pdb
-rwxrwxr-x 1 razumovsky_r razumovsky_r  142840 May 24 18:26 MangoAPI.Presentation
-rwxrwxr-x 1 razumovsky_r razumovsky_r  203748 May 24 18:26 MangoAPI.Presentation.deps.json
-rwxrwxr-x 1 razumovsky_r razumovsky_r   90624 May 24 18:26 MangoAPI.Presentation.dll
-rwxrwxr-x 1 razumovsky_r razumovsky_r   40580 May 24 18:26 MangoAPI.Presentation.pdb
```

Figure 12. Check files at remote VM.

Therefore, the specified .NET Core web application is copied to the Ubuntu virtual machine so that it can be executed thanks to the previously installed .NET SDKs and runtimes.

5. Configure Ubuntu service

In this section the main aim is to implement an Ubuntu service such that runs our previously built .NET Core web application. It means that we have to configure the environment variables used in our application as well as to configure the firewall rules so that application will be able to communicate with another resources like databases, blobs etc. Ubuntu server refers to the entry point of the web app, that is

```
/home/razumovsky_r/mango-backend/MangoAPI.Presentation
```

Use the command to create service

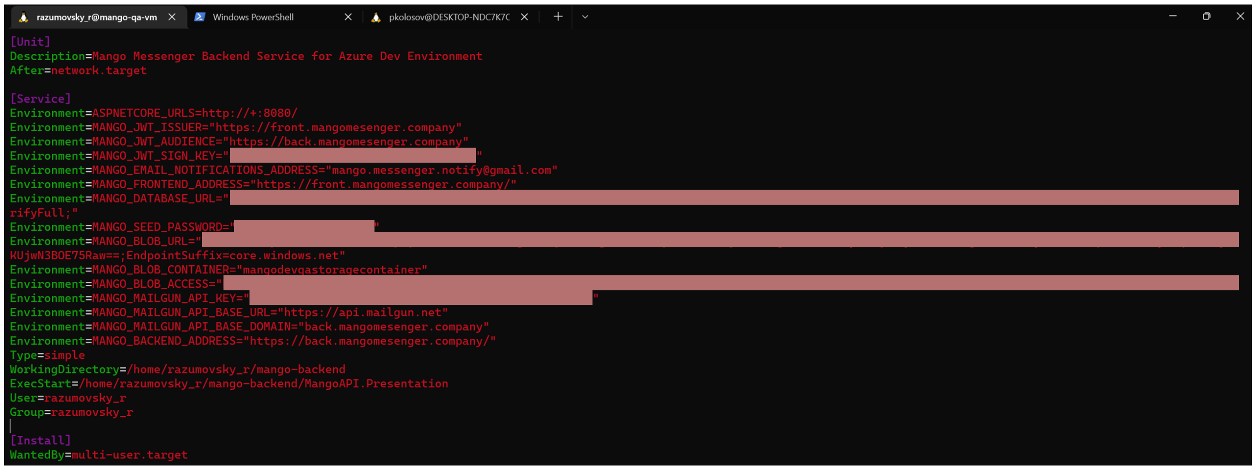
```
sudo vim /etc/systemd/system/mangoback.service
```

Paste the following text there

```
[Unit]
Description=Mango Messenger Backend Service for Azure Dev Environment
After=network.target

[Service]
Environment=ASPNETCORE_URLS=http://+:8080/
Environment=MANGO_JW_ISSUER="https://front.mangomessenger.company"
Environment=MANGO_JWT_AUDIENCE="https://back.mangomessenger.company"
Environment=MANGO_JWT_SIGN_KEY="d32d7cea-4cb8-4488-aa94-323ffb8cbdf4"
Environment=MANGO_EMAIL_NOTIFICATIONS_ADDRESS="mango@gmail.com"
Environment=MANGO_FRONTEND_ADDRESS="https://front.mangomessenger.company/"
Environment=MANGO_DATABASE_URL="database.connection.string"
Environment=MANGO_SEED_PASSWORD="seedPass"
Environment=MANGO_BLOB_URL="blob.url.connection.string"
Environment=MANGO_BLOB_CONTAINER="container.name"
Environment=MANGO_BLOB_ACCESS="blob.access.url"
Environment=MANGO_MAILGUN_API_KEY="mailgun.api.key"
Environment=MANGO_MAILGUN_API_BASE_URL="https://api.mailgun.net"
Environment=MANGO_MAILGUN_API_BASE_DOMAIN="back.mangomessenger.company"
Environment=MANGO_BACKEND_ADDRESS="https://back.mangomessenger.company/"
Type=simple
WorkingDirectory=/home/razumovsky_r/mango-backend
ExecStart=/home/razumovsky_r/mango-backend/MangoAPI.Presentation
User=razumovsky_r
Group=razumovsky_r
```

```
[Install]
WantedBy=multi-user.target
```

From the vim it should look as follows [Kol22b]



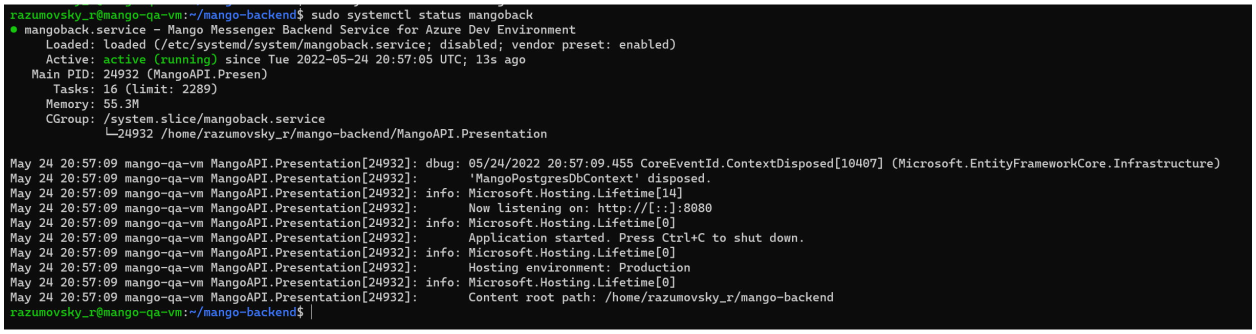
```
[Unit]
Description=Mango Messenger Backend Service for Azure Dev Environment
After=network.target

