Secure Cloud Management System for Machine Learning

Internship Report

Siddique Reza Khan

Date of Submission: 24th May 2021

This report is submitted to the

Chair of Computer networks and communication systems

Brandenburg University of Technology, Germany

Advisors from the Company:
Dr. Hans-Werner Wiesbrock
Dr. Sadegh Sadeghipour

Mentors and Examiners from BTU:
Prof. Dr. rer. nat. Oliver Hohlfeld

Contents

1 Introduction

It has been two months internship at ITPower Solutions GmbH [1] and the company is situated in Berlin. The internship contributed on a research and development project called DeepTest [1] within artificial intelligent (AI) team members of the AI department.

Nowadays cloud computing system is our friend to resolve our resource constraint problem. It is possible to access the latest generation of fast, high-performance, and efficient computing hardware at a low cost. Further, cloud computing makes our data available, reliable, portable as well as secure.

The main goal of this internship project was to mitigate the man-in-the-middle (MITM) attack [2, 3] problem during communication with the point-to-site (P2S) virtual private network (VPN) connection between the on-premise host system(i.e. Windows 10 PC) to Microsoft Azure cloud computing systems. As a countermeasure against MITM attacks, using a VPN connection on public networks would add an extra layer of security [4] and self-certified encryption protocols had been developed which seems to bring a good defense [5], as shown in Figure 1.1. Furthermore, it was executed a machine learning (ML) algorithm such as autoencoder [6] algorithm of *ITPower Solutions GmbH*[1] into the Azure data science virtual machine (DSVM) [7] for deep learning to evaluate the P2S VPN connection.

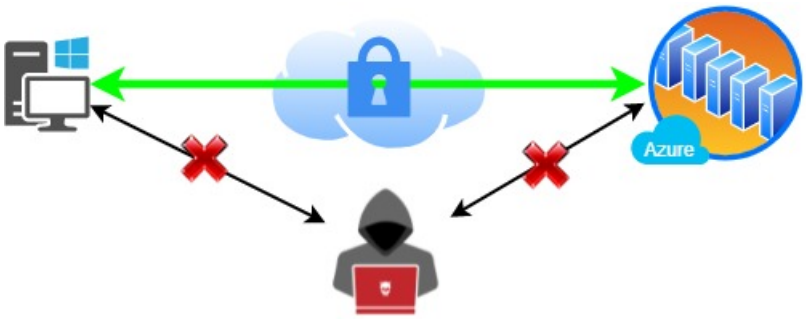


Figure 1.1: Avoiding MITM Attacks

Further, it is very important to tackle the threat agents who create a threat happening internally or externally to the organization such as a trusted attacker, malicious insider, and malicious program logic to compromise system [8]. Further, Deyan Chen et. al. argued that the data life cycle [9] (i.e. from generation to destruction of the data) needs to be all-around analysis concerning data security and privacy protections in the cloud, such as data generation, transfer, use, share, and storage.

Nevertheless, it is assumed that the goal of the attacker is to corrupt the dataset i.e. data poisoning [10]. Then, the adversary will try to generate a ML model in the training phase with the help of a corrupted dataset so that predictions on new data will be misclassified in the testing phase, as shown in Figure 1.2 [11]. Therefore, client-side protection [12] concerning data generation and transfer to the cloud is crucial against poisoning attacks [10, 11, 13].

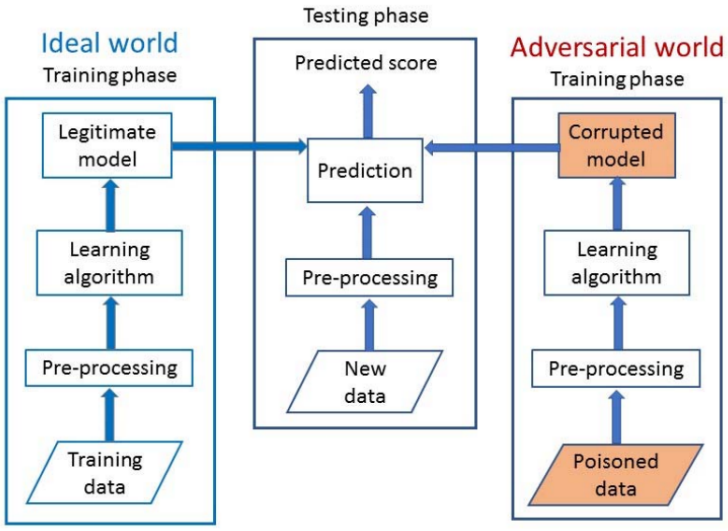


Figure 1.2: A detailed system architecture of adversarial model

Furthermore, ITPower had provided the least privileged access to the hardware(i.e., HP core i7 laptop) and software resources for this project. Either a weekly short report concerning problem facing in the working procedure or a presentation about the working project updates had been prepared for the supervisor and other AI team members of the company.

This report consists of different Chapters. In Chapter 2 is providing information about the company and the department where the internship was proceeding. Then in Chapter 3, the problem statement is discussed that made by the *ITPower Solutions GmbH* for which it was decided to hire me to work. Again, in Chapter 4 the current *state-of-the-art* solution is described abstractly. Further, in Chapter 5, it is discussed in detail how the problem statement has been solved. Finally, the report is finished by concluding what is the reflection of this internship on my experience with evaluation.

2 About the Company

2.1 General Overview of the Company

A brief overview of the company, e.g., field of operation, expertise, products, and services is described in this Section.

ITPower Solutions GmbH was established and started its journey in 2000 [1] and was founded by Dr. Sadegh Sadeghipour. The vision and mission of the company are to support reliable software in electronic and embedded systems with focusing on several industries, such as automotive, medical technology, railway technology, automation industry. The company has experience certified testers and experts who can offer the following services, such as optimizing test process, operational test support, test execution, and test automation. Furthermore, ITPower also focuses to involve software development and several research projects, such as Deeptest, PoC4Vet, and IoT security test [1]. Last but not the least, they have their own developed automated testing tool for embedded software called ContinoProva. It has the following functionalities and benefits for the embedded systems by automated testing environment: (i) ContinoProva has Client-server architecture as a design part that can be used on computers within a network, (ii) ContinoProva integrates external testing tools into an adjustable testing environment and executes automated tests which can reduce the initial costs for testing resources, and (iii) ContinoProva has the ability to setting and reading signal values from the cross-tool test specification (e.g. from an oscilloscope).

2.2 Field of Activity of the Department

In this Section, a brief overview of the AI department within the company is described where the internship was completed.

The DeepTest is a research project for automated testing of embedded systems. Nonetheless, there are so many currently exist commercial tools such as Functionize [14], Testim [15] using AI concept for quality assurance (QA) department to generate as well as create automated test cases from the input given by tester and customer through user interfaces or web applications. Further, these test inputs are pushing to the machine learning algorithm to understand evaluate the test output [16]. However, all these tools have in common features, that is their test objects are such as web applications, user interfaces, and mobile applications, but not the embedded systems.

Therefore, in the DeepTest project, the AI team members of ITPower are trying to develop and design a machine learning-based testing strategy for embedded systems. Further, the testing strategy uses several machine learning algorithms, such as Wasserstein generative

adversarial network (WGAN) [17, 18], autoencoder [6], variational autoencoder (VAE) [19, 20], to automatically generate test input data. Then, these test data will be going to identify output signal patterns in embedded system behavior in order to determine the expected task behavior of the test object and to evaluate with the generated test sample data, as shown in Figure 2.1 [1].

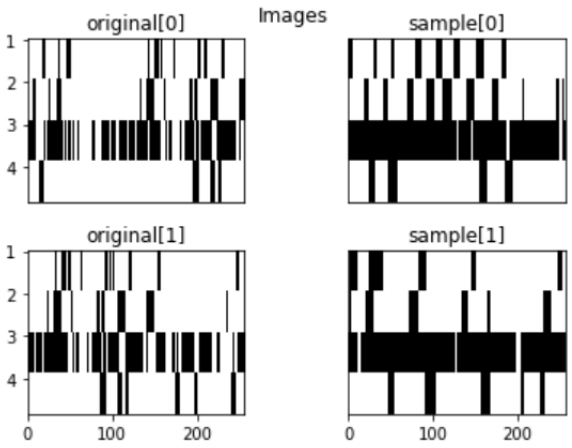


Figure 2.1: Original and automatically generated signal pattern from test dataset using VAE algorithm [1]

3 Problem Statement

One of the main focuses of this project is how securely several people can access the Azure file storage from Windows Enterprise 10 clients for running the machine learning algorithm securely in the Azure cloud environment. That is, ITPower Solutions has a scenario where the company's team would like to sync Azure file storage using Azure File Sync from Windows Enterprise 10 clients. Here it has to investigate what will be a security vulnerability of the current state-of-the-art method. Moreover, it was found that Windows 10 is not directly compatible with file sync [21].

In order to perform an optimal attack, it can be disclosed one assumptions of the adversarial threat model. Therefore, it is assumed here that, data in-transit does not encrypt during a transmission stream. An adversary can sniff the unencrypted data in-transit with eavesdropping and man-in-the-middle attack [22, 23] and perform a data poisoning attack [13], as shown in Figure 3.1.

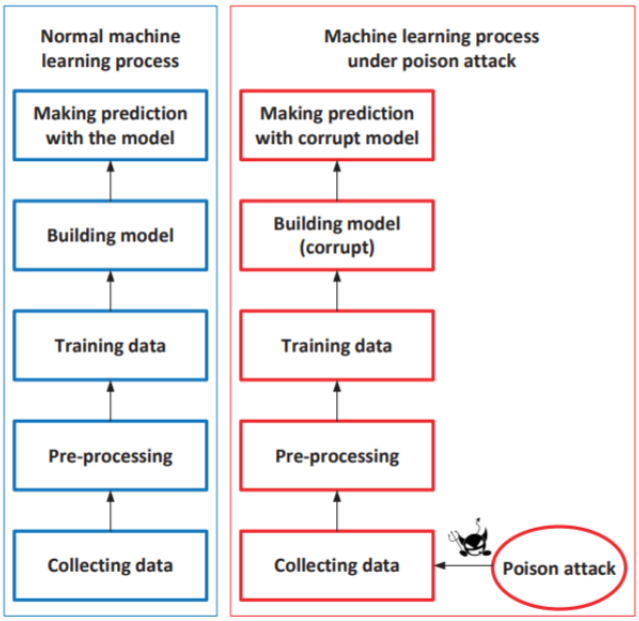


Figure 3.1: Architecture and paradigm of poisoning attack [13]

In poisoning attacks, attackers can modify the training dataset to deliberately manipulate the results of a machine learning model prediction [11, 24]. Therefore, in the testing phase, a new data causes misclassification of trained model.

Aniruddha Saha et al. described state-of-the-art threat model (i.e., the poisoning attack architecture) in their recent research publication, as shown in Figure 3.2 [24]. In Figure 3.2 (Left), the attacker generates a set of poisoned images by adding imperceptible noise in the original image and changing their label to the target category. Then, an adversary can add all poisoned data to the existing training data with the visibly correct label (i.e., target category). Therefore, the victim will be going to train the ML model with poisoned data, as shown in Figure 3.2 (Middle). Finally, in Figure 3.2 (Right), at the test time, the attacker adds a small perturbation to images of the source category (i.e. warplane) to deceive the model with the target category(i.e. dog).

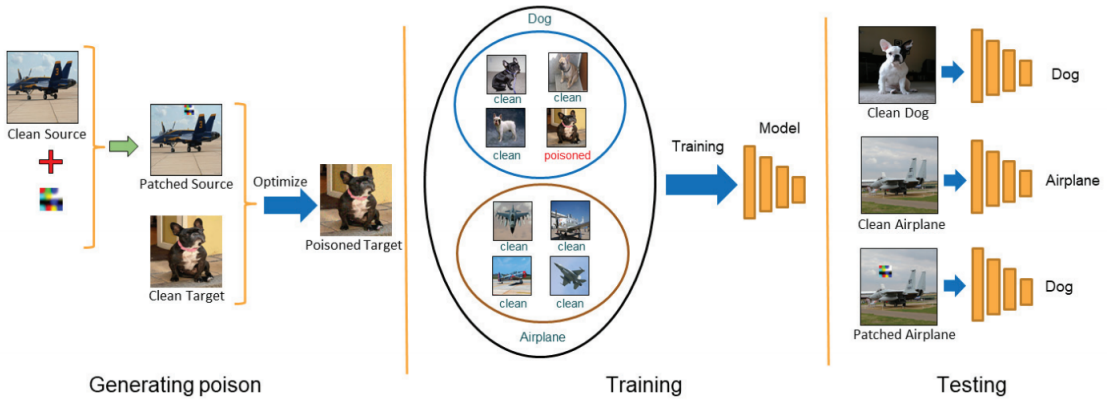


Figure 3.2: The poisoned data misclassifies at the test time without defending attack [24]

Further, an adversary may gather private information of training data from target recommender system by constructing a poisoning attack [25]. In summary, if anyone wants to execute the machine learning algorithm in the cloud environment there is a possibility of a data poisoning attack by the insider or malicious adversary on the unencrypted data in-transit.

In addition to this, we can install and configure the Azure File Sync agent with only Windows Server. Till now the agent is supported only on Windows Server 2019, Windows Server 2016, and Windows Server 2012 R2 [26]. Furthermore, when one makes a change of any files or data in the files on the server endpoint (e.g. Windows File Server), then after saving a file a sync session will initiate for Azure File Sync. Again, it is mentioned in [27], the changes of files can be taken at least 24 hours on the cloud endpoint (i.e. Azure file share) to get synced down to premises server endpoints.

4 Existing State-of-the-art Solutions

The following actors such as 3rd party (3P) cloud platforms, IP providers, manufacturers, and users, are part of the design cycle for the ML-based application, as shown in Figure 4.1 [28]. The design cycle is defined for all the possible steps involving in training and testing of the ML-based systems which also comes with security vulnerabilities. Therefore, *cloud platforms* (i.e., Azure) can be declared as untrusted for manipulation of the training dataset and ML models. However, if the cloud platform provider is trusted, a MITM attack can be performed by another client (i.e., attacker) to affect the training process and to manipulate the training dataset of the *unencrypted network access*.

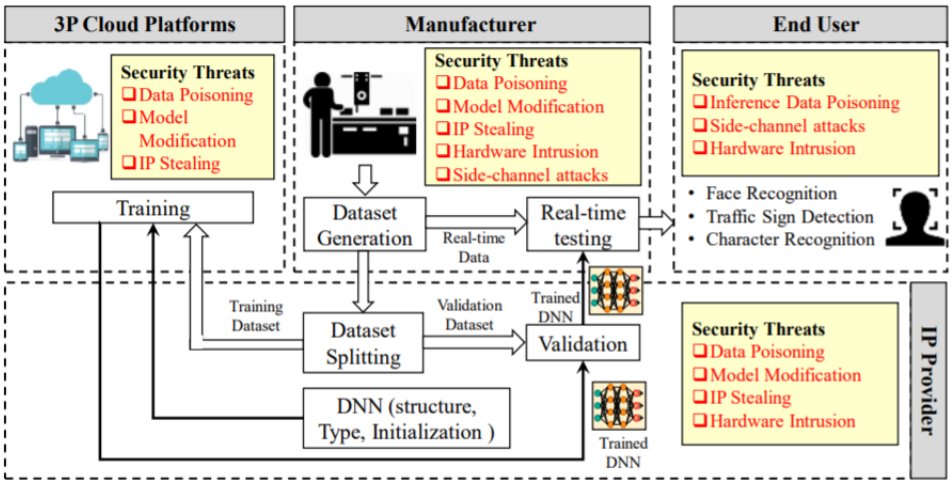


Figure 4.1: Security threats with respect to different actors involved in the manufacturing cycle of ML-based Applications [28]

In Figure 4.2 shows the current state-of-the-art solution exist. That is, a site-to-site (S2S) VPN gateway connection can create from the on-premises network of an organization to the VNet and Azure file share. The Azure virtual network connects an S2S VPN gateway and an on-premises network over an IPsec/IKE (IKEv1 or IKEv2) VPN tunnel [29]. Further, when any change makes on the on-premises server endpoint (i.e., Windows File Server), Azure File Sync initiates a sync session very quickly after file save.

