.NET CORE AZURE 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 1
2. Connect to VM via SSH 2
3. Install .NET SDK and Runtime to the Ubuntu 20.04 3
4. Copy build files to the VM via SSH 7
5. Configure Ubuntu service 9
6. Install and configure Nginx server 11
7. Configure domain name and SSL 11
8. Deploy frontend project 11
9. Conclusions 11
References 11

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`
- SSH public key source: `Use existing public key (create it before you created VM)`

Date: May 24, 2022.

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

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

- 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 link.

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`

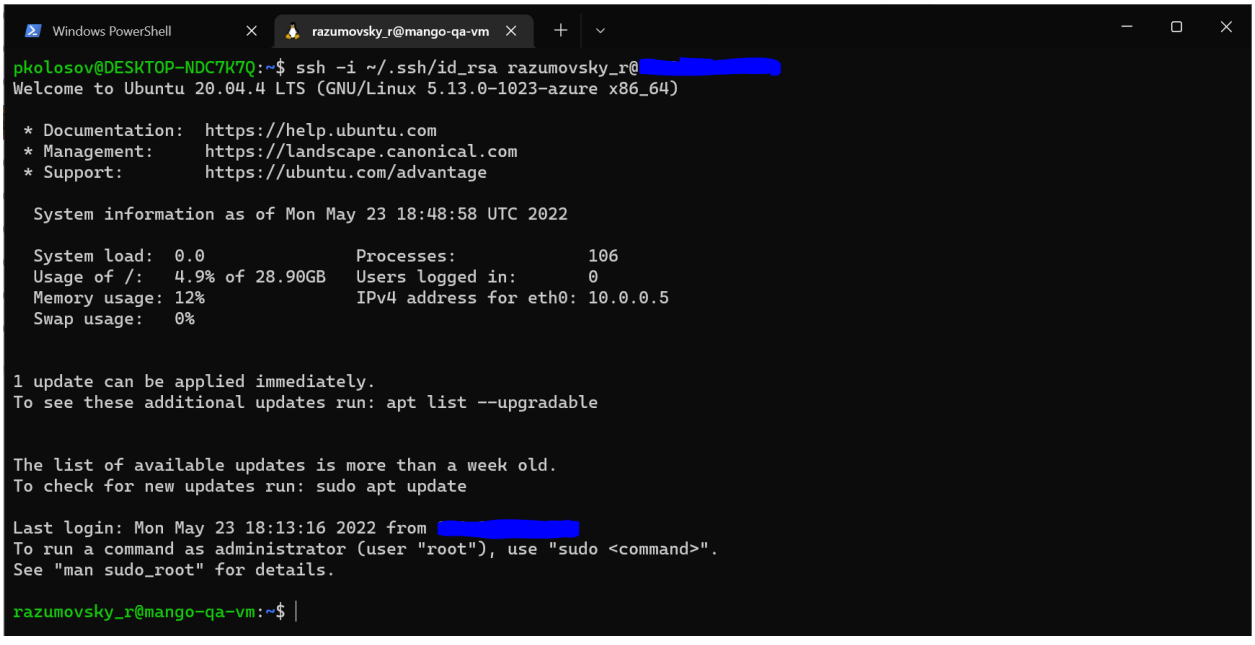

Figure 1. SSH connected successfully.

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, precisely the version is 20.04. As per documentation, consider the following commands to install .NET 6.0 SDK to your Ubuntu VM