[Service]
Environment=ASPNETCORE_URLS=http://+:8080/
Environment=MANGO_JWT_ISSUER="https://front.mangomesenger.company"
Environment=MANGO_JWT_AUDIENCE="https://back.mangomesenger.company"
Environment=MANGO_JWT_SIGN_KEY="[redacted]"
Environment=MANGO_EMAIL_NOTIFICATIONS_ADDRESS="mango.messenger.notify@gmail.com"
Environment=MANGO_FRONTEND_ADDRESS="https://front.mangomessenger.company/"
Environment=MANGO_DATABASE_URL="[redacted]
rifyFull;"
Environment=MANGO_SEED_PASSWORD="[redacted]"
Environment=MANGO_BLOB_URL="[redacted]
KUjwN3BOE75Raw==;EndpointSuffix=core.windows.net"
Environment=MANGO_BLOB_CONTAINER="mangodevqastoragecontainer"
Environment=MANGO_BLOB_ACCESS="[redacted]
Environment=MANGO_MAILGUN_API_KEY="[redacted]"
Environment=MANGO_MAILGUN_API_BASE_URL="https://api.mailgun.net"
Environment=MANGO_MAILGUN_API_BASE_DOMAIN="back.mangomesenger.company"
Environment=MANGO_BACKEND_ADDRESS="https://back.mangomesenger.company/"
Type=simple
WorkingDirectory=/home/razumovsky_r/mango-backend
ExecStart=/home/razumovsky_r/mango-backend/MangoAPI.Presentation
User=razumovsky_r
Group=razumovsky_r

[Install]
WantedBy=multi-user.target
```

Figure 13. Ubuntu service opened in vim.

Make sure all resources are listening from the outside, check firewall rules on database side prior to run the service. Start and check health of the service using

- `sudo systemctl start mangoback`
- `sudo systemctl status mangoback`

Terminal output:



```
razumovsky_r@mango-qa-vm:~/mango-backend$ sudo systemctl status mangoback
● mangoback.service - Mango Messenger Backend Service for Azure Dev Environment
     Loaded: loaded (/etc/systemd/system/mangoback.service; disabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-05-24 20:57:05 UTC; 13s ago
   Main PID: 24932 (MangoAPI.Presen)
      Tasks: 16 (limit: 2289)
     Memory: 55.3M
     CGroup: /system.slice/mangoback.service
             └─24932 /home/razumovsky_r/mango-backend/MangoAPI.Presentation

May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]: dbug: 05/24/2022 20:57:09.455 CoreEventId.ContextDisposed[10407] (Microsoft.EntityFrameworkCore.Infrastructure)
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]:       'MangoPostgresDbContext' disposed.
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]: info: Microsoft.Hosting.Lifetime[14]
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]:       Now listening on: http://[::]:8080
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]: info: Microsoft.Hosting.Lifetime[0]
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]:       Application started. Press Ctrl+C to shut down.
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]: info: Microsoft.Hosting.Lifetime[0]
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]:       Hosting environment: Production
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]: info: Microsoft.Hosting.Lifetime[0]
May 24 20:57:09 mango-qa-vm MangoAPI.Presentation[24932]:       Content root path: /home/razumovsky_r/mango-backend
razumovsky_r@mango-qa-vm:~/mango-backend$
```

Figure 14. Run ubuntu service and check status, terminal output.

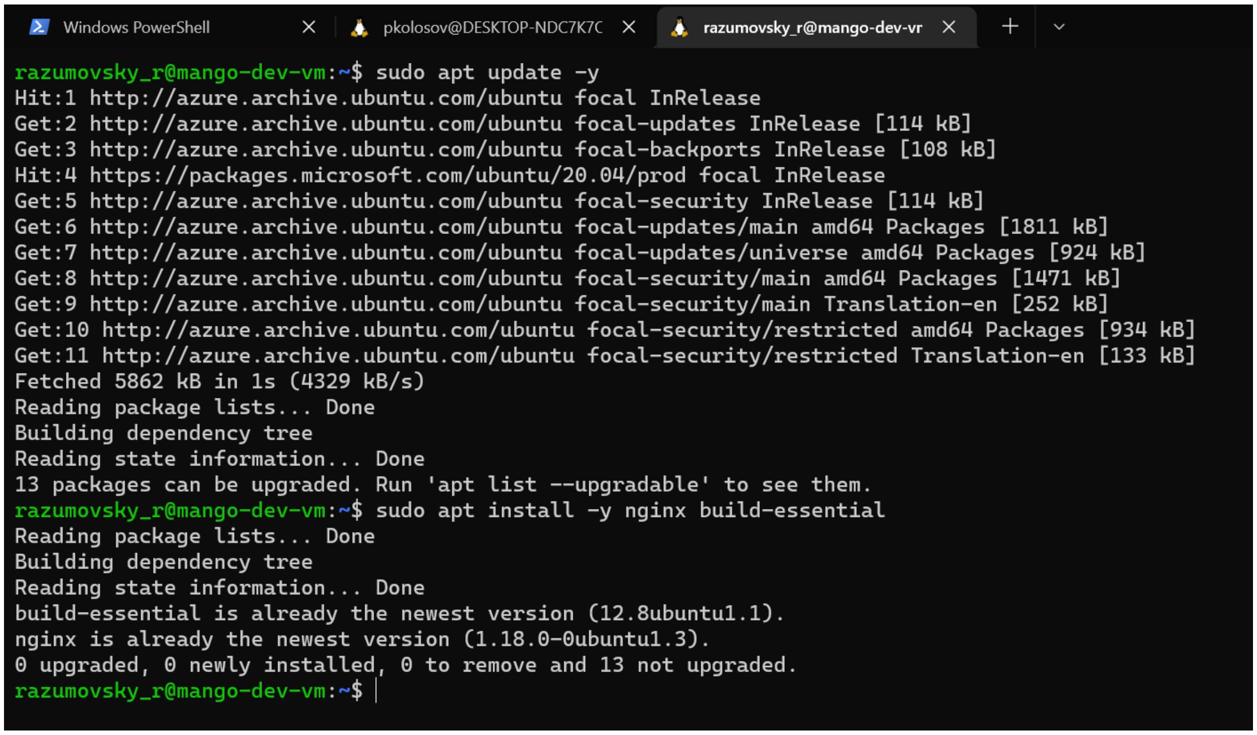
As a result of this section, we have created a specified ubuntu service that runs our previously copied .NET Core web application using installed .NET SDK and runtime.

6. Install and configure Nginx server

Now we have to configure the `nginx` server in order to expose our .NET Core web application to the outside. As a result of this section web app will be exposed and accessible via VM's external IP address. Let's install it using the commands

- `sudo apt update -y`
- `sudo apt install -y nginx build-essential`

Terminal output:


Windows PowerShell | pkolosov@DESKTOP-NDC7K7C | razumovsky_r@mango-dev-vr