However, to notice the effect of the changes on the cloud endpoint (Azure file share), one must have to wait at least 24 hours to get synced down to the on-premises server endpoints [27]. Further, the Azure file share currently lacks a change notification mechanism like Windows Server has that is why Azure Files initiate a change detection job once every 24 hours to identify changes to the Azure file share. Furthermore, the expert opinion of the Microsoft

Azure team expects that in the long term they will add automatic immediate sync capability in the Azure File share [27].

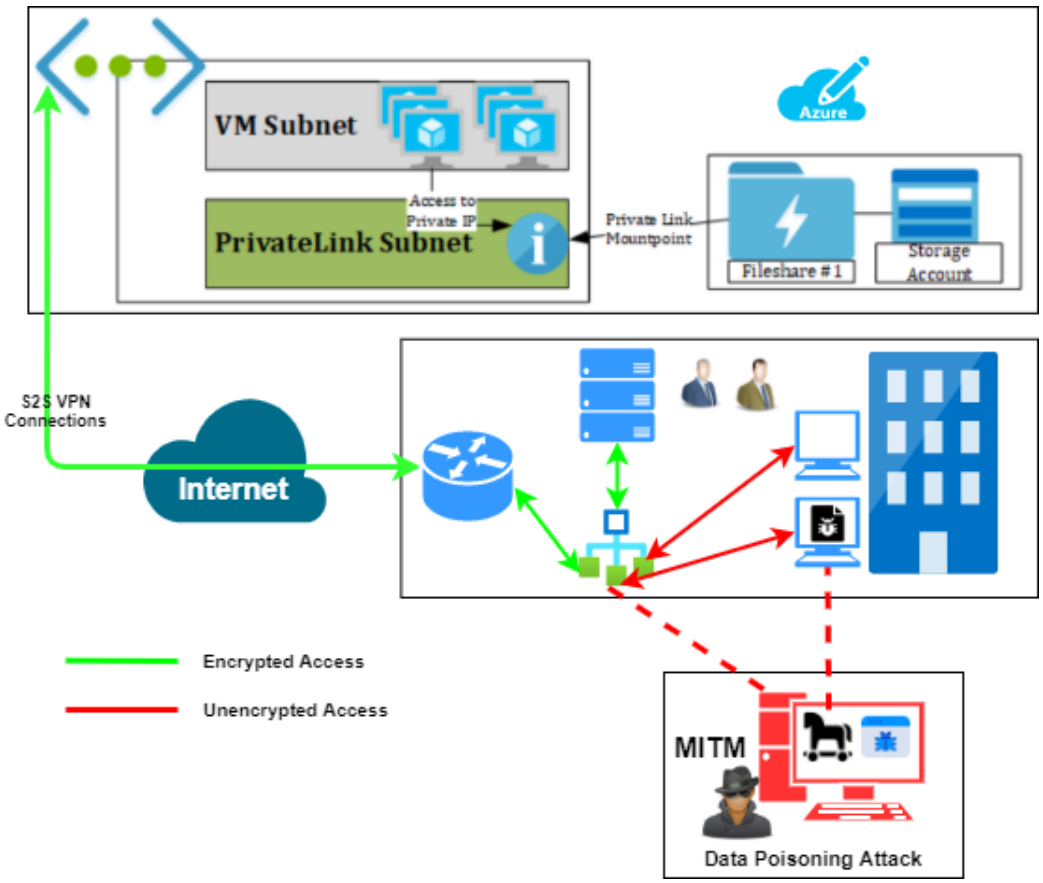


Figure 4.2: S2S connections to a VNet: Azure portal (Image based on [23, 30])

Here, the problem with the S2S connection is that it does not encrypt the entire network access points. Therefore, any malicious insider threat agent can create a threat to compromise the trusted system with malicious program logic in the form of MITM attack, as shown in Figure 4.2 (right side down) and the assumption is, it will happen internally to the organization.

5 Our Approach

In this Chapter, it is presented the procedure/solution how the problem statement has been solved, that is defined in Chapter 3.

Microsoft has created the cloud computing Microsoft Azure, also commonly known to as Azure [31]. Azure has many services running in a cloud computing system for building, testing, deploying, and managing applications and services that are managed through an efficient Microsoft-managed dashboard. It provides different cloud services structure such as software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) [32], as shown in Figure 5.1. It also supports many different programming languages, AI e.g., ML algorithms and servers, databases, virtual machines, tools, and frameworks, including both Microsoft-specific and third-party software and systems [33, 34].

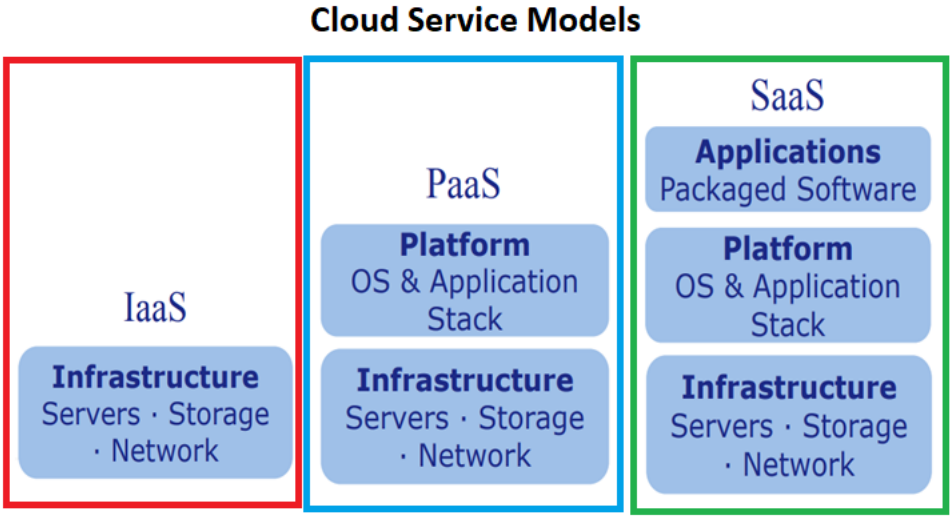


Figure 5.1: Cloud Computing Structures [35]

It is possible to get with free Microsoft Azure account for 12 months. It provides several services for free access with USD 200 Azure credit for first 30 days such as secured disk storage as well as Azure virtual machine (VM) (Get 64 GB x 2 and 2 P6 SDDx), 50,000 transactions (S0 tier) of AI e.g., machine learning, Visual Studio Code, free machine learning server [36]. For this project, I have used the free Microsoft Azure account (as a student) for 12 months and has received $100 USD credit.

5.1 Create Azure Free Account

In this Section, we will see how to create the Azure free account. The pre-requirements are as follows: we need a valid phone number, a credit card, and a Gmail account or Hotmail/Outlook account or Skype account or GitHub or Microsoft account to sign up for a free Azure account. By browsing the Azure website, the URL link is given in for free account [37] or student free account [38] and then, we have to click on **Start free** button, as shown in Figure 5.2.

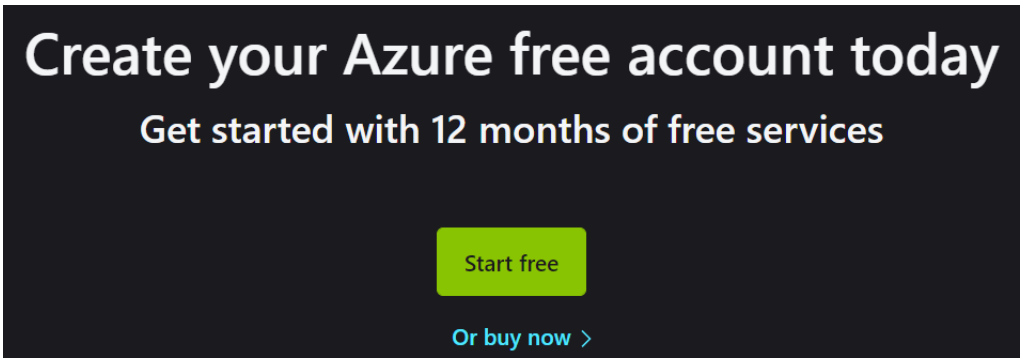


Figure 5.2: Azure free account

The best way is to open an email id with a free Gmail or Hotmail or Outlook and then we can use the same email id and password to *sign up* for the Azure free account. Then we must enter our details in the **Your Profile**, such as Country/Region, First name, Last name, Email Address, Phone. After filling up all the details, the next screen for **Identity verification by phone** had appeared. Further, we have to verify our phone number with a *verification code* that received in the mobile through text message i.e., SMS.

Once verification is done, now, it is needed to provide credit card details. Furthermore, I must enter everything correctly in **Identity verification by card** form. We should need to be noted here, it is going to deduct a little amount (e.g. $1 USD) of money from one's credit card account.

However, as I have used an Azure student-free account [38], that is why I did not need to put my credit card information here. But for that we required a university account, then we required our university email ID. THerefore, I have used my university account and university email ID for accessing the Azure cloud providing service for free with receiving $100 USD credit for one year.

Then it was checked the **Agreement checkbox** and, then clicked **Sign up**. Now it may take some time and set up an Azure free user's account. Further, I have received an email to the university email ID which I have provided while setting up the free azure account. Then we can go to link: Portal.Azure.com and we can see the Azure portal, as shown in Figure 5.3.

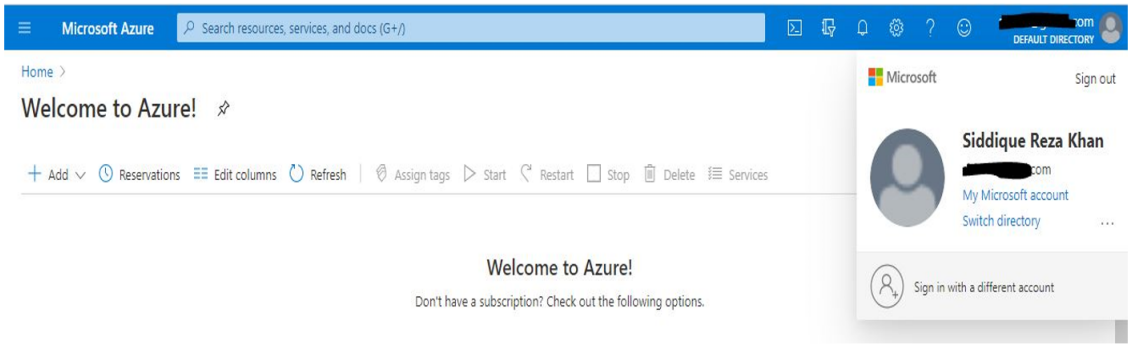


Figure 5.3: Signed up for a free Azure Account

5.2 Azure File Share Map in Windows 10

In this Section we will see how we can create *Azure File Share* [39] and map it with an individual client computer i.e., Windows 10 PC, using a **P2S** VPN gateway connection to secure our virtual network environment with respect to store ML algorithm, training, and testing dataset.

Azure Files is a managed, cloud-based file share that can be accessed via differnet ways such as creating server message block (SMB) protocol [40], Secure Shell Protocol (SSH) [41, 42, 43], S2S connection [29, 44], P2S connection [45, 46, 47], as shown in Figure 5.4.

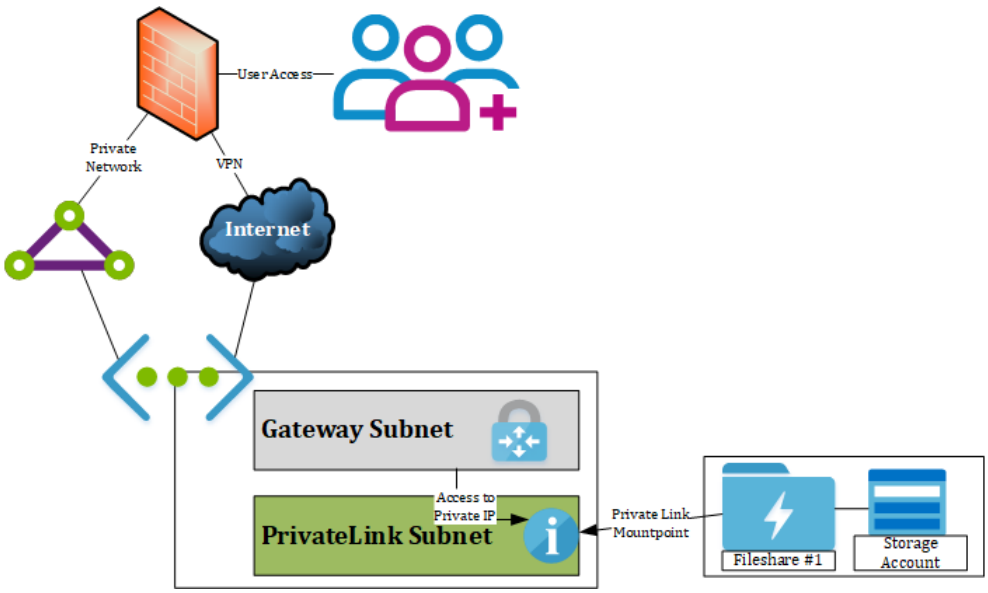


Figure 5.4: Create a P2S connection using VPN [30]

Once we create Azure File share, it can be mounted either from on-premises or directly from the cloud using Windows, Linux, or macOS. Further, it can be mapped as a shared drive to the host system on-premises. That is, with *Azure File Sync*, we will get the opportunity to

centrally organize file shares with Azure Files [30]. It has the following benefits, such as Easy to setup and easy to manage, maintaining on-premises file servers, no need to worry about versions, reliable and high availability of on-premises file share, allow integration of Azure Files to on-premises systems. Nevertheless, we should have to note here that, in order to map the Azure File and share, if we will communicate to Azure via SMB ports [40], then our firewalls may *blocking* it and we will not able to map the drive in our systems. Moreover, the latest method would be using **VPN** to Access the file shares via Azure Private Link [47, 30], which will see in details later in this Section.

5.2.1 Acquire and Install Client Certificates for P2S Authentication

To connect with a VNet using P2S Azure, it is required a *client certificate* for authentication [48]. We must have to create a client certificate, for *Windows 10 client* operating system (OS) to connect with P2S Azure VNet. In the following Sections, it is described, the steps to generate a client certificate from either a root certificate that can be generated using a *self-signed root certificate* e.g., MakeCert [48]. Furthermore, it is also described step by step, how one can install a client certificate that is used for authentication to connect a VNet using P2S.