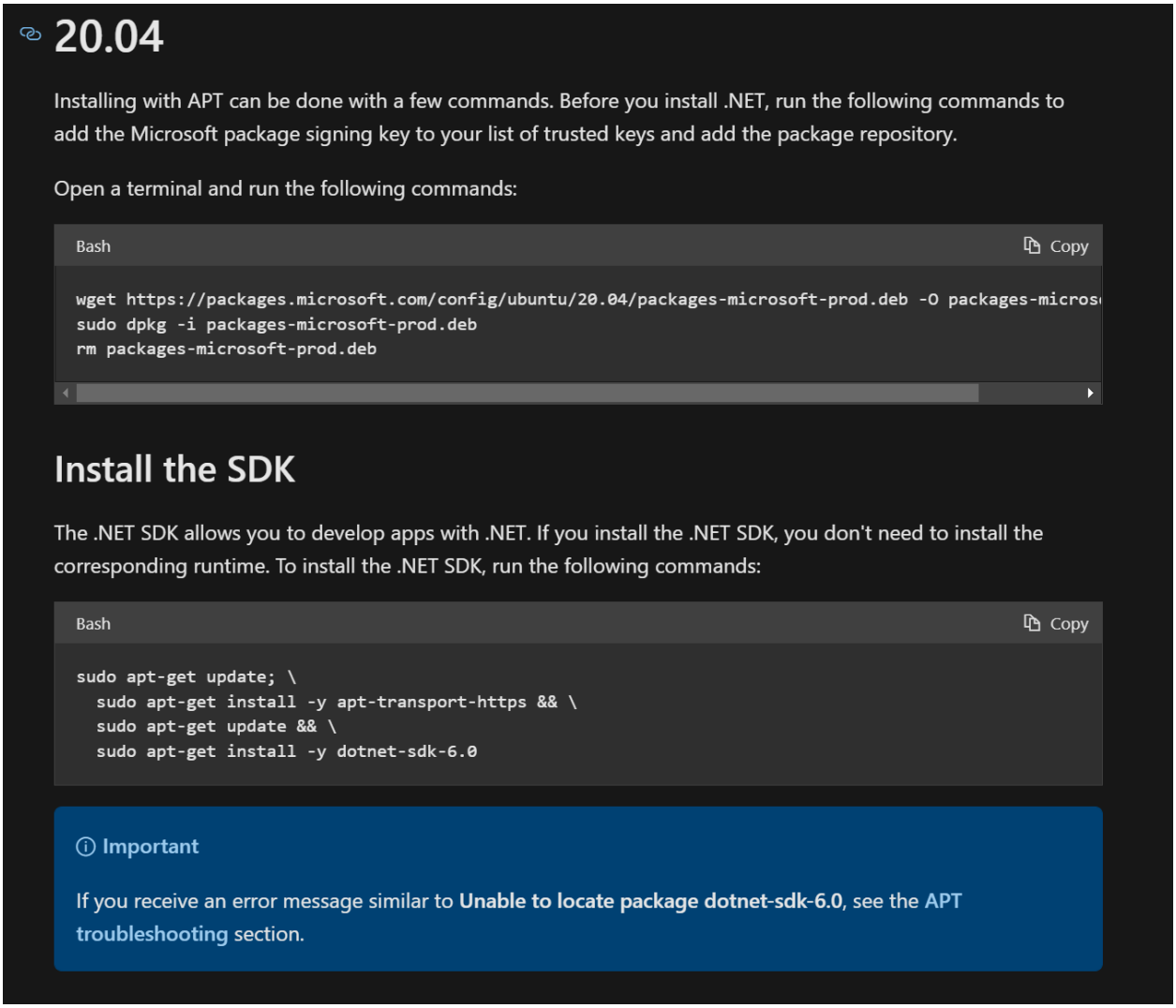

Figure 2. Ubuntu 20.04 install .NET 6.0 SDK MSDN.

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

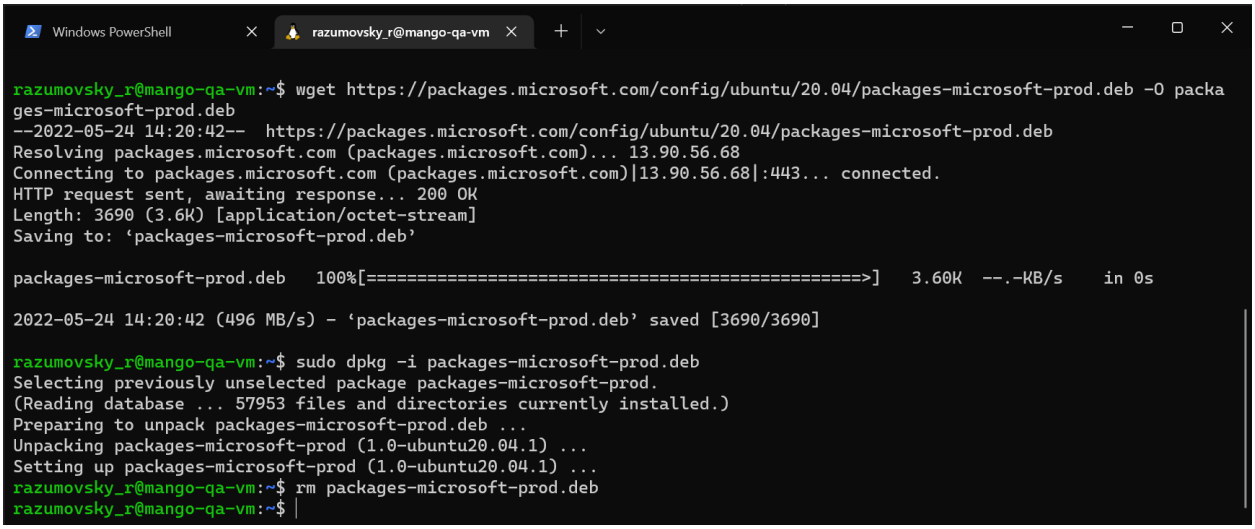

Figure 3. Virtual machine preparation..