```
razumovsky_r@mango-dev-vm:~$ sudo apt update -y
Hit:1 http://azure.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://azure.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu focal-backports InRelease [108 kB]
Hit:4 https://packages.microsoft.com/ubuntu/20.04/prod focal InRelease
Get:5 http://azure.archive.ubuntu.com/ubuntu focal-security InRelease [114 kB]
Get:6 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 Packages [1811 kB]
Get:7 http://azure.archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [924 kB]
Get:8 http://azure.archive.ubuntu.com/ubuntu focal-security/main amd64 Packages [1471 kB]
Get:9 http://azure.archive.ubuntu.com/ubuntu focal-security/main Translation-en [252 kB]
Get:10 http://azure.archive.ubuntu.com/ubuntu focal-security/restricted amd64 Packages [934 kB]
Get:11 http://azure.archive.ubuntu.com/ubuntu focal-security/restricted Translation-en [133 kB]
Fetched 5862 kB in 1s (4329 kB/s)
Reading package lists... Done
Building dependency tree
Reading state information... Done
13 packages can be upgraded. Run 'apt list --upgradable' to see them.
razumovsky_r@mango-dev-vm:~$ sudo apt install -y nginx build-essential
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.8ubuntu1.1).
nginx is already the newest version (1.18.0-0ubuntu1.3).
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
razumovsky_r@mango-dev-vm:~$
```

Figure 15. Ubuntu install nginx terminal output.

Next, it is necessary to create nginx configuration [Kol22a] that exposes our application, that is

```
server {
    server_name STATIC_IP_ADDRESS_OF_VM;

    location / {
        include proxy_params;
        proxy_pass http://127.0.0.1:8080;
    }

    location /swagger {
        include proxy_params;
        proxy_pass http://127.0.0.1:8080;
    }

    location /api {
        include proxy_params;
        proxy_pass http://127.0.0.1:8080;
    }

    location /notify {
        proxy_pass http://127.0.0.1:8080;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

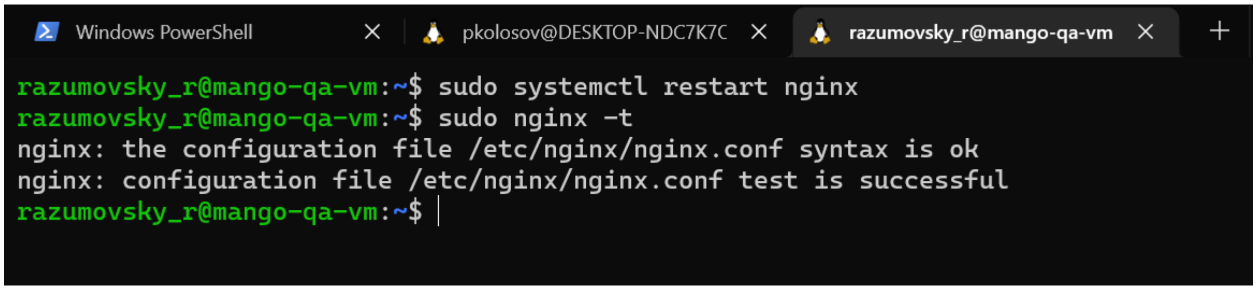
We create it at the following path on behalf of our Azure VM via SSH

```
sudo vim /etc/nginx/conf.d/back.mangomesenger.company.conf
```

Restart nginx and validate its state using the commands

- `sudo systemctl restart nginx`
- `sudo nginx -t`

Terminal output:



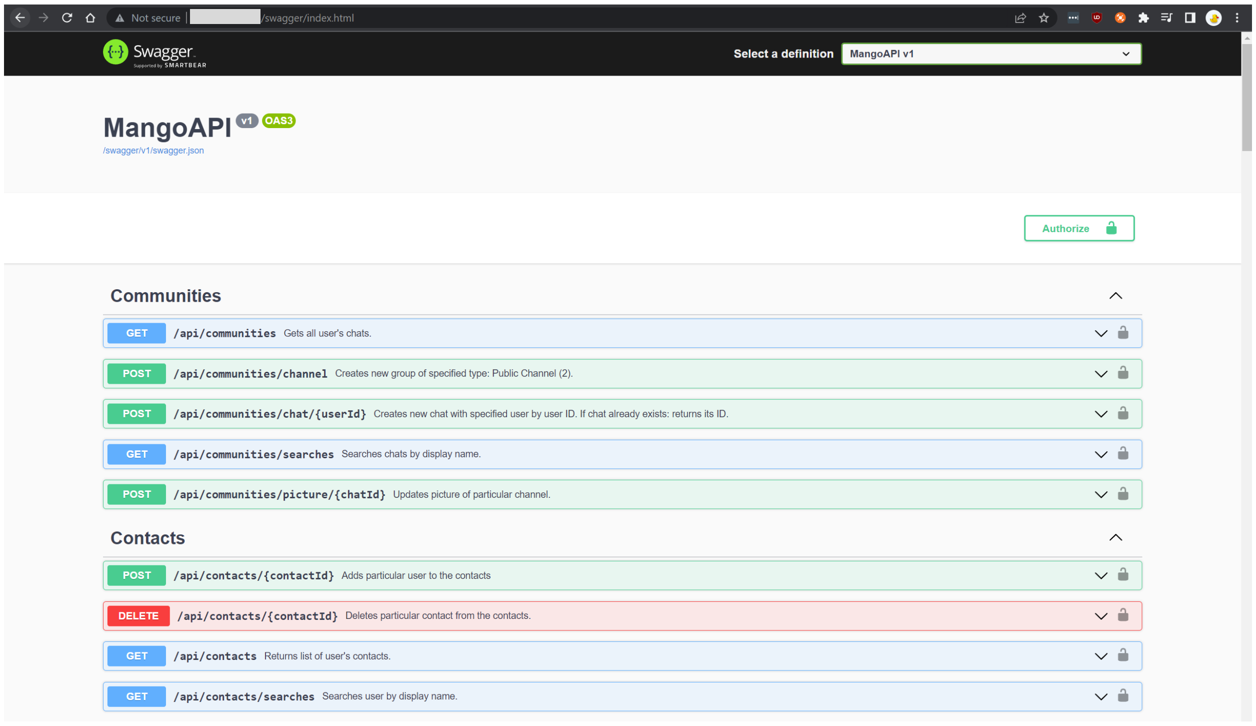
```
razumovsky_r@mango-qa-vm:~$ sudo systemctl restart nginx
razumovsky_r@mango-qa-vm:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
razumovsky_r@mango-qa-vm:~$
```

Figure 16. Restart and test nginx terminal output.