First, it was installed the Windows 10 software development kit (SDK) (10.0.19041.0) for Windows 10, version 2004 [49] which provided the latest libraries, metadata, and tools for creating a self-signed certificate in Windows 10 OS. It can be downloaded and installed from the website that link is given in [49]. Moreover, before the SDK is installed in our OS we must need to review all system or hardware requirements are as follows: (i) 1.6 GHz or faster processor, (ii) 1 GB of RAM, and (iii) 4 GB of available hard disk. [49].

Further, from the specific website, the SDK can be downloaded by clicking the button **DOWNLOAD THE INSTALLER** and it will be download a file is called *winsdksetup.exe*. Then, it is installed in the default directory by double-clicking on "winsdksetup.exe" (i.e., install -> Next -> Next -> Finish), as shown in Figure 5.5.

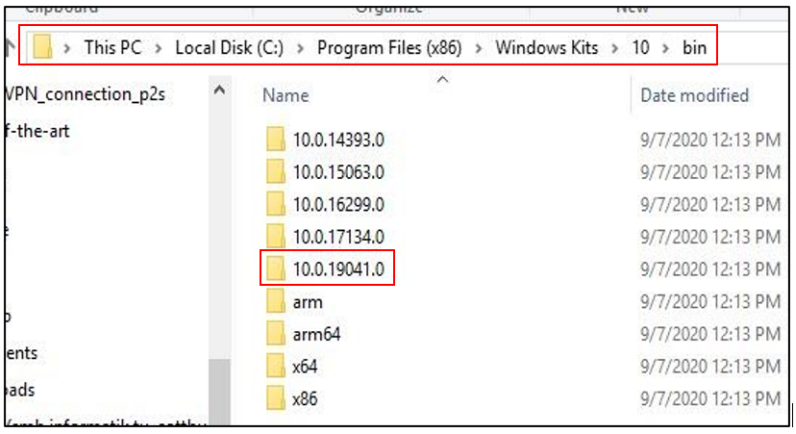


Figure 5.5: Installed the Windows 10 SDK (10.0.19041.0) for Windows 10

Export Certificates using MakeCert for Point-to-Site Connections

The **MakeCert** tool creates an *X.509* certificate [48], signed by the test root key or other specified keys, that binds our name to the public part of the key pair. To verify a user, computer, or service with public key infrastructure (PKI) that contained a public key, it is required to generate the *X.509* certificate, which is a worldwide accepted digital certificate [50]. The certificate is saved to a file. The MakeCert tool is installed in the \Bin folder of the SDK installation path. The *MakeCert* tool uses the following command syntax:

MakeCert [BasicOptions|ExtendedOptions] OutputFile

where $OutputFile$ is the name of the file where the certificate will be written and $BasicOptions$ are those options that are most commonly used to create a certificate. The options are given in the following Table 5.1 [48].

Table 5.1: Basic options specific to certificate store technology only

Basic option	Description
-a Algorithm	Hash algorithm. Must be set to either SHA-1 or MD5 (default).
-m nMonths	Duration of the validity period.
-n "Name"	Name for the publisher's certificate. This name must conform to the X.500 standard. The simplest method is to use the "CN=MyName" format. For example: -n "CN=Test".
-pe	Marks the private key as exportable.
-r	Creates a self-signed certificate.
-sky SubjectKeySpec	Subject's key specification. SubjectKeySpec must be one of *three* possible values: • Signature (AT_SIGNATURE key specification) • Exchange (AT_KEYEXCHANGE key specification) • An integer, such as 3
-ss SubjectCertStoreName	Name of the subject's certificate store where the generated certificate will be stored.

Azure uses certificates to authenticate clients connecting to a VNet over a P2S VPN connection. It is uploaded the *public key* information of the certificate to Azure when it is obtained a root certificate. This *root certificate* is considered as **trusted** by Azure when it is connected over P2S to the virtual network. Further, it is also generated *client certificates* from the trusted root certificate and then installed the client certificate on the client computer. That is, it will use to authenticate the client when a client initiates a connection to the VNet.

Generate a Self-signed Root Certificate

The root certificate generation can be divided into two parts: 1. Create a self-signed root certificate, and 2. Export the public key [45, 51].

1. **Create Self-signed root certificate**

   Here it is described the steps in the following articles how to generate a compatible self-signed root certificate with *MakeCert*:

   - After installation "Windows 10 SDK (10.0.19041.0)", it can be find the **makecert.exe** utility under the path (default):

     path: 'C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x86'

   - The Command Prompt(or Windows PowerShell) is opened as administrator and navigated to the location of the MakeCert utility. Is was used the following command for adjusting the proper location:

     > cd C:\Program Files (x86)\Windows Kits\10\bin\10.0.19041.0\x86

   - It is created with the command [45, 51] as mentioned below and is stored on my computer in above-mentioned path directory, as shown in Figure 5.6 and Figure 5.7. As in the following example, the created root certificate is in .cer file format that is called 'P2SRootCert.cer'.

     > makecert -sky exchange -r -n "CN=P2SRootCert" -pe -a sha256 -len 2048 -ss My "P2SRootCert.cer"

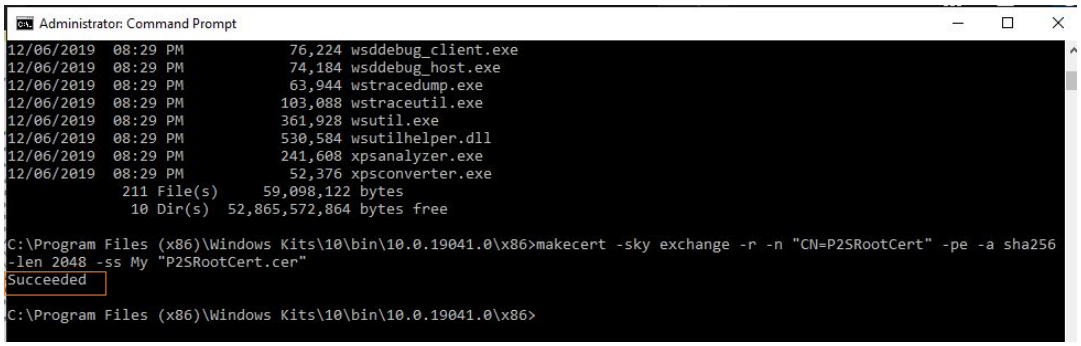


Figure 5.6: Command for creation of a root certificate

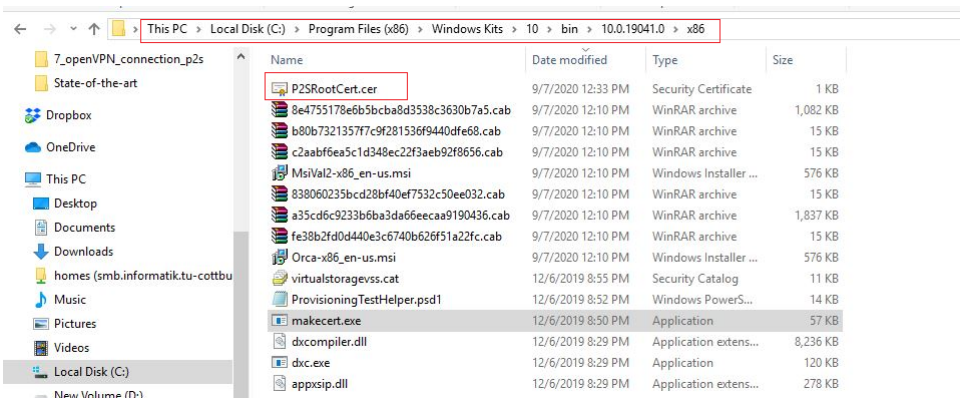


Figure 5.7: A root certificate creates a corresponding .cer file for configuring P2S

2. **Export the public key (.cer)**

   After creating a self-signed root certificate, it needs to export only **public key** (i.e., should not select the private key options for export) from P2SRootCert.cer file in *.cer file* format [45, 51].

   The following steps help to export the public key in a *.cer file* from afore-mentioned created self-signed root certificate:

   - To obtain a *.cer file* from the certificate, it was opened **Manage user certificates** application from the *start* menu. Again, it can also possible to open the **Manage user certificates** application with the help of Command Prompt or Windows PowerShell by typing *certmgr* command in the console window.
   - The self-signed root certificate was located in 'Certificates - Current User\Personal\ Certificates'. Further, the **Certificate Export Wizard** was opened with a right-mouse-click, then clicked the **All Tasks** option, and then again clicked the **Export** option, as shown in Figure 5.8.

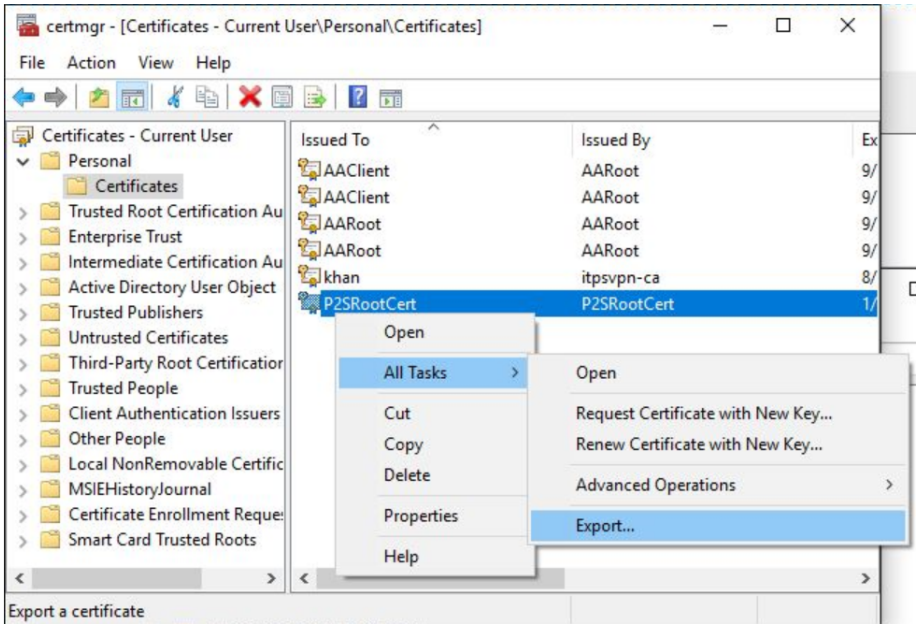
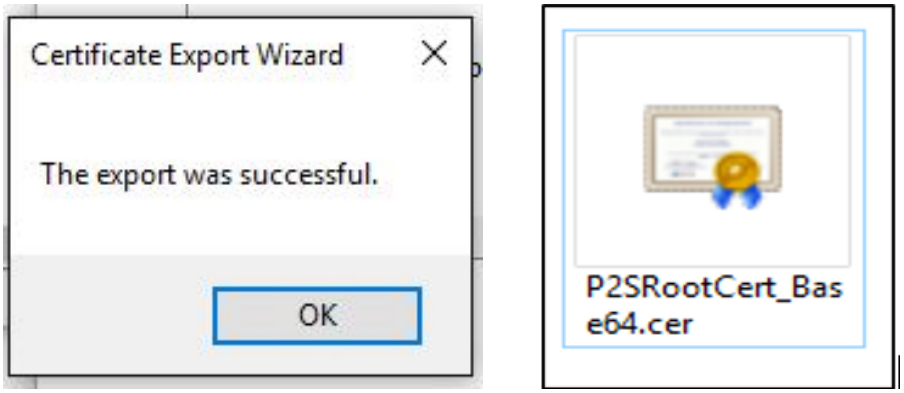


Figure 5.8: Manage user certificates application (certmgr)

   - Further, in **Certificate Export Wizard**, I have clicked **Next** button, then the **Export Private Key** page has appeared. Here, I have selected **No, do not export the private key** option, and then I have clicked on the **Next** button.
   - When the **Export File Format** page, I have selected **Base-64 encoded X.509 (.CER)**., and then it was clicked **Next**. It appeared for **File to Export** page. Then is was Browsed to the location to which was wanted to export the certificate. For **File name**, it was named the certificate file as *"P2SRootCert_Base64.cer"*. Then, it was clicked **Save** button and then clicked **Next** button.

- Finally, it was clicked on **Finish** button to export the certificate. Nonetheless, a **message box** with *"The export was successful"*,as shown in Figure 5.9(a), which was proof of our certificate was successfully exported. Furthermore, the exported certificate looks similar to this, as shown in Figure 5.9(b).



(a) Certificate is successfully exported (b) Outlook of exported certificate

Figure 5.9: Export the Public Key in *P2SRootCert_Base*64*.cer* file

In summary, it was used **MakeCert** to generate a self-signed root certificate. After it was created the *root certificate*, export the public certificate data (not the private key) as a *Base*64 *encoded X*.509 *.cer file*. Then, it was uploaded the *public certificate data* to the Azure server for the P2S VNet authentic connections [46].

Create and Install Client Certificates

It is not possible to install the root self-signed certificate directly on the client computer. So it is mandatory, to generate a client certificate from the self-signed certificate. Then only, a client certificate will export and install to the client computer.

The client certificate generation can be divided into three parts: 1. Create a client certificate from the self-signed root certificate, 2. Export both the private and public keys, and 3. Install the client certificatey [45, 51, 52].

1. **Generate a client certificate**

   A client certificate from a self-signed root certificate can be generated through the following steps. It may also possible to generate multiple client certificates from the same root certificate. That is, if we need to install a new client certificate for another client's computer, we can export the certificate from the same root certificate.

   a) The Command Prompt(or Windows PowerShell) is opened as administrator and navigated to the location of the MakeCert utility as described in subsection 1 (i.e., Create Self-signed Root Certificate).

b) It is created with the command [45, 51] as mentioned below and is stored on my computer in above-mentioned path directory, as shown in Figure 5.10. Further, it is changed the **'CN='** value that was specified when you created the self-signed root *"P2SRootCert"* to *"P2SChildCert"*. That is the name of the client certificate is modified when it is created from the self-signed root certificate. Therefore the following command was the result of a client certificate named **P2SChildcert** which would be stored in my **Manage user certificates** application, as shown in Figure 5.11.