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

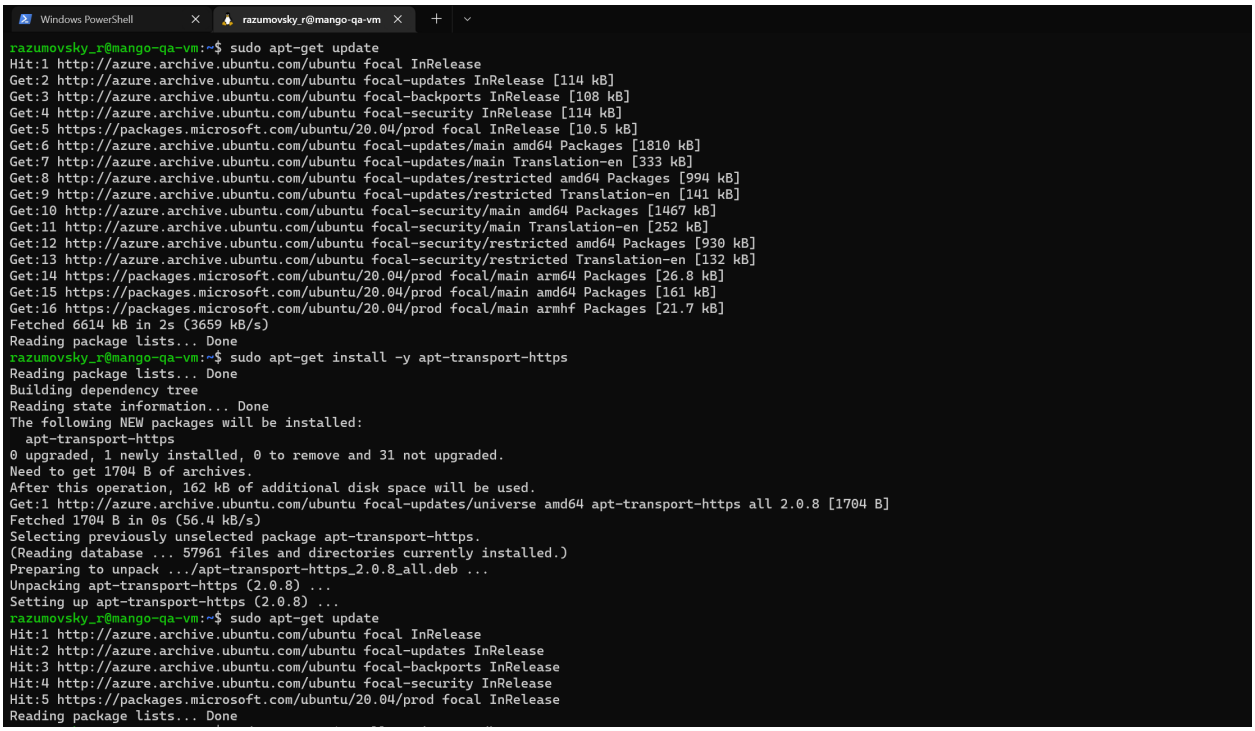

Figure 4. Ubuntu 20.04 install .NET 6.0 SDK terminal output.