Now we must be able to find our application listening to the

```
http://STATIC_IP_ADDRESS_OF_THE_VM
```

And actually it works as expected


Not secure | [redacted]/swagger/index.html

Select a definition: MangoAPI v1

MangoAPI v1 OAS3

/swagger/v1/swagger.json

Communities

```
GET    /api/communities  Gets all user's chats.
POST   /api/communities/channel  Creates new group of specified type: Public Channel (2).
POST   /api/communities/chat/{userId}  Creates new chat with specified user by user ID. If chat already exists: returns its ID.
GET    /api/communities/searches  Searches chats by display name.
POST   /api/communities/picture/{chatId}  Updates picture of particular channel.
```

Contacts

```
POST   /api/contacts/{contactId}  Adds particular user to the contacts
DELETE /api/contacts/{contactId}  Deletes particular contact from the contacts.
GET    /api/contacts  Returns list of user's contacts.
GET    /api/contacts/searches  Searches user by display name.
```

Figure 17. .NET Core web app accessed via browser using static IP address of the virtual machine.

In this section we have installed and configured the `nginx` web server so that it exposes our .NET Core web application (run on behalf of Ubuntu service) from the previous section and makes it available from the web browser under the url `http://STATIC_IP_ADDRESS_OF_THE_VM`.

7. Configure domain name and SSL

In this section our main aim is to assign specified (previously bought) domain name to our .NET Core web application as well as to configure SSL certificate for it. What is domain name?

Domain name – is a string of text that maps to a numeric IP address, used to access a website from client software [Clo22]. The actual address of a website is a complex numerical IP address (e.g. 103.21.244.0), but thanks to DNS, users are able to enter human-friendly domain names and be routed to the websites they are looking for.

7.1. Buy and configure domain name using Cloudflare.

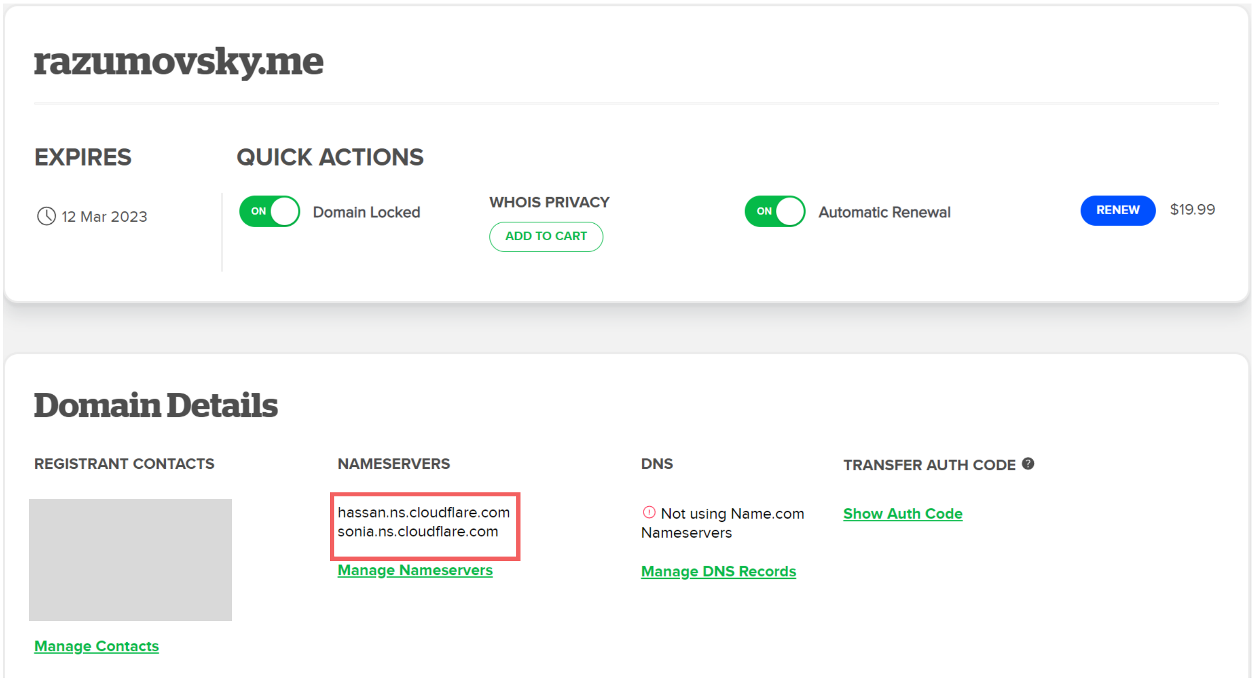
For instance, the domain name can be bought on the one of the following resources

- `https://www.name.com`
- `https://www.namecheap.com`
- `https://get.tech`

After that we have to associate our domain with the `cloudflare.com` service in order to manage out domain name and get some free DDoS protection and request analytics. For instance, I have bought a domain name withing `name.com` service and configured it using the following DNS records:

- `hassan.ns.cloudflare.com`
- `sonia.ns.cloudflare.com`

So it looks like as follows



razumovsky.me

EXPIRES

12 Mar 2023

QUICK ACTIONS

ON Domain Locked

WHOIS PRIVACY

ADD TO CART

ON Automatic Renewal

RENEW $19.99

Domain Details

REGISTRANT CONTACTS

Manage Contacts

NAMESERVERS

hassan.ns.cloudflare.com
sonia.ns.cloudflare.com

Manage Nameservers

DNS

Not using Name.com Nameservers

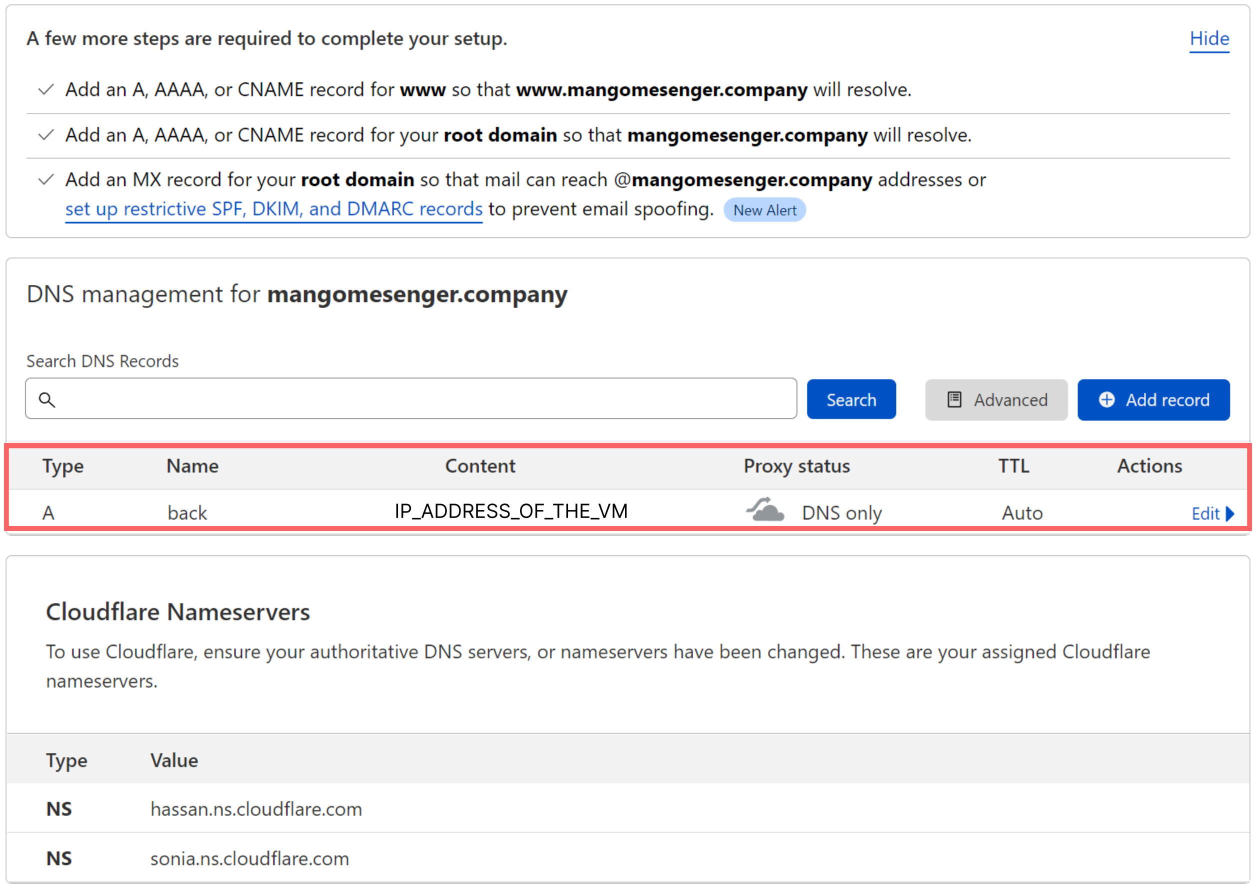
Manage DNS Records

TRANSFER AUTH CODE

Show Auth Code

Figure 18. Domain name configuration at `name.com`.

After that we have to configure our domain name at cloudflare providing an IP address of the virtual machine we host our .NET Core web application, that is


A few more steps are required to complete your setup.

Hide

✓ Add an A, AAAA, or CNAME record for **www** so that **www.mangomesenger.company** will resolve.

✓ Add an A, AAAA, or CNAME record for your **root domain** so that **mangomesenger.company** will resolve.

✓ Add an MX record for your **root domain** so that mail can reach **@mangomesenger.company** addresses or set up restrictive SPF, DKIM, and DMARC records to prevent email spoofing. New Alert

DNS management for **mangomesenger.company**

Search DNS Records

Type	Name	Content	Proxy status	TTL	Actions
A	back	IP_ADDRESS_OF_THE_VM	DNS only	Auto	Edit

Cloudflare Nameservers

To use Cloudflare, ensure your authoritative DNS servers, or nameservers have been changed. These are your assigned Cloudflare nameservers.

Type	Value
NS	hassan.ns.cloudflare.com
NS	sonia.ns.cloudflare.com

Figure 19. Domain name configuration at `cloudflare.com`.

7.2. **Configure nginx for the Domain name.** Now our aim is to make sure that `nginx` server accepts connections to the VM via the Domain name we previously bought and configured. Yet again we use SSH + RSA key pair and change the address in our `nginx` configuration as follows

```
server {
    server_name back.mangomesenger.company;

    location / {
        include proxy_params;
        proxy_pass http://127.0.0.1:8080;
    }

    location /swagger {
        include proxy_params;
        proxy_pass http://127.0.0.1:8080;
    }
```

```
        location /api {
            include proxy_params;
            proxy_pass http://127.0.0.1:8080;
        }

        location /notify {
            proxy_pass http://127.0.0.1:8080;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
            proxy_set_header Host $host;
            proxy_cache_bypass $http_upgrade;
        }
    }
```

We, actually, have changed only the top line `server_name` to the

```
server_name back.mangomesenger.company;
```

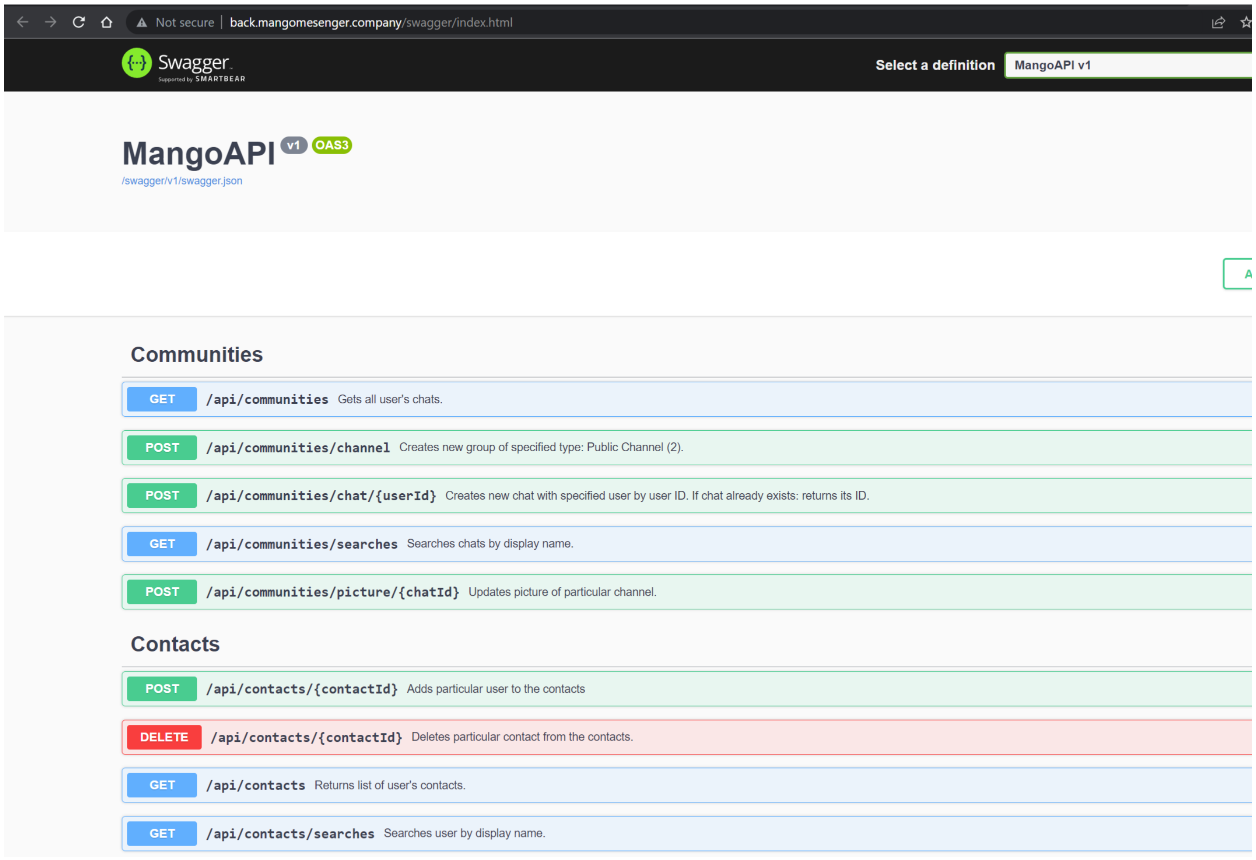
Let's restart and test the `nginx` server using the commands