> makecert.exe -n "CN=P2SChildCert" -pe -sky exchange -m 96 -ss My -in "P2SRootCert" -is my -a sha256

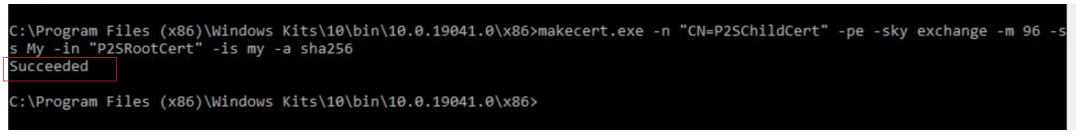


Figure 5.10: Command for creation of a child certificate

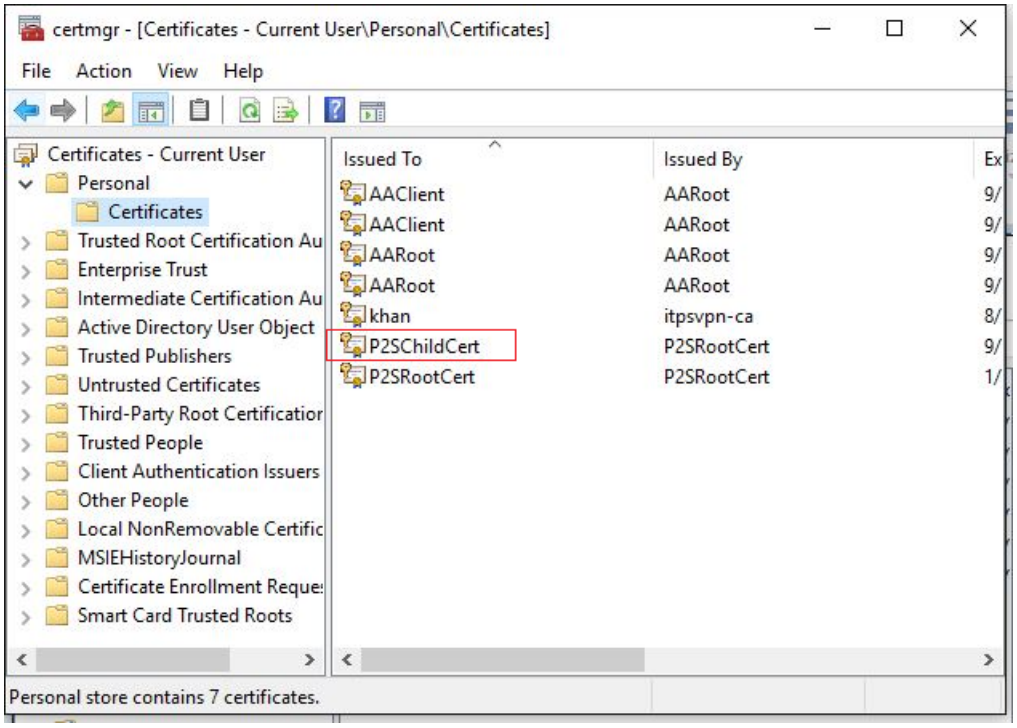


Figure 5.11: A child certificate in Manage user certificates application

2. **Export a client certificate**

After generating a client certificate, it is required to export the client certificate. The following steps help to export the private and public key in a .pfx file from the afore-mentioned created client certificate from the self-signed root certificate:

a) A client certificate is exported by opening **Manage user certificates** application and the default location was in 'Certificates - Current User\Personal\Certificates'. Then **Certificate Export Wizard** was opened with right-clicking on the client certificate and after that is was clicked on **all tasks**, and finally, clicked on the **Export** option. Further, **Next** button was clicked in the **Certificate Export Wizard** to continue. Again, in the **Export Private Key** page, *Yes, export the private key* was selected, and then clicked **Next** button.

b) Now, on the **Export File Format** page, it was left the defaults selected as well as made sure that *Include all certificates in the certification path if possible* is selected, as shown in Figure 5.12. This setting additionally would help to successfully export client certificates with required authentication information by the root certificate [51]. Without the afore-mentioned settings, it is clear that the client authentication would fail due to the absence of a trusted root certificate in the client.

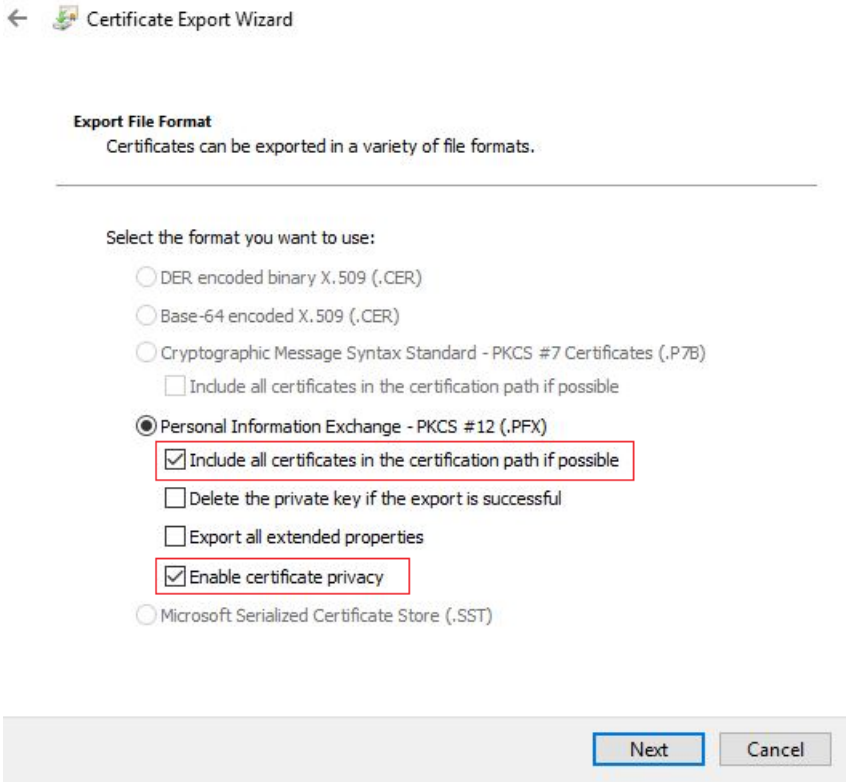


Figure 5.12: Selected options On the **Export File Format** page

c) The *private key* is protected on the **Security** page which must be protected. Here, a password e.g. *"admin123456+"* was used to make sure the protection of this certificate. On the **File to Export**, the client certificate was named as *"P2SChildCert.pfx"* and to export the certificate the location was selected with the

help of **Browse** button. By clicking the **Finish** button, the porcess of exporting the client certificate was completed.

3. **Install client certificates for P2S authenticate connections with windows 10** After generating the *.pfx* file on the client computer, it was installed on the client computer by double-clicking the *.pfx* file. It was kept the default **Store Location** as **Current User** [52]. Again, on the **File to Import** page, it wasn't made any changes. Further, on the **Private key protection** page, the correct password i.e, *"admin123456+"* was entered for verification of the certificate. Finally, all the other options were left as default and then selected with **Finish** button to complete the certificate installation process and imported the certificate successfully [52].

In summary, to connect to a VNet using Point-to-Site it is a must for each client computer should have a client certificate installed. It is generated a client certificate from the self-signed root certificate and then exported and installed on the client computer. Moreover, authentication will fail during the authentication of the P2S connection due to not install of the client certificate on the client computer.

5.2.2 Configure a Point-to-Site VPN Connection to a VNet

The following Architectures is used to configure the Point-to-Site native Azure certificate authentication connections use.

- To create a RouteBased VPN gateway.
- To upload the public key (.cer file) for a root certificate to Azure so it would be considered a trusted certificate and used for authentication.
- The generated and installed client certificate was used for client computer that would connect to the VNet for client authentication, see details in Section 5.2.1.
- Finally, to configure a VPN client configuration it was used the specific settings in a configuration file for the Windows operating system.

In this Section, it is discussed step by step how a configuration can be prepared for Point-to-Site VPN Connection to a VNet.

Create a Resource Group

To create a resource group in Azure, it is required to sign-up with a free account before beginning. Then, in **Search resources, service, and docs (G+/)**, it was typed *resource group* to search [53], as shown in Figure 5.13. Further, it was selected the **Subscription** as *Azure for Students* and **Region** was *"(US) East US"*. It was entered in the **Resource group** name as *syncrg*, which must be a unique name for my new resource group [54]. Further after filling up all the required information on the resources group, it was selected **Review + Create to** -> **Create** button to deploy a new resource group that had taken a few seconds to create. Furthermore, once it was created, it could be seen as a resource group on the Azure portal dashboard [54, 55].

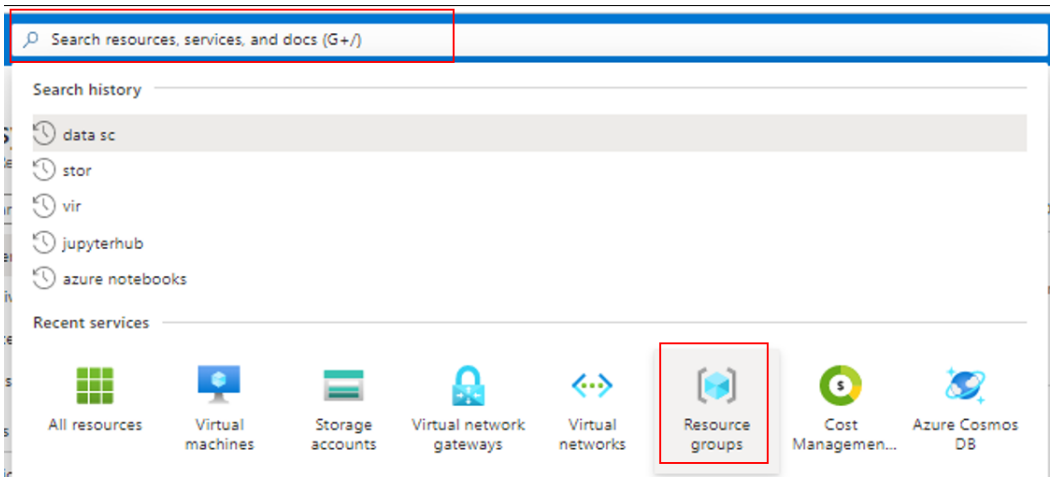


Figure 5.13: Microsoft Azure Portal Dashboard

Create a Virtual Network

It was created a VNet with the help of Azure Resource Manager (ARM) templates model and the Azure portal by following steps:

- It was typed *virtual network* in **Search resources, service, and docs (G+/)** and then *Virtual Network* was selected from the **Marketplace** results [46]. Again, on the **Virtual Network** page, once it was selected **Create** button, the **Create virtual network** page opens.
- *Project details* and *Instance details* were configured on the **Basics** tab for VNet settings. It was entered with specific content in the fields such as **Subscription**, **Resource group** as *syncrg*, entered the name for my **virtual network** Name as *syncvnet*.
- On the **IP Addresses** tab, the items were adjusted according to the settings of my requirement such as **IPv4 address space** (by default, an address space was automatically created) [46], **Subnet** also a default subnet is created automatically [46], one additional **Subnet name** was called *storagesubnet* and **Subnet address range** was 10.1.1.0/24, another **Subnet name** was called *GateWaySubnet* and **Subnet address range** was 10.1.2.0/24, as shown in Figure 5.14.
- Now to deploy the virtual network, it was selected **Review + create** to validate the virtual network settings, and once validation had been done, it was selected **Create** button, which finally deployed the virtual network, as shown in Figure 5.15.

Create a Virtual Network Gateway

In this Section, a deployment of **virtual network gateway** was described for my VNet. The **virtual network gateway** could be created with the ARM deployment model and the Azure portal by following steps:

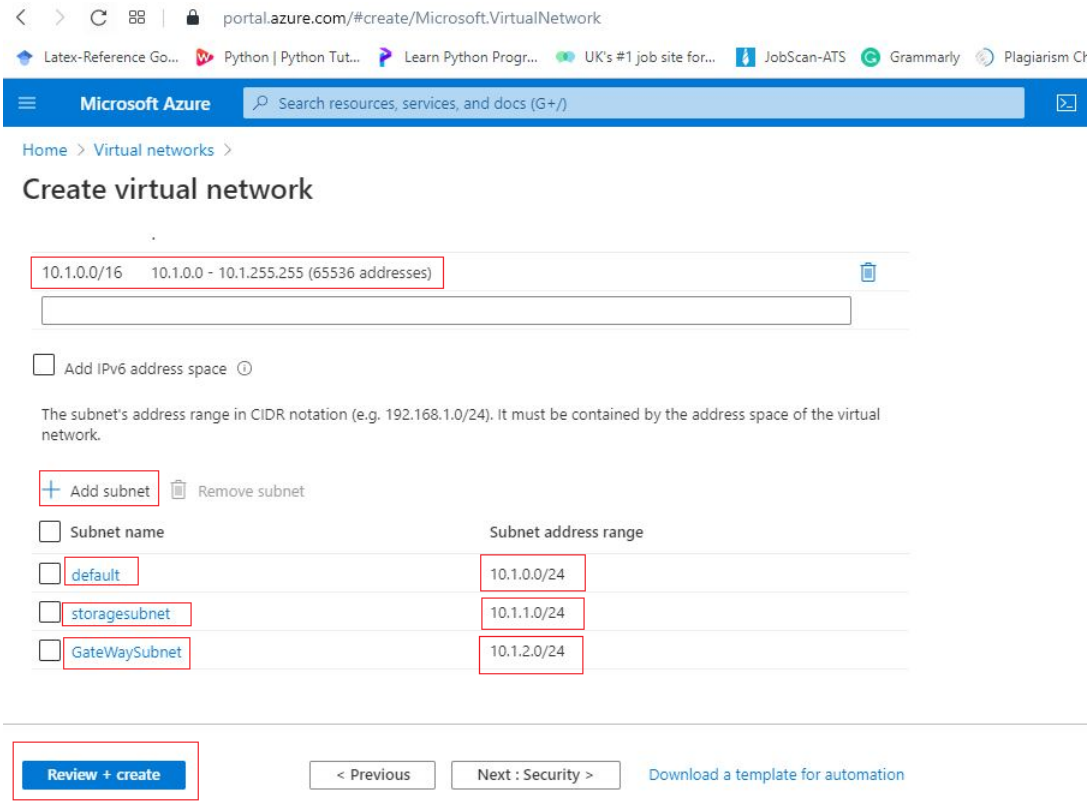


Figure 5.14: To add the values for configure of Subnet Name and Subnet address range

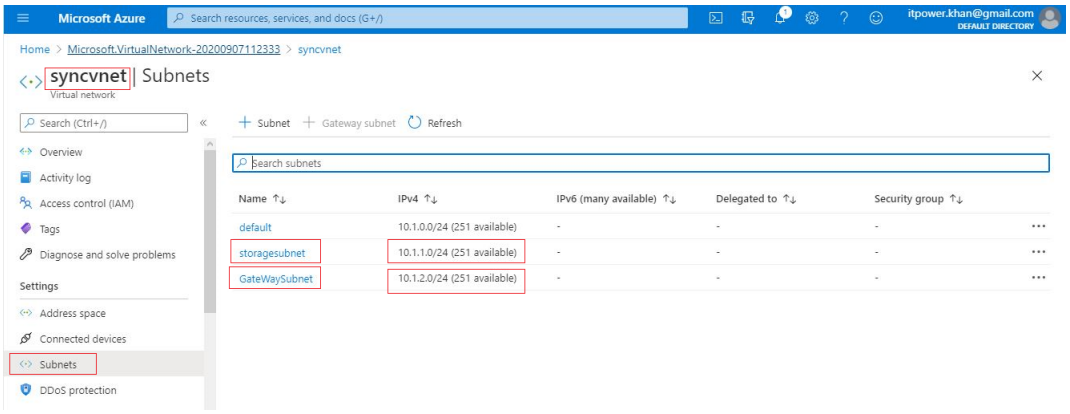


Figure 5.15: Deployed Two Virtual network

- First, from the *Azure portal menu*, **Create a resource** was selected, and in the **Search the Marketplace** field, *Virtual Network Gateway* was typed for locating *Virtual network gateway* from the search returned. After selecting the entry the **Virtual network gateway** page was opened, where **Create** button was clicked to open the **Create virtual network gateway** page.
- Then, on the **Basics** tab, for *Project details* and *Instance details* was filled with specific values such as **Subscription**, **Resource Group**, **Name** of my gateway was called *sync-vng*, **Region**, **Gateway type** was selected *VPN*, **VPN type** was selected *Route-based* [46], **SKU** was select from the dropdown list as *VpnGw1* (here, it should be noted that the SKUs was selected on the **VPN type** for tunneling, connection, and throughput which support *P2S IKEv2/OpenVPN Connections*) [56], **Virtual network** was selected from the dropdown called *syncvnet* [46], as shown in Figure 5.16.