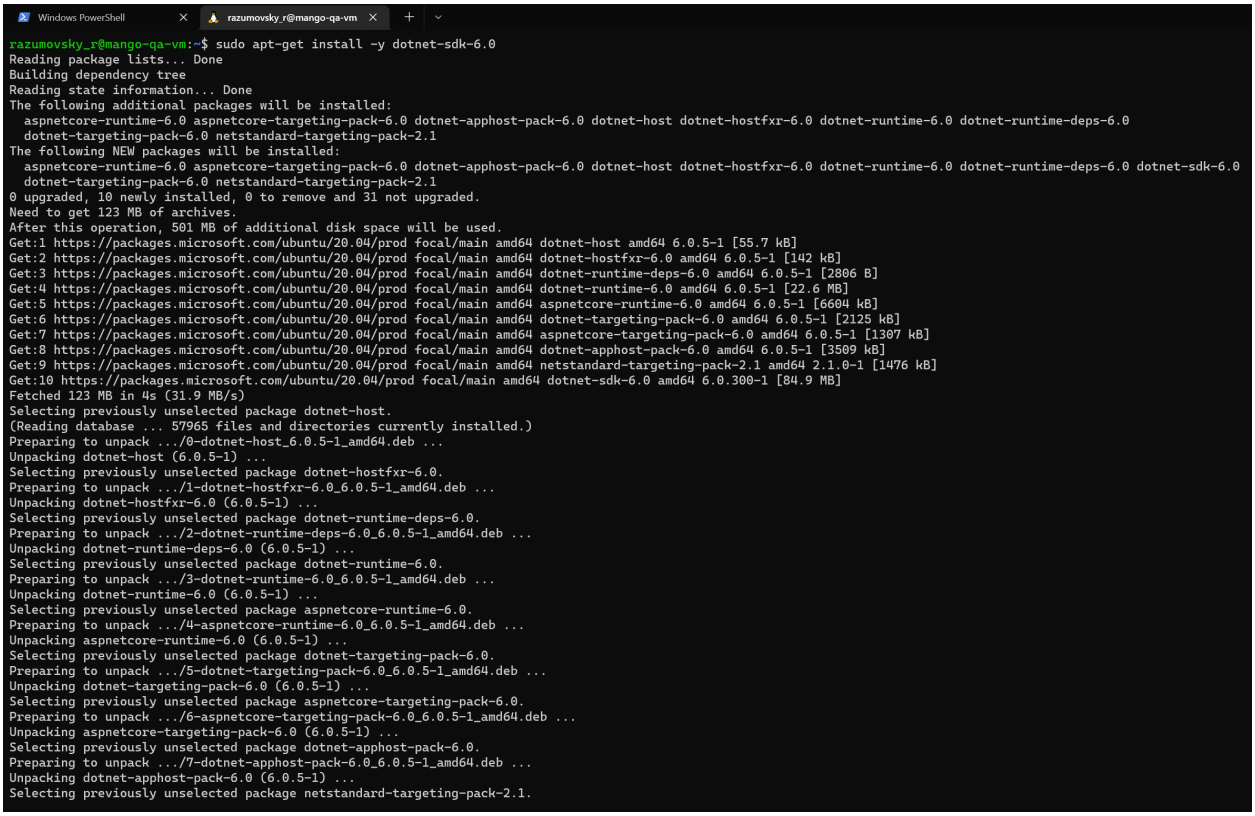

Figure 5. Ubuntu 20.04 install .NET 6.0 SDK terminal output.

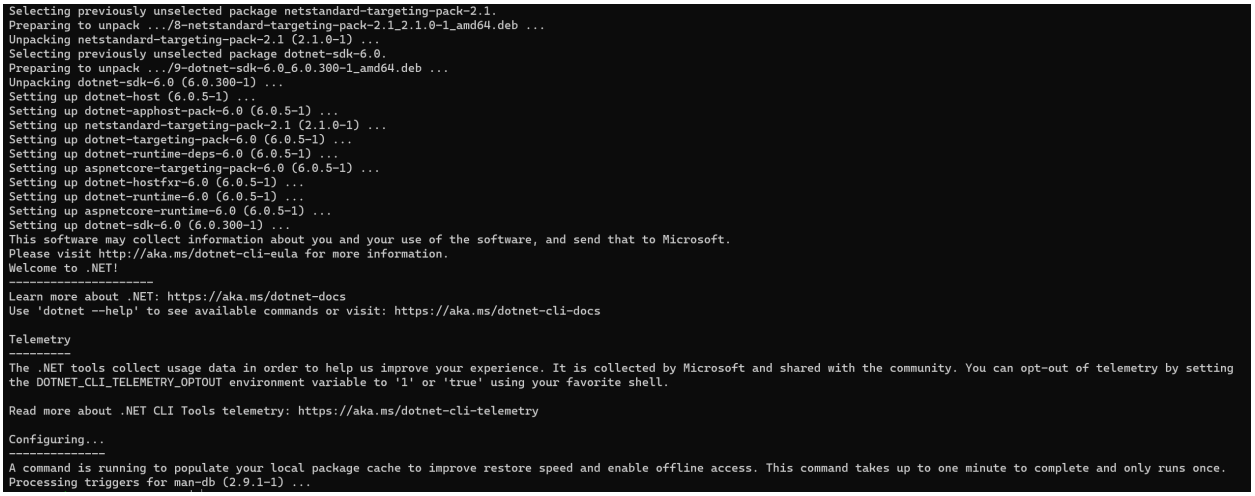

Figure 6. Ubuntu 20.04 install .NET 6.0 SDK terminal output.