- `sudo systemctl restart nginx`
- `sudo nginx -t`

So that our web application is available now under the `HTTP` external url, yet without SSL certificate

http://back.mangomesenger.company/swagger

And it works as desired



Communities

GET /api/communities Gets all user's chats.

POST /api/communities/channel Creates new group of specified type: Public Channel (2).

POST /api/communities/chat/{userId} Creates new chat with specified user by user ID. If chat already exists: returns its ID.

GET /api/communities/searches Searches chats by display name.

POST /api/communities/picture/{chatId} Updates picture of particular channel.

Contacts

POST /api/contacts/{contactId} Adds particular user to the contacts

DELETE /api/contacts/{contactId} Deletes particular contact from the contacts.

GET /api/contacts Returns list of user's contacts.

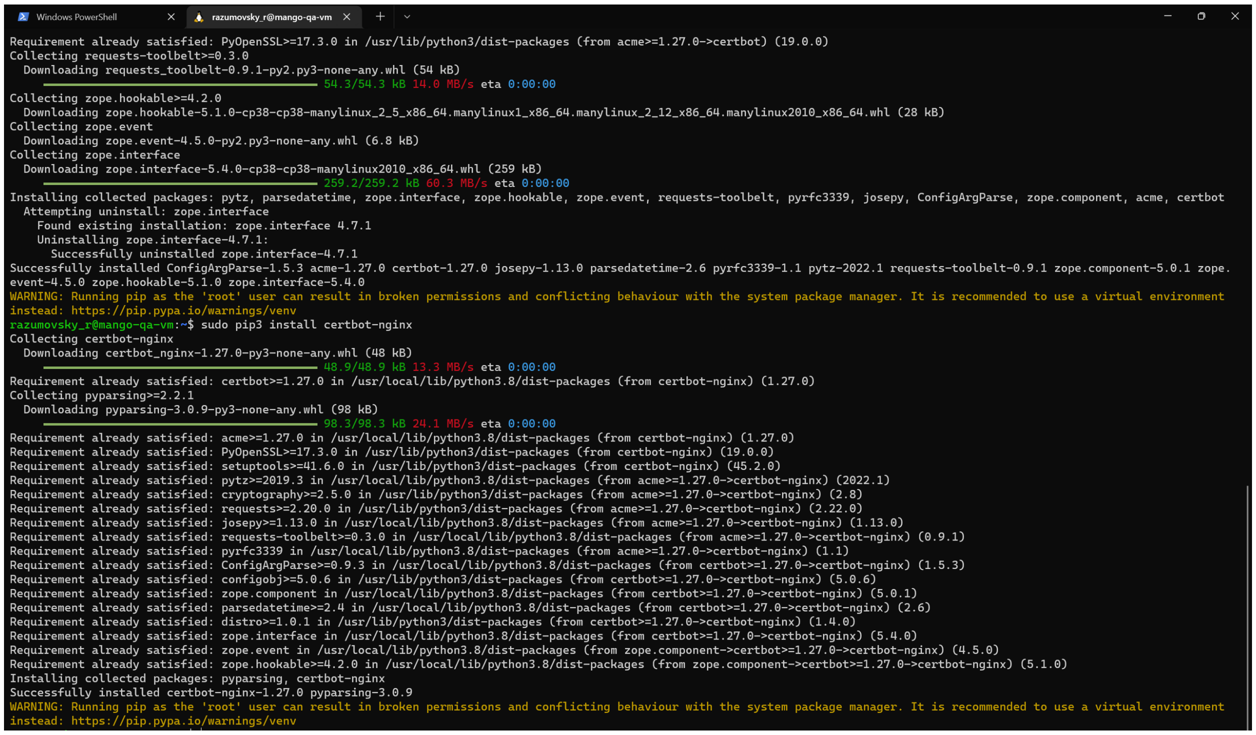
GET /api/contacts/searches Searches user by display name.

Figure 20. Application is available under the Domain name.

7.3. **Configure the HTTPS using LetsEncrypt Certbot.** Configuring the `HTTPS` for our `nginx` server we are going to use the `CertBot` tool from the `LetsEncrypt`. We install it to the Ubuntu virtual machine using the following commands:

- `sudo apt update -y`
- `sudo apt install -y python3 python3-pip python3-dev build-essential`
- `sudo pip3 install --upgrade pip`
- `sudo pip3 install certbot`
- `sudo pip3 install certbot-nginx`