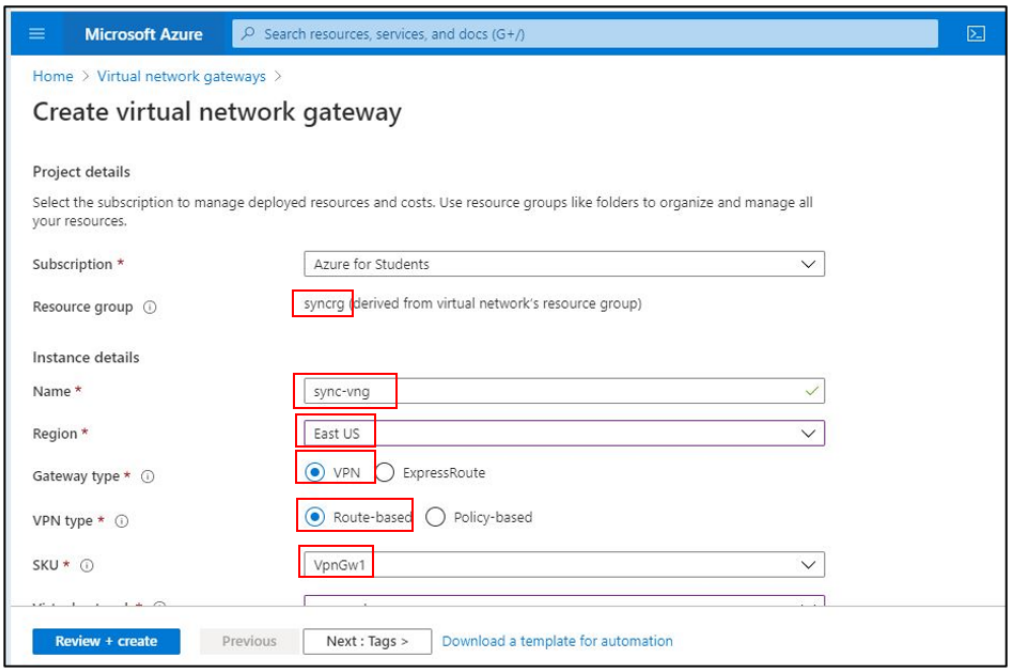


Figure 5.16: Configuration for **Basics** tab of *Project details* and *Instance details*

- For the setting of the **Public IP address** object that gets associated to the *VPN gateway* which was dynamically assigned to this object when the VPN gateway was created. Further, it was leveled and selected few fields such as the leave for **Public IP address** was selected *Create new*, a name called *vng-ip* was typed for **Public IP address name**, **Assignment** was supported only *Dynamic* by default, and **Enable active-active mode** was left for this setting as *disabled*, as shown in Figure Åfter this setting, it was clicked **Review + create** a button for running validation and once this validation had passed, the **Create** button was selected to deploy the VPN gateway.

In summary, it could be often taken *45 minutes or more*, depending on the selected gateway **SKU** for creating a gateway. Here, the *Basic* gateway SKU did not support *IKEv2 or*

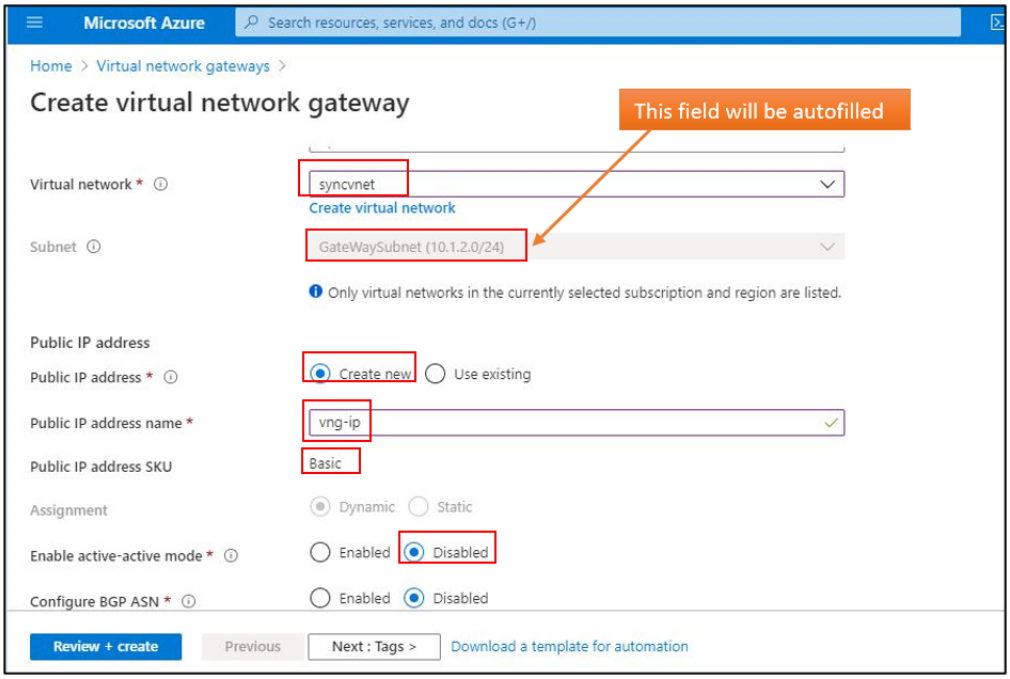


Figure 5.17: The specifies setting associated with the **public IP address** for VPN gateway

RADIUS authentication [56], that is why **SKU** was selected *VpnGw1* [46, 56] for the virtual network gateway when configuring my virtual network. After the gateway was created, the IP address was viewed, as shown in Figure 5.18 that had been assigned to it by looking at the virtual network in the Azure portal and the gateway had appeared as a connected device.

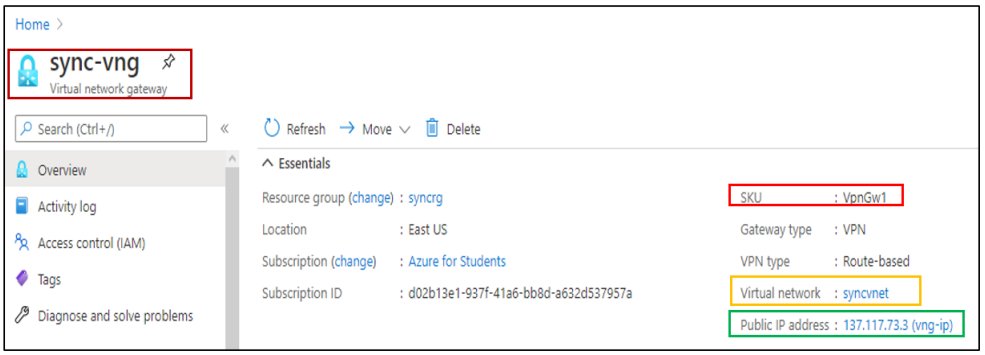


Figure 5.18: The IP address is created for the the VPN gateway

Create Point-to-site Configuration

In the previous Section, my virtual network gateway had been created. In this Section, the Settings for *P2S configuration* would be described. The steps for configuring the P2S *VPN client* configuration as follows: first, the **User VPN configuration** option was selected,

and then selected the **Configure now** to open the configuration page. To configure and *download a VPN client* for the Windows 10 machine, it is needed several steps which are mentioned as following:

Client Address Pool

The client address pool is a range of private IP addresses that we have to specify. When the clients will be connected over a Point-to-Site VPN, an IP address will dynamically receive from this range. It must be taken care to use a private IP address range that will not overlap with the on-premises location that I connect from, or the VNet that I want to connect to [45, 46].

In the **Address pool** box, it was added the private IP address range 172.16.0.0/24, as shown in Figure 5.19 which would use for *VPN clients* to dynamically receive an IP address from the range that was specified.

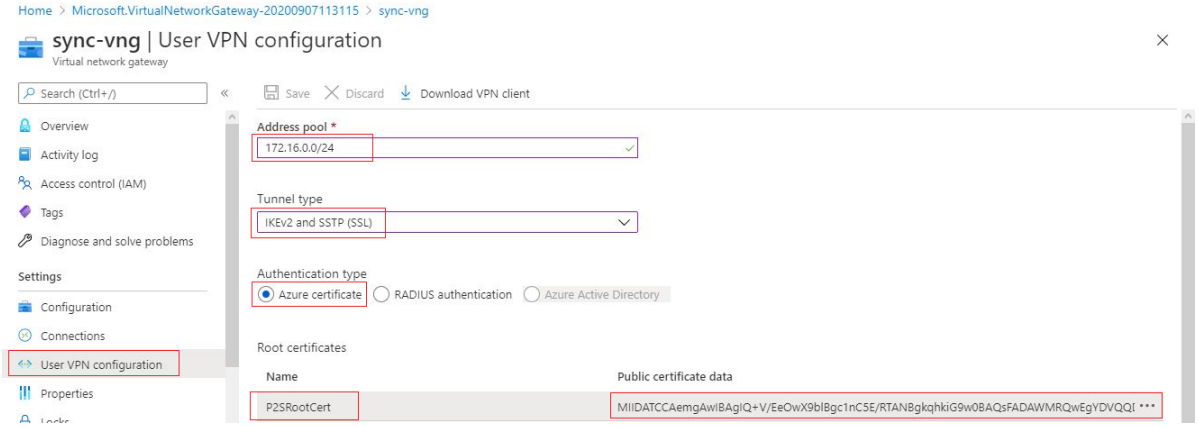


Figure 5.19: Configure Point-to-Site Connection through *VPN client*

Tunnel Types

In this Section, it was configured tunnel type. On the Point-to-site configuration page, it was selected **Tunnel type** as *IKEv2 and SSTP (SSL)*, as shown in Figure 5.19. However, it should be noted here that if anyone would use the *Basic* value during setting the parameter called **SKU** for *VPN gateway configuration* [46], then he/she will not see **Tunnel type** or Authentication type on this page. That is the reason, it was selected the options for **SKU** level as *VpnGw1*.

Authentication Types

In this Section, it was configured the authentication type. For **Authentication type**, *Azure certificate* was selected from the various option, as shown in Figure 5.19 [46].

Root Certificate Data

In this Section, the *public root certificate* data was uploaded to Azure, which was already created and exported the root certificate as a Base-64 encoded X.509 (.cer) file called *"P2SRootCert_Base64.cer"* (See details in the Section 1 i.e., Generate a Self-signed Root Certificate).

Further, the file had opened the certificate with a *text editor*, such as Notepad [46], as shown in Figure 5.20 and Figure 5.21. When the certificate data was copied from *"P2SRootCert_Base64-.cer"* file, it had made sure that the copied text was one *continuous line without carriage returns* or line feeds [46]. Then, the certificate data was pasted into the **Public Certificate Data** field and the **Name** field was typed *P2SRootCert* as name for the **Root certificates**, as shown in Figure 5.19, and then selected **Save** button.

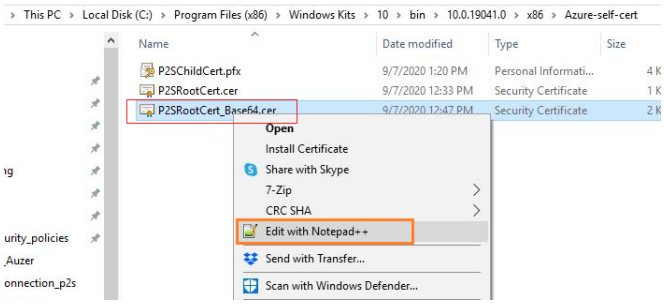


Figure 5.20: The "P2SRootCert_Base64.cer" file from self-signed root certificate

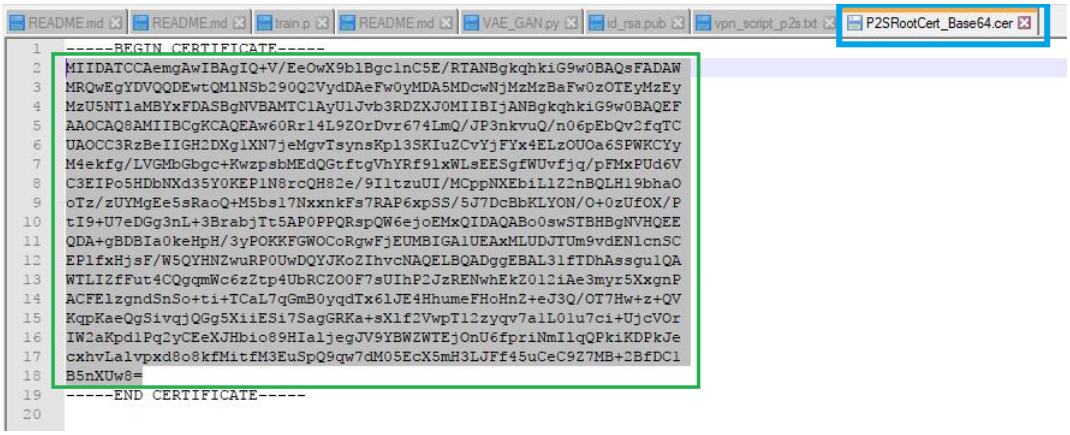


Figure 5.21: The "P2SRootCert_Base64.cer" file opened in a Notepad

Therefore, once the public certificate data was uploaded, Azure can use it to authenticate clients.

In summary, after clicking the **Save** button at the top of the page was saved all of the configuration settings [46]. Further, with selecting the **Download VPN client** button at the top of the page of the configuration settings was downloaded a .zip file(*sync-vng.zip*) in

Windows 10 client computer. Then the *sync-vng.zip* was unzipped and three numbers of files was revealed in the *sync-vng* folder, as shown in Figure 5.22 (within blue rectangle).

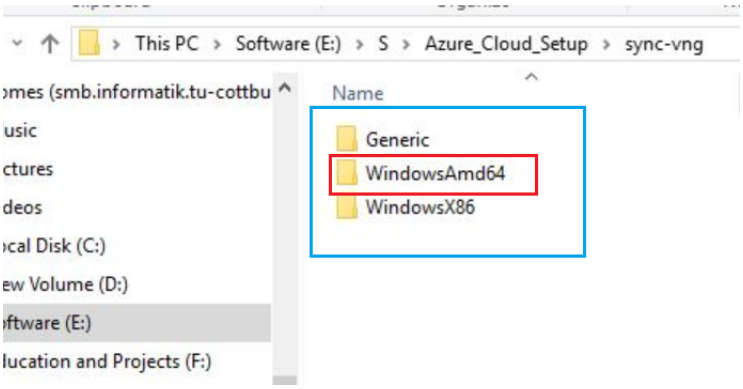


Figure 5.22: Downloaded VPN client files from Azure Portal

Client Certificate

A client certificate was generated from the trusted root certificate (see details in Section 3). To create a P2S connection from a client computer, it should be generated and then export a client certificate. When installing a client certificate from (.pfx) file, it is necessitated to put the same password, i.e. *"admin123456+"*, that was formulated when the client certificate was exported. Otherwise, the root certificate information will not present on the client computer and the *Windows 10 client* won't be able to authenticate properly [46].