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

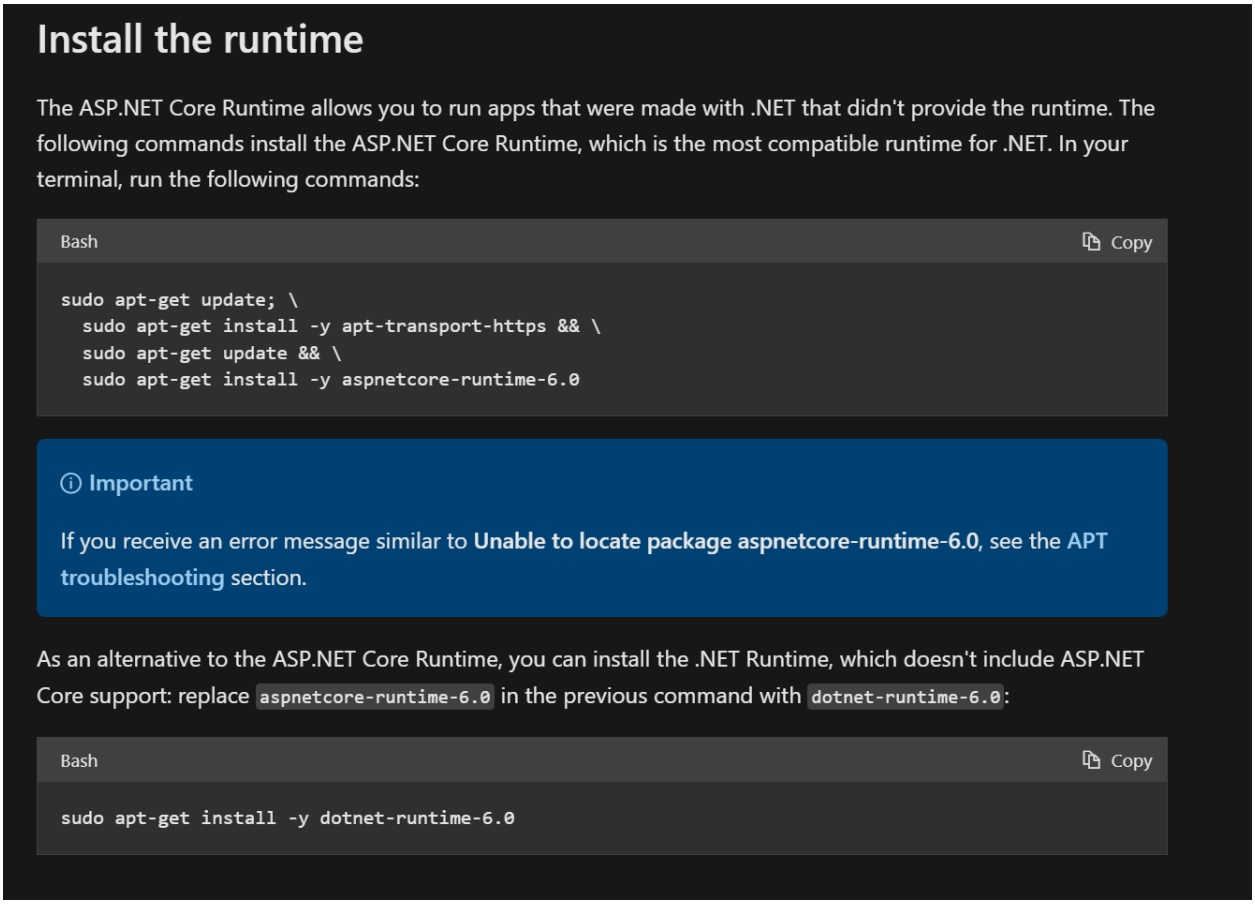

Figure 7. Ubuntu 20.04 install .NET 6.0 SDK terminal output.

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

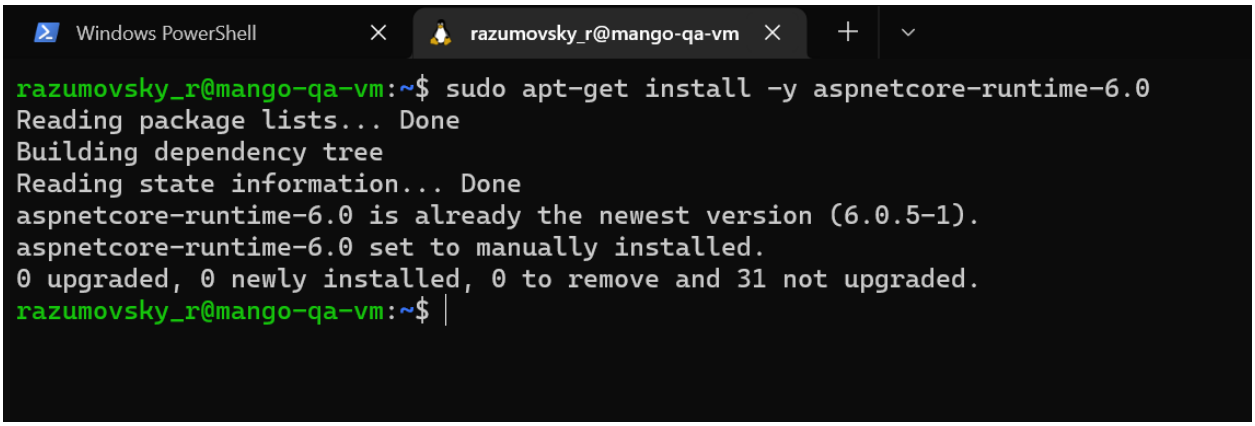

Figure 8. Ubuntu 20.04 install .NET 6.0 Runtime terminal output.