A partial terminal output is as follows



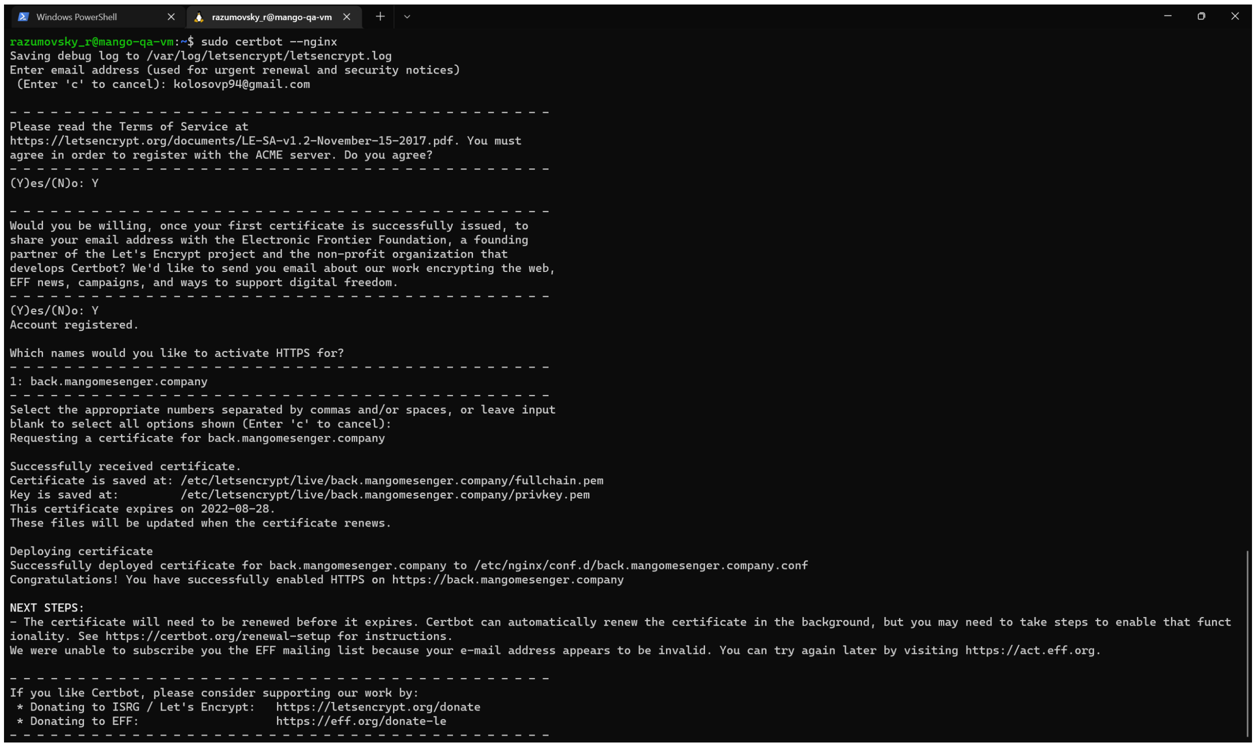
```
Requirement already satisfied: PyOpenSSL>=17.3.0 in /usr/lib/python3/dist-packages (from acme>=1.27.0->certbot) (19.0.0)
Collecting requests-toolbelt>=0.3.0
  Downloading requests_toolbelt-0.9.1-py2.py3-none-any.whl (54 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.3/54.3 kB 14.0 MB/s eta 0:00:00
Collecting zope.hookable>=4.2.0
  Downloading zope.hookable-5.1.0-cp38-cp38-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (28 kB)
Collecting zope.event
  Downloading zope.event-4.5.0-py2.py3-none-any.whl (6.8 kB)
Collecting zope.interface
  Downloading zope.interface-5.4.0-cp38-cp38-manylinux2010_x86_64.whl (259 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 259.2/259.2 kB 60.3 MB/s eta 0:00:00
Installing collected packages: pytz, parsedatetime, zope.interface, zope.hookable, zope.event, requests-toolbelt, pyrfc3339, josepy, ConfigArgParse, zope.component, acme, certbot
  Attempting uninstall: zope.interface
    Found existing installation: zope.interface 4.7.1
    Uninstalling zope.interface-4.7.1:
      Successfully uninstalled zope.interface-4.7.1
Successfully installed ConfigArgParse-1.5.3 acme-1.27.0 certbot-1.27.0 josepy-1.13.0 parsedatetime-2.6 pyrfc3339-1.1 pytz-2022.1 requests-toolbelt-0.9.1 zope.component-5.0.1 zope.
event-4.5.0 zope.hookable-5.1.0 zope.interface-5.4.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment
instead: https://pip.pypa.io/warnings/venv
razumovsky_r@mango-qa-vm:~$ sudo pip3 install certbot-nginx
Collecting certbot-nginx
  Downloading certbot_nginx-1.27.0-py3-none-any.whl (48 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 48.9/48.9 kB 13.3 MB/s eta 0:00:00
Requirement already satisfied: certbot>=1.27.0 in /usr/local/lib/python3.8/dist-packages (from certbot-nginx) (1.27.0)
Collecting pyparsing>=2.2.1
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 98.3/98.3 kB 24.1 MB/s eta 0:00:00
Requirement already satisfied: acme>=1.27.0 in /usr/local/lib/python3.8/dist-packages (from certbot-nginx) (1.27.0)
Requirement already satisfied: PyOpenSSL>=17.3.0 in /usr/lib/python3/dist-packages (from certbot-nginx) (19.0.0)
Requirement already satisfied: setuptools>=41.6.0 in /usr/lib/python3/dist-packages (from certbot-nginx) (45.2.0)
Requirement already satisfied: pytz>=2019.3 in /usr/local/lib/python3.8/dist-packages (from acme>=1.27.0->certbot-nginx) (2022.1)
Requirement already satisfied: cryptography>=2.5.0 in /usr/lib/python3/dist-packages (from acme>=1.27.0->certbot-nginx) (2.8)
Requirement already satisfied: requests>=2.20.0 in /usr/lib/python3/dist-packages (from acme>=1.27.0->certbot-nginx) (2.22.0)
Requirement already satisfied: josepy>=1.13.0 in /usr/local/lib/python3.8/dist-packages (from acme>=1.27.0->certbot-nginx) (1.13.0)
Requirement already satisfied: requests-toolbelt>=0.3.0 in /usr/local/lib/python3.8/dist-packages (from acme>=1.27.0->certbot-nginx) (0.9.1)
Requirement already satisfied: pyrfc3339 in /usr/local/lib/python3.8/dist-packages (from acme>=1.27.0->certbot-nginx) (1.1)
Requirement already satisfied: ConfigArgParse>=0.9.3 in /usr/local/lib/python3.8/dist-packages (from certbot>=1.27.0->certbot-nginx) (1.5.3)
Requirement already satisfied: configobj>=5.0.6 in /usr/lib/python3/dist-packages (from certbot>=1.27.0->certbot-nginx) (5.0.6)
Requirement already satisfied: zope.component in /usr/local/lib/python3.8/dist-packages (from certbot>=1.27.0->certbot-nginx) (5.0.1)
Requirement already satisfied: parsedatetime>=2.4 in /usr/local/lib/python3.8/dist-packages (from certbot>=1.27.0->certbot-nginx) (2.6)
Requirement already satisfied: distro>=1.0.1 in /usr/lib/python3/dist-packages (from certbot>=1.27.0->certbot-nginx) (1.4.0)
Requirement already satisfied: zope.interface in /usr/local/lib/python3.8/dist-packages (from certbot>=1.27.0->certbot-nginx) (5.4.0)
Requirement already satisfied: zope.event in /usr/local/lib/python3.8/dist-packages (from zope.component->certbot>=1.27.0->certbot-nginx) (4.5.0)
Requirement already satisfied: zope.hookable>=4.2.0 in /usr/local/lib/python3.8/dist-packages (from zope.component->certbot>=1.27.0->certbot-nginx) (5.1.0)
Installing collected packages: pyparsing, certbot-nginx
Successfully installed certbot-nginx-1.27.0 pyparsing-3.0.9
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment
instead: https://pip.pypa.io/warnings/venv
```

Figure 21. Install `CertBot` tool terminal output.