Generate and install the VPN client configuration package

In this Section, the VPN clients must be configured [57] and it is installed the VPN client configuration files in *WindowsAmd*64 folder, as shown in Figure 5.22 in order to connect to a VNet over a P2S connection, see in previous Section for the details discussion.

To configure the native Windows 10 VPN client, It was used the following steps for certificate authentication:

- First, the VPN client configuration file was selected to correspond to the architecture of the Windows 10 computer. For example, for a 64-*bit processor* architecture, it had to choose the *'VpnClientSetupAmd*64*.exe'* installer package from the *WindowsAmd*64 folder. Further, the installer package had selected and installed as an administrator on *Windows* 10 *computer*.
- On the client computer, the VPN connection could be viewed by navigating to Network Settings and clicked VPN, as shown in Figure 5.23. The VPN connection had shown the name of the virtual network that it was needed to connect through a P2S connection.

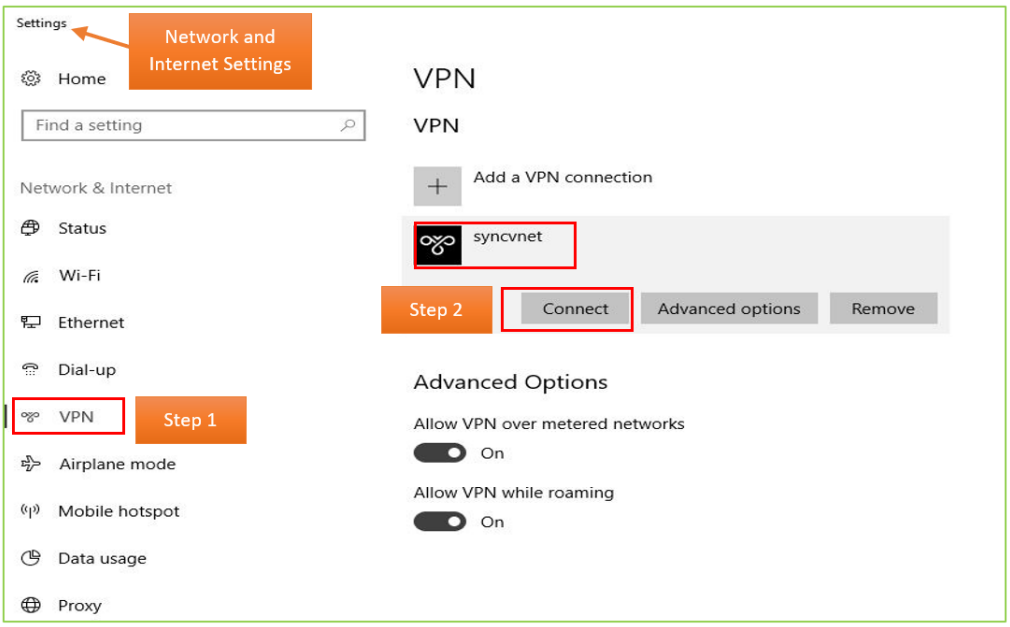


Figure 5.23: Configuration of the native Windows 10 VPN client

- On the Connection status page, when the Connect button was selected to start the connection, as shown in Figure 5.24 and the connection was established, as shown in Figure 5.25.

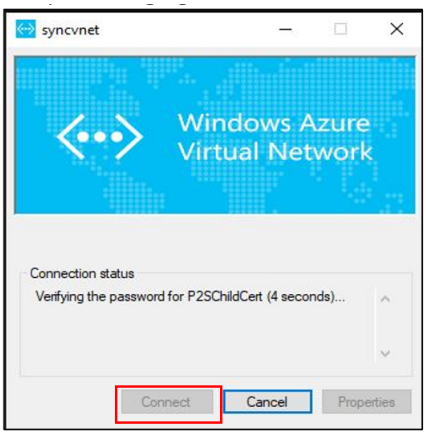


Figure 5.24: Connection status page for Azure VPN

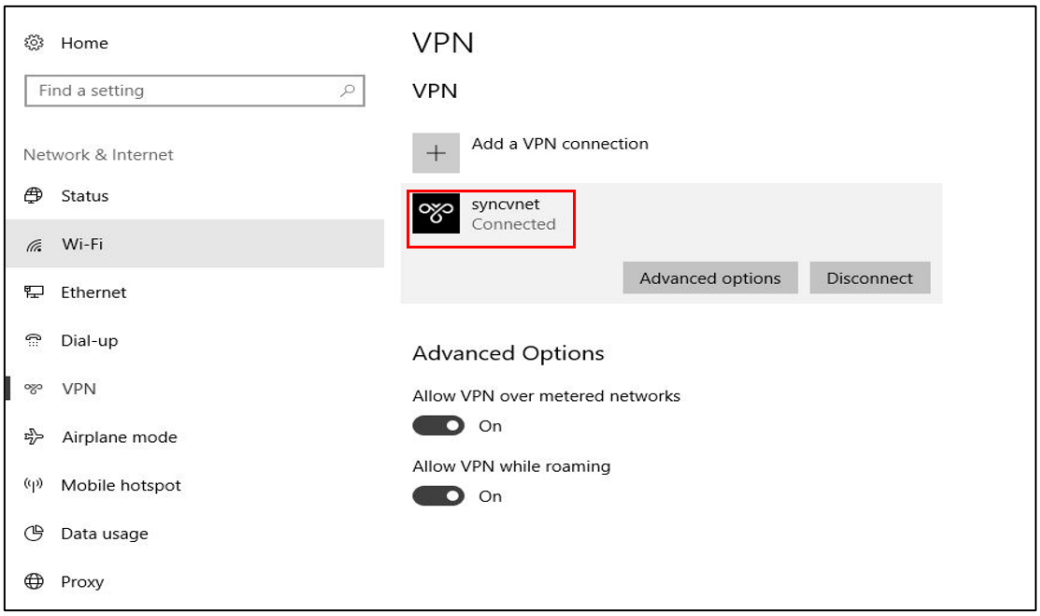


Figure 5.25: Verification of my established connection.

5.2.3 Use an Azure File Share with Windows 10

In this Section, it is described, what are the prerequisites and how to create an Azure file share by requiring three fundamental things about how it will be used: (i) for the performance requirements of Azure Files share offers standard file shares, which are hosted on hard disk-based (HDD-based) hardware, (ii) the size of the Standard file shares can span up to 100 TiB, and (iii) for the redundancy requirements of an Azure Standard file shares offer locally-redundant storage (LRS) [58].

For using an Azure file share with Windows 10 computer the steps are describing as under:

1. **Create a Storage Account**

   First, From the dashboard + **Create a resource** was selected, which resulting in the Azure Marketplace search window page and it was typed in the search field *storage account* and pressed **enter** button. This would lead to an overview page for *storage accounts*. Further, the **Create** button was selected to proceed with the *Create storage account* wizard.

   Further, in the **Basics** Section, a general purpose v2 (GPv2) storage account was created by selecting from the **Performance** radio button is set to Standard, and the Account kind was selected to *StorageV2 (general purpose v2)* from the drop-down list. Now, the other basics fields were independently chosen for the storage account, such as **Subscription**, **Resource group** as called *syncrg*, **Storage account name** was created as named *itpowerstorage* (this name must be globally unique), **Replication** was labeled as *locally-redundancy (LRS)*, and **Blob access tier** was chosen *Hot* from one of the radio buttons [58].

Finally, the **Review + create** button followed by the **Create** button was selected to create the storage account, as shown in Figure 5.26.

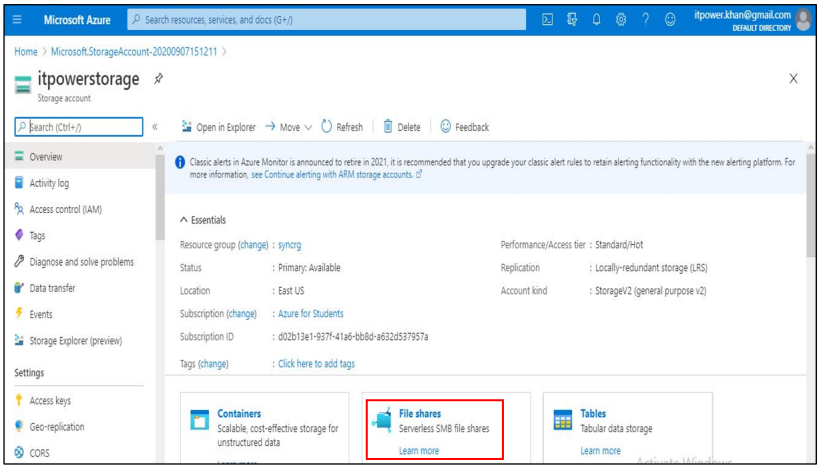


Figure 5.26: Create a storage account is called *itpowerstorage*

2. **Create a File Share**

Once the storage account was created, all that was left, to create a *file share*. The following steps should be considered to create a file share.

As shown in Figure 5.26 with *red square mark* on the storage account page, the **File shares** label was selected to navigate to the File shares page in the Azure portal. In the file share listing, it should be seen as an *empty table* list because no file shares had been created yet. Now, **+ File share** tab was selected to create a new file share with entering few empty fields, such as the **Name** of the file share to be created called *itpowerfileshare*, the **Quota** size of the file share was provisioned 20*(GiB)*, and the **Tiers** for a file share was selected as *Hot*. Then, a **Create** button was selected to finish creating the new share in the list of the table called *itpowerfileshare*, as shown in Figure 5.27.

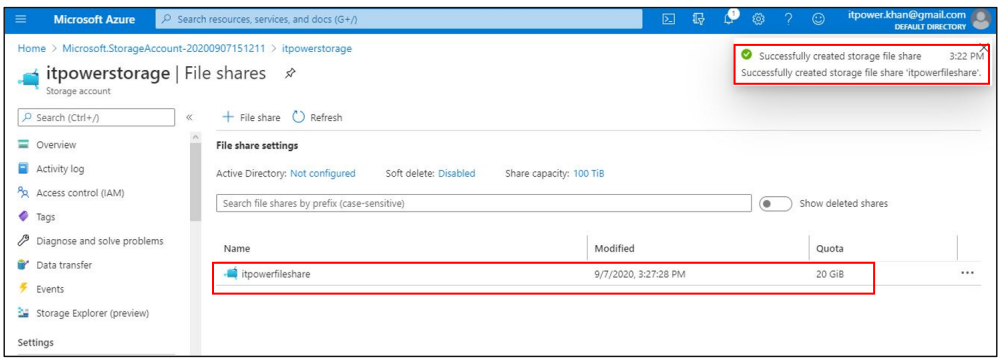


Figure 5.27: Create a storage account is called *itpowerstorage*

3. **Mount the Azure File Share on Windows** 10 **Computer** The steps to map a storage file share with localhost i.e., Windows 10 machine is as follows:

   - **Mount the Azure File Share**

     First, *itpowerfileshare*, as shown in Figure 5.27, was clicked to open the *itpowerfileshare page*, and then it was clicked on Connect tab. It would provide the Windows CLI command, as shown Figure 5.28. The Windows CLI command (see details command in A) was copied and pasted in a text editor. Because, the user and password would need for future configuration, as as shown Figure 5.28 marked with a yellow color border.

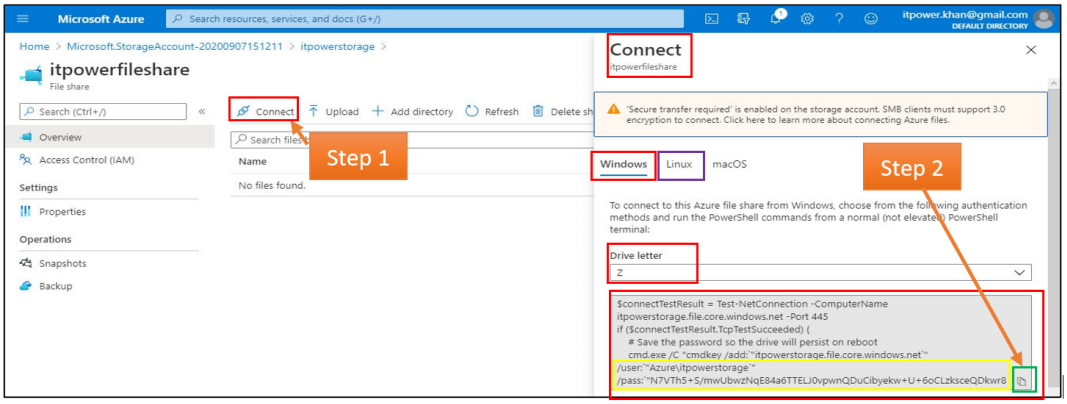


Figure 5.28: Windows CLI command to mount the Azure file share on Windows 10 PC

To connect with the Azure file share from Windows, it was chosen the following authentication methods [59].

/user:‘"Azure\itpowerstorage‘"

/pass:‘"N7VTh5+S/mwUbwzNqE84a6TTELJ0vpwnQDuCibyekw+U+6oCLzksceQDkwr8q+W3y1YRWraSQktaANBV6gOaaA==‘""

Further, it can be run into the PowerShell commands (see details command in A) from a normal PowerShell terminal. However, for this project, the Azure file share called *itpowerfileshare*, had been mounted with the Windows File Explorer.

   - **Mount the Azure File Share with File Explorer**

     The **File Explorer** was opened by pressing *Win+E* shortcut key and then navigated to **This PC** on the left-hand side of the window. This would change the menus available in the ribbon and under the **Network** menu, the **Map network drive** was selected [59]. Further, in the **Map network drive** form the *drive letter (Z:)* was selected and entered the uniform resource locator (URL) path, which copied from the Azure file share (*itpowerfileshare*) **Properties**. Again, the URL path format was changed as per universal naming convention (UNC) [60] such as \\<storageAccountName>.file.core.windows.net\<fileShareName>. For this project, the

UNC path was for example: "\\itpowerstorage.file.core.windows.net\itpowerfileshare", as shown in Figure 5.29.