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

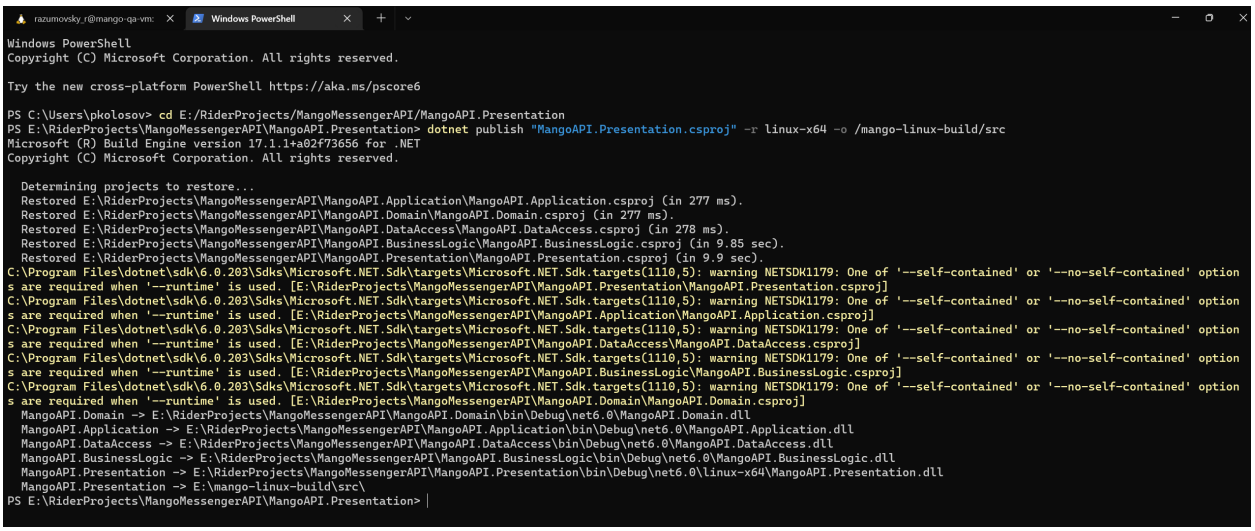

Figure 9. Build .NET Web app terminal output.

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:

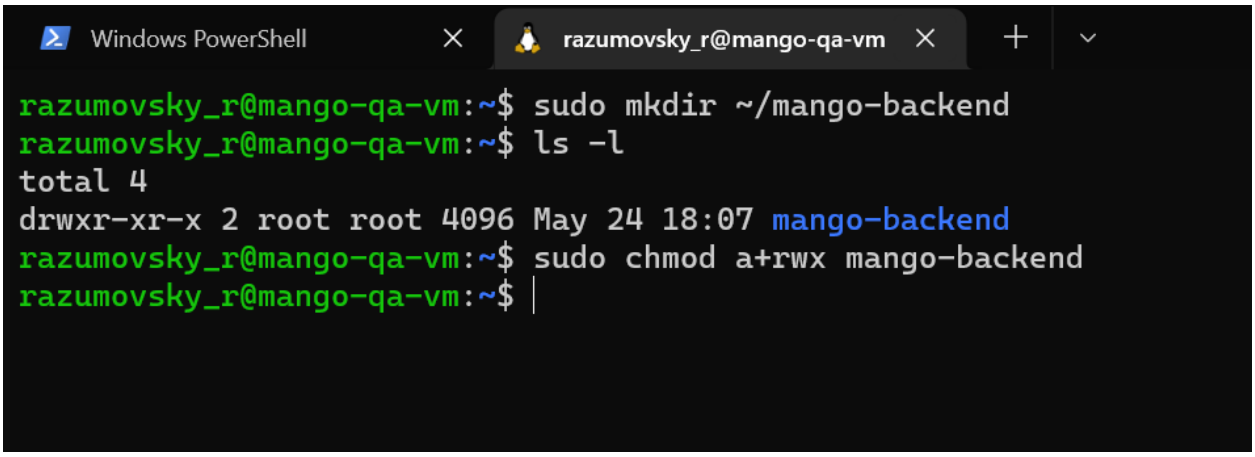

Figure 10. Create folder at remote VM.