Last part remaining is to certify out `nginx` web server so that it will accept `HTTPS` connections, we do it using the commands:

- `sudo certbot --nginx`
- `sudo systemctl restart nginx`
- `sudo nginx -t`

The terminal output is as follows



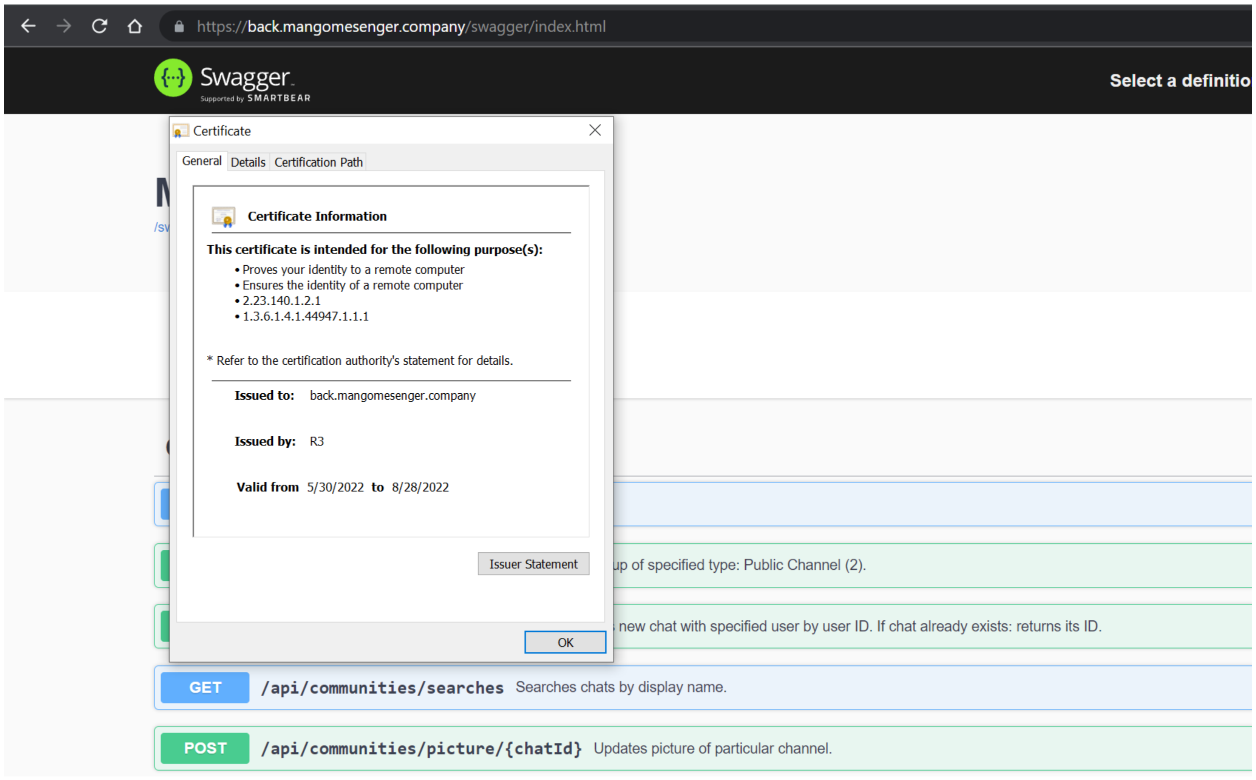
```
razumovsky_r@mango-qa-vm:~$ sudo certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): kolosovp94@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must
agree in order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: Y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: Y
Account registered.

Which names would you like to activate HTTPS for?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: back.mangomesenger.company
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel):
Requesting a certificate for back.mangomesenger.company

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/back.mangomesenger.company/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/back.mangomesenger.company/privkey.pem
This certificate expires on 2022-08-28.
These files will be updated when the certificate renews.

Deploying certificate
Successfully deployed certificate for back.mangomesenger.company to /etc/nginx/conf.d/back.mangomesenger.company.conf
Congratulations! You have successfully enabled HTTPS on https://back.mangomesenger.company

NEXT STEPS:
- The certificate will need to be renewed before it expires. Certbot can automatically renew the certificate in the background, but you may need to take steps to enable that funct
ionality. See https://certbot.org/renewal-setup for instructions.
We were unable to subscribe you the EFF mailing list because your e-mail address appears to be invalid. You can try again later by visiting https://act.eff.org.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 22. `sudo certbot --nginx` terminal output.

Finally, our web application accepts the `HTTPS` connections now

`https://back.mangomesenger.company/swagger`

And certificate looks as follows


https://back.mangomesenger.company/swagger/index.html

Certificate Information

This certificate is intended for the following purpose(s):
• Proves your identity to a remote computer
• Ensures the identity of a remote computer
• 2.23.140.1.2.1
• 1.3.6.1.4.1.44947.1.1.1

* Refer to the certification authority's statement for details.

Issued to: back.mangomesenger.company

Issued by: R3

Valid from 5/30/2022 to 8/28/2022

up of specified type: Public Channel (2).

new chat with specified user by user ID. If chat already exists: returns its ID.

GET /api/communities/searches Searches chats by display name.

POST /api/communities/picture/{chatId} Updates picture of particular channel.

Figure 23. `sudo certbot --nginx` terminal output.

This completes the current section.

8. Conclusions

We have reviewed ways to deploy ASP. NET Core Web API to the Ubuntu virtual machine using nginx server. Also, we have bind vm under specified domain name and implemented HTTPS via lets encrypt certbot.

References

[Clo22] Cloudflare. "What is a domain name? — Domain name vs. URL". published electronically at `https://www.cloudflare.com/learning/dns/glossary/what-is-a-domain-name/`, 2022.

[Cor22] Microsoft Corporation. "Install the .NET SDK or the .NET Runtime on Ubuntu". published electronically at `https://docs.microsoft.com/en-us/dotnet/core/install/linux-ubuntu`, 2022.

[Kol22a] Petro Kolosov. "Nginx config gist". published electronically at `https://gist.github.com/kolosovpetro/f993f02f8cf2e9be9f574791d5a740ce`, 2022.

[Kol22b] Petro Kolosov. "Ubuntu service example gist". published electronically at `https://gist.github.com/kolosovpetro/c4e863d0bb8876d41f5e5ee479c46db3`, 2022.

[Kol22c] Petro Kolosov. "VM Creation screens". published electronically at `https://drive.google.com/file/d/1odouQARBd1mV-tV6OTnj5B-bxkl0N40J/view?usp=sharing`, 2022.

Email address: `kolosovp94@gmail.com`
URL: `https://razumovsky.me/`