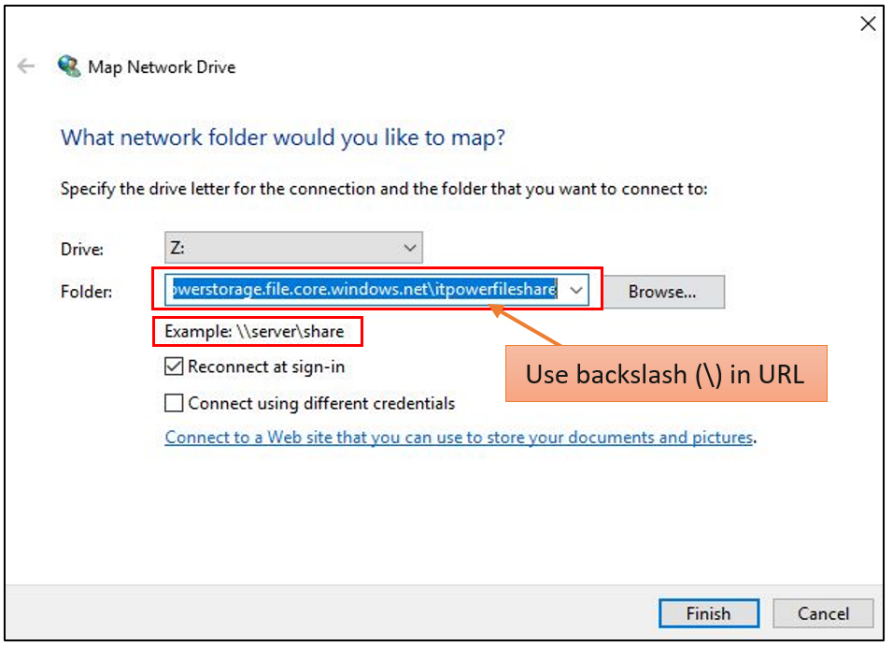


Figure 5.29: Azure File Share mapped for network Drive (Z:)

- To verify, when it was clicked the Finished button, another Map network drive dialogue box popped up, as shown in Figure 5.30. This means the connection was established with the Azure file storage system.

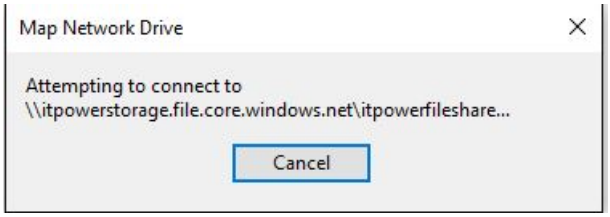


Figure 5.30: Verification of the connection establishment

4. **Use Private Endpoints for Securely Connect to Azure Storage**

   Azure Private Endpoint is a network interface that connects us privately and securely to an Azure service, such as Azure Storage, Azure Cosmos DB, SQL over an *Azure Private Link* [61, 62]. **Azure Private Link** enables us to access Azure PaaS Services, for example, Azure Storage and SQL Database [63]. **Private Endpoint** uses a private IP address from our VNet i.e., *syncvnet*, and effectively bringing the service, i.e., *itpowerstorage* into our VNet.

For eliminating exposure from the public internet network traffics between the clients such as Microsoft's DSVM on the VNet and the storage account, i.e., *itpowerstorage* traverses over the VNet and a private link on the Microsoft backbone network. Using *private endpoints* for storage account enables to: (i) secure storage account to block all connections for the public endpoint [64], (ii) increase security by blocking the exfiltration of data from the VNet [64], and (iii) connect on-premises networks securely with storage accounts and the VNet using VPN [64], as shown in Figure 5.31.

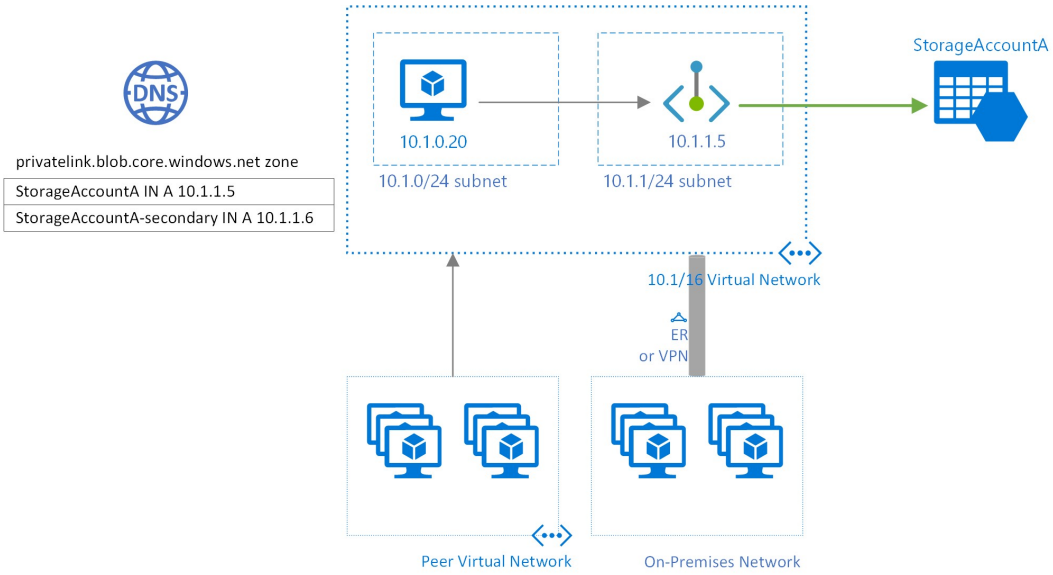


Figure 5.31: Conceptual overview of private endpoints connects storage service [64]

In this Section, a private endpoint was created for storage account service which can be securely accessed using VPN gateway from on-premises networks. The steps are as follows:

- **Create a Private Endpoint**

  First, in the storage account of Azure portal, the **Private endpoint connections** [65] was found and clicked on + **Private endpoint** tab which would be redirected to the **Basics** tab of *Create a private endpoint* page. It was entered this following information for *Project details* and *Instance details* such as, **Subscription**, selected the **Resource group** as *syncrg*, entered the **Name** as *syncpvtendpoint*, and seleted **Region** as *East US* [65], as shown in Figure 5.32.

  Now, in **Resource** Tab, the following settings had been changed such as the **Connection method** selected as *Connect to an Azure resource in my directory*, The **Resource type** selected as *Microsoft.Storage/storageAccounts*, the **Resource** selected *itpowerstorage* and the **Target sub-resource** selected as *file* [65], as shown in Figure 5.33.

  Further, in the **Configuration** Tab, the following settings of *Networking* and *Private DNS integration* had been changed such as, the **Virtual network** selected

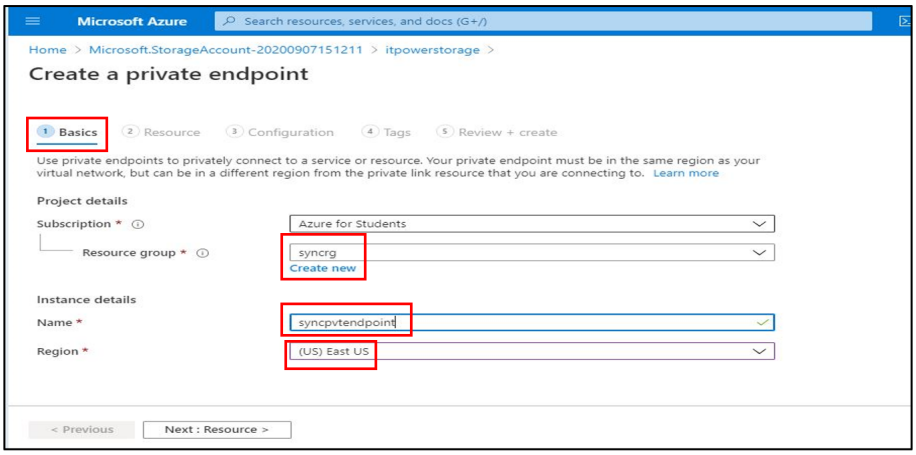


Figure 5.32: **Basic** Tab information in Create a Private Endpoint page.

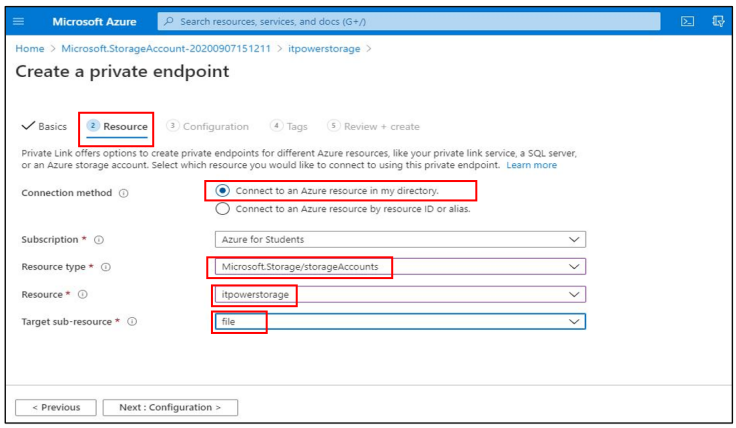


Figure 5.33: **Resource** Tab information in Create a Private Endpoint page.

as *syncvnet*, the **Subnet** selected as *storagesubnet* (10.1.1.0/24), the **Integrate with private DNS zone** left with the default value of *Yes* and the **Private DNS zones** also left with the default of *(New) privatelink.azurewebsites.net* [65], as shown in Figure 5.34.

Finally, **Review + create** button was clicked, and hit the **Create** button after waiting for *validation passed*. Further, the Private endpoint i.e., *syncpvtendpoint* deployed with a *private IP address* e.g., 10.1.1.4 for the VPN network, as shown in Figure 5.35.

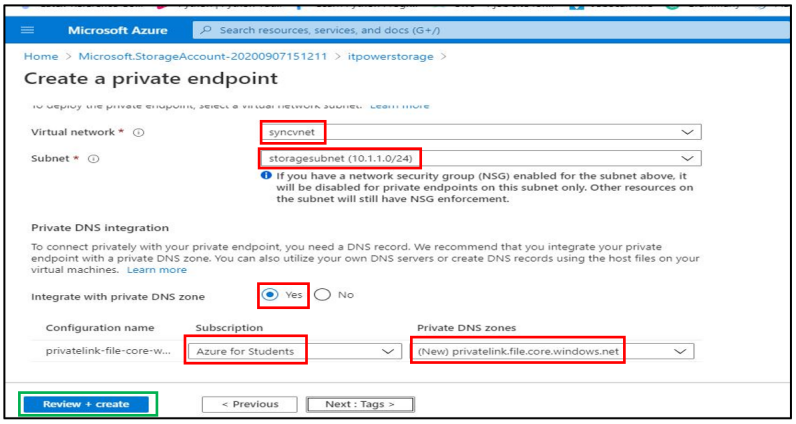


Figure 5.34: **Configuration** Tab information in Create a Private Endpoint page.

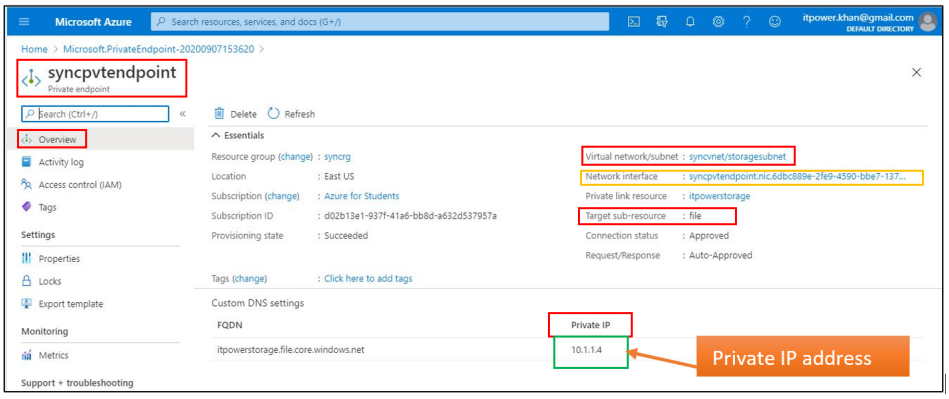


Figure 5.35: Deploy the Private Endpoint with a private IP address.

- **Test Connectivity to Private Endpoint**

  As detailed discussed in previous Section 3 (i.e. **Mount the Azure File Share with File Explorer**) and shown in Figure 5.35 and Figure 5.36, the *Private IP* address was copied and pasted as "\\10.1.1.4\itpowerfileshare" instead of "\\itpowerstorage.file.core.windows.net\itpowerfileshare" into the **Map Network Drive** [65, 66]. The **Finish** button was clicked to go to **Windows Security** form. Further, the network credential (i.e. *User name* and *Password*) was given as input (the details credential can be found in Appendix A), as shown in Figure 5.37 and Figure 5.38. Finally, the Share file of the Storage account will appear on Windows 10 PC, as shown in Figure 5.39.

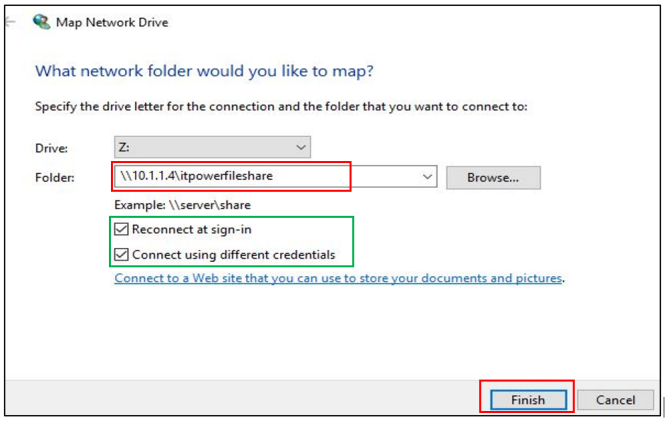


Figure 5.36: Private IP address for P2S connection

```
$connectTestResult = Test-NetConnection -ComputerName
itpowerstorage.file.core.windows.net -Port 445
if ($connectTestResult.TcpTestSucceeded) {
    # Save the password so the drive will persist on reboot
    cmd.exe /C "cmdkey /add:`"itpowerstorage.file.core.windows.net`"
/user:"Azure\itpowerstorage"
/pass:"N7VTh5+S/mwUbwzNqE84a6TTELJ0vpwnQDuCibyekw+U+6oCLzksceQDkwr8q+W3y1YRWraSQktaA
NBV6gOaaA==`""
    # Mount the drive
    New-PSDrive -Name Z -PSProvider FileSystem -Root
"\\itpowerstorage.file.core.windows.net\itpowerfileshare" -Persist
} else {
    Write-Error -Message "Unable to reach the Azure storage account via port 445.
Check to make sure your organization or ISP is not blocking port 445, or use Azure
P2S VPN, Azure S2S VPN, or Express Route to tunnel SMB traffic over a different
port."
}
```

Don't need to copy the quotation mark

Figure 5.37: Network credential (i.e. *User name* and *Password*).

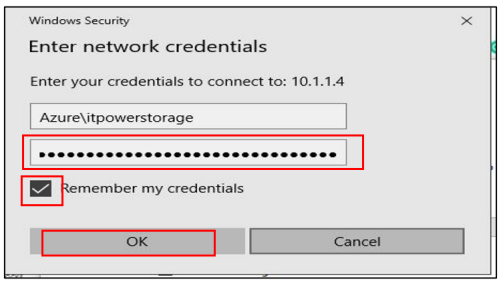


Figure 5.38: Windows Security form with User name and Password.