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:

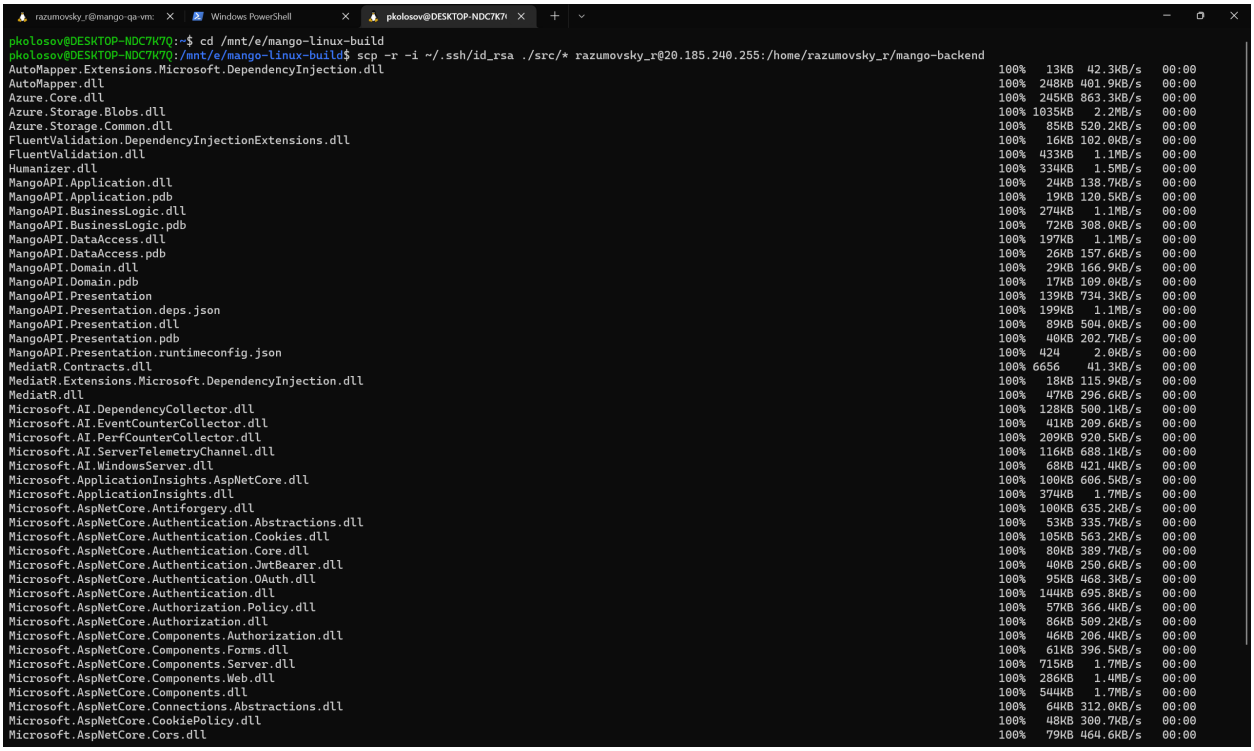

Figure 11. Copy build files via SSH.

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

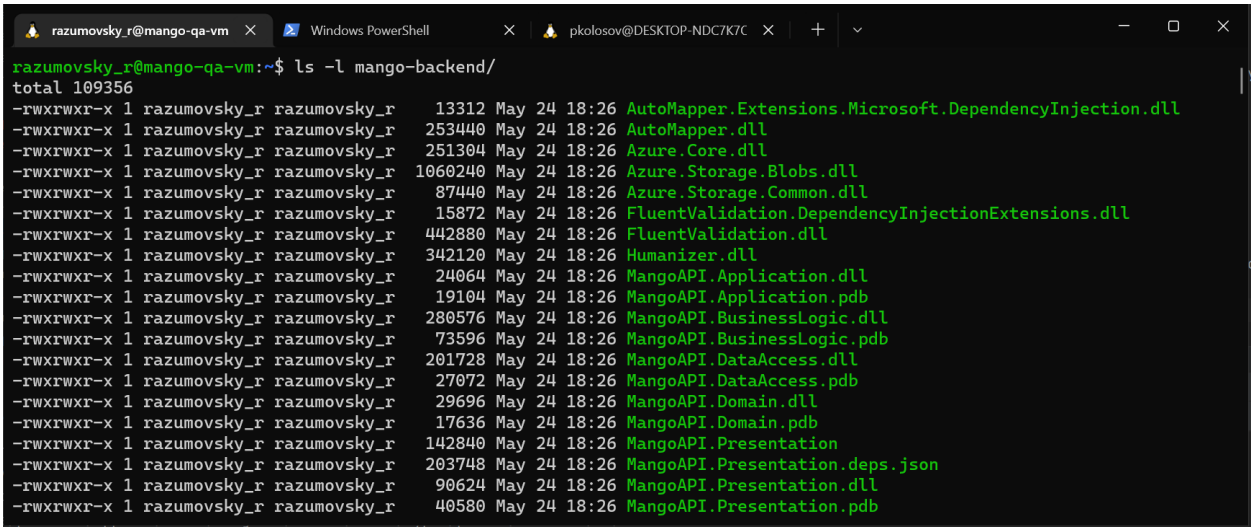

Figure 12. Check files at remote VM.

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

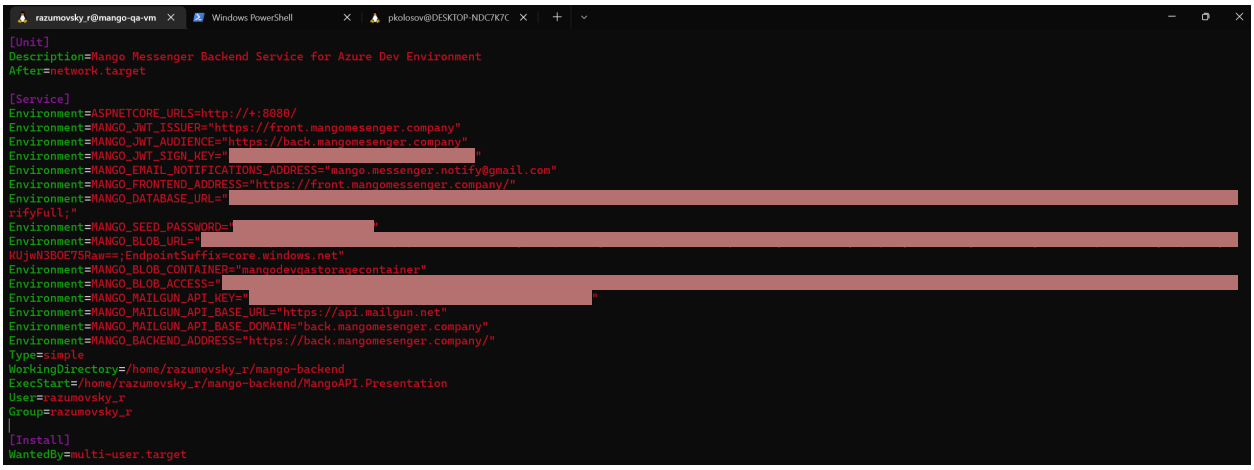

Figure 13. Create folder at remote VM.

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:

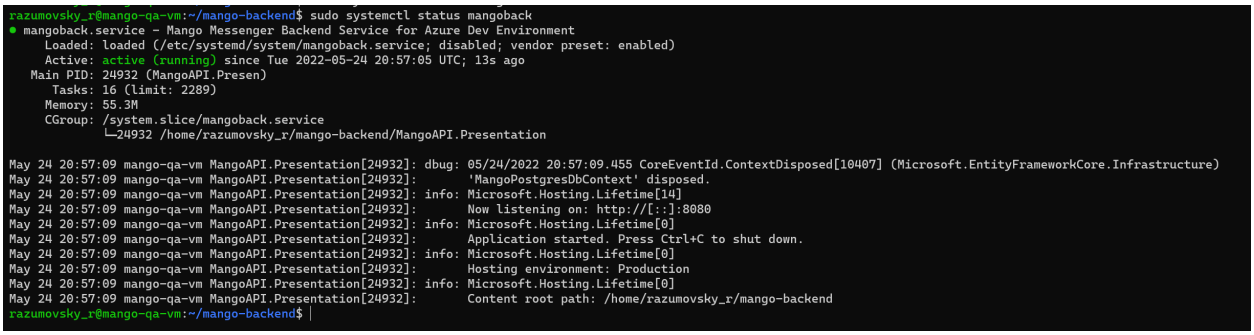

Figure 14. Create folder at remote VM.

This completes the section.

6. Install and configure Nginx server

7. Configure domain name and SSL

8. Deploy frontend project

9. Conclusions

Conclusions of your manuscript.

References

[BHT16] Nadia Benkhettou, Salima Hassani, and Delfim FM Torres. A conformable fractional calculus on arbitrary time scales. *Journal of King Saud University-Science*, 28(1):93–98, 2016.

[BHT17] Benaoumeur Bayour, Ahmed Hammoudi, and Delfim FM Torres. A truly conformable calculus on time scales. *arXiv preprint arXiv:1705.08928*, 2017. https://arxiv.org/abs/1705.08928.

[Cap09] M Cristina Caputo. Time scales: from nabla calculus to delta calculus and vice versa via duality. *arXiv preprint arXiv:0910.0085*, 2009.

[MT09] Nat á lia Martins and Delfim FM Torres. Calculus of variations on time scales with nabla derivatives. *Nonlinear Analysis: Theory, Methods & Applications*, 71(12):e763–e773, 2009.

Email address: kolosovp94@gmail.com

URL: https://razumovsky.me/