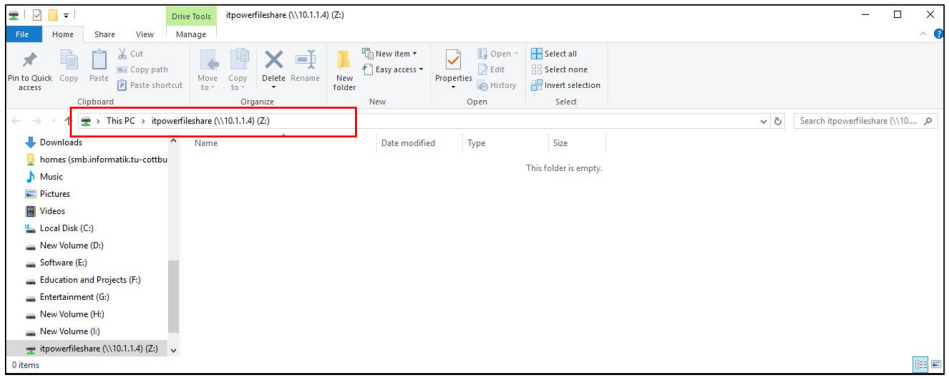


Figure 5.39: Azure share file of the storage account on Windows 10 PC.

In summary, when it is created a private endpoint for the storage account, i.e., *itpowerstorage* then the private endpoint would provide secure connectivity between clients on VNet (e.g., *syncvnet*) and the afore-mentioned storage account. Further, the private endpoint was assigned an IP address from the IP address range of the VNet. Again, a secure private link uses for the connection between the private endpoint and the storage service [64].

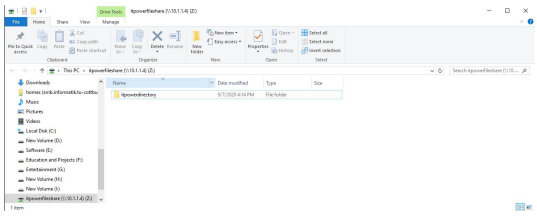
Therefore, the new approach to connect from Windows 10 PC to Azure storage service through P2S communication comes to an end. In the next Section, the evaluation and testing of the afore-mentioned approach have been described in detail.

6 Evaluation of the Implemented Solution

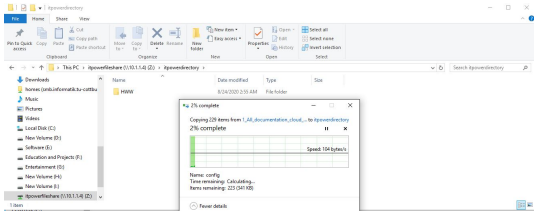
In this Chapter, it is described how could be tested the implementation (details in Chapter 5) aiming to solve the problem(s) statement presented in Chapter 3. Further, it is also discussed the feedback of supervisors of the company.

6.1 File Upload and Synchronization

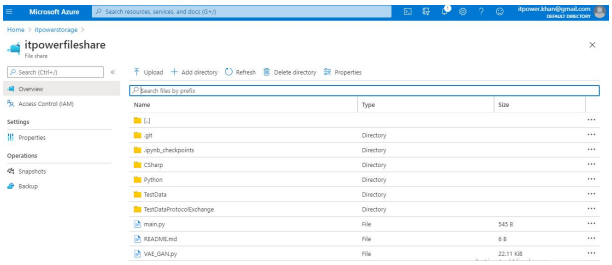
It had been found that when the ML algorithm files are uploaded to the Azure file share, they would instantly store in the Azure cloud environment. One subfolder(i.e., HWW) had been created in the parent folder (i.e., itpowerdirectory), as shown in Figure 6.1(a) and 6.1(b) and uploaded the ML algorithm python extension files into the HWW subfolder, as shown in Figure 6.1(c).



(a) The "itpowerdirectory" folder created



(b) The "HWW" folder copied to Azure storage file



(c) The ML files uploaded to Azure storage file

Figure 6.1: The ML File Upload and Synchronization with Azure storage file

6.2 Create Microsoft's DSVM for Deep Learning

Creating a DSVM instance (VM) was simple but must have remembered for the specific paraments during the creation of the graphics processing unit (GPU) instances (VM). This Section will be a quick-start guide to launching one DSVM instance.

First, *"Data Science Virtual Machine for Linux"* was searched in Microsoft Azure **"Create Resource"** search box. Then, for the basic settings, it was configured few parameters such as **"Resource Group"** was used the existing "*syncrg*", **Region** was selected *(US) East US*, and the **Size** of the instance was selected as *Standard_NC6* (the number of vCPUs is 6 with K80 GPU machine and 56 Gigabyte (GB) Random-access memory (RAM)), as shown in Figure 6.2 [7].

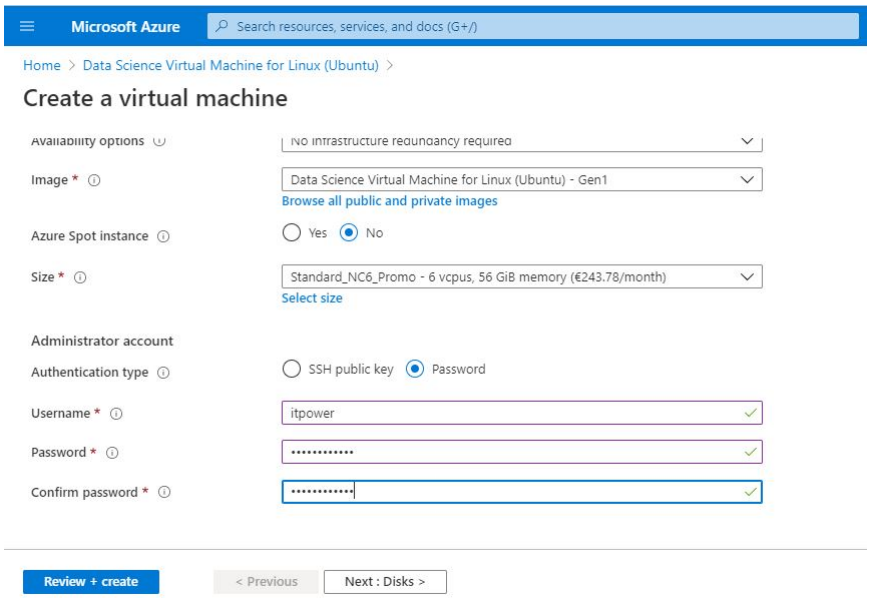


Figure 6.2: Microsoft Azure resource "Basic Settings".

Further, It was elected to use a simple *Password* rather than a SSH public key file for **Authentication type**. Again, The **OS disk type** was selected as *Standard HDD* in the **Disk options** (i.e. do not select SSD). Finally, on the **Networking** page, the **Subnet** was selected *storagesubnet*(10.1.1.0/24), as shown in Figure 6.3.

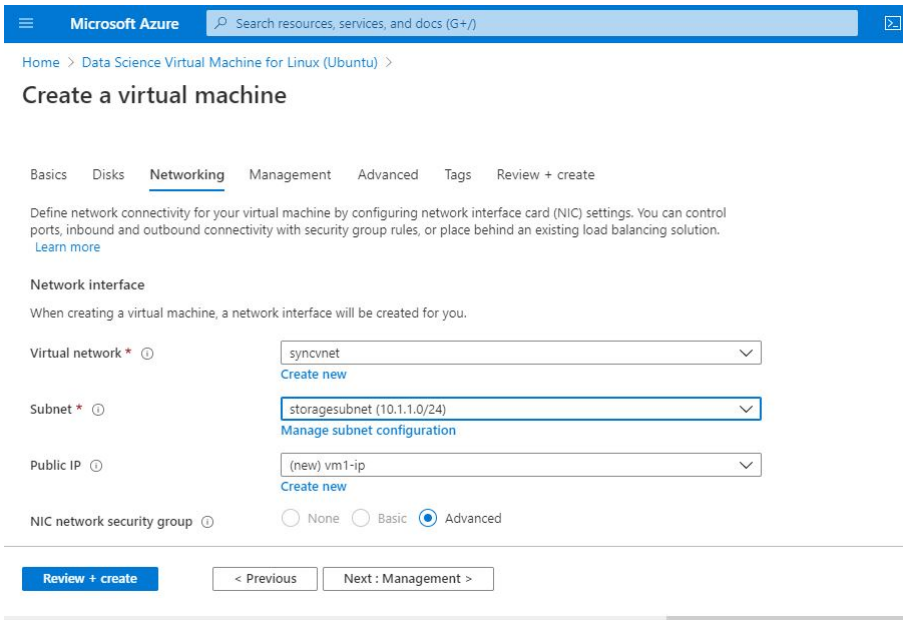


Figure 6.3: Microsoft Azure resource "Networking Settings".

After deployment of the instance, the Azure portal would provision one reserve public IP address e.g., 52.170.22.119 to the DSVM instance i.e., *VM*1, as shown in Figure 6.4.

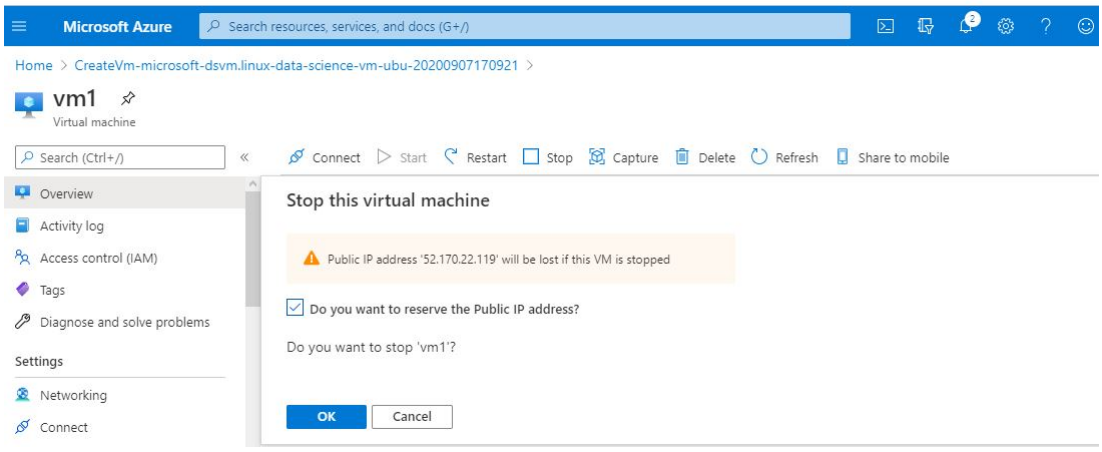


Figure 6.4: Reserve the public IP address for the *VM*1 DSVM instance.

6.3 Execute ML Algorithm in DSVM using SSH

In this Section, it is discussed how to connect and run ML Algorithm in Azure virtual machine from Windows 10 PC via Visual Studio (VS) Code which is an integrated development environment (IDE) [67] to combines common developer tools, such as *Azure Virtual Machines*

extension [68, 69] and *Remote SSH* [43, 70, 71], into a single graphical user interface (GUI).

First, the **Azure Virtual Machines extension** plug-in was Downloaded and installed for the VS Code. Now, this plug-in can create and manage Azure Virtual Machines directly from VS Code. It has several features such as view, create, delete, start, stop Azure Virtual Machines and add an SSH key to existing Azure Virtual Machines [69].

Further, to add the Azure file share to the VS code. It was one of the most time-consuming and difficult tasks for me. Because no concrete study article and material were found regarding how to connect the VS code with the Azure file share. Therefore, after doing lots of trial and error methods and troubleshooting was done to complete this task. There were three steps as following: (i) SSH to the Azure VM i.e, **vm1** using the following command: **ssh <vm username>@<vm ip address>** such as *vm*1@52.170.22.119 [41, 42], (ii) it must be logged-in again to the Azure subscription account from VS code with the following command: *az login* [72], into the command line, as shown in Figure 6.5, and (iii) the VS code could be mounted with the vm1 by copying and pasting the Linux command (see details command in Appendix B) for the Azure Files share [72].

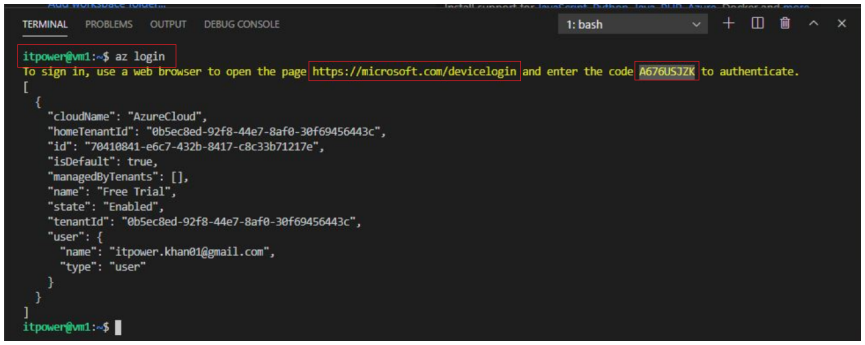


Figure 6.5: Sign in to Azure account with VS code.

In Figure 6.6 and Figure C.1, The final secure network topology would be depicted for P2S connection with Windows 10 PC to Azure file share and mounting the file share with DSVM with a detailed description (see datails in Appendix C).

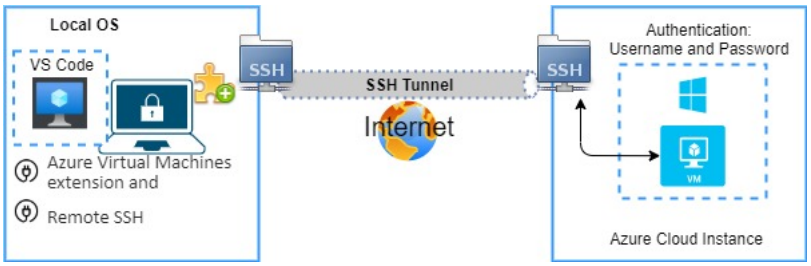


Figure 6.6: Remote SSH with VS code.

Furthermore, the ML code (i.e., VAE_GAN.py file) has been executed in the proposed P2S connection from VS code using SSH to DSVM,as shown in Figure 6.7 and Figure 6.8. Moreover, it was proof of confirmation to evaluate the approach (discussed in Chapter 5) was worked.

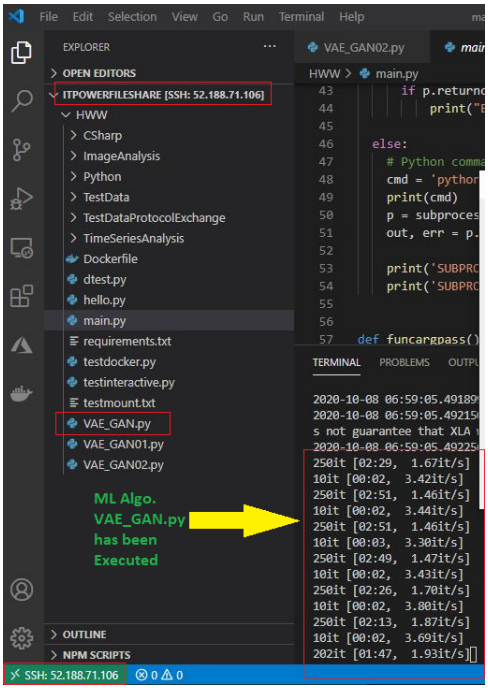


Figure 6.7: The execution of ML code (i.e., VAE_GAN.py file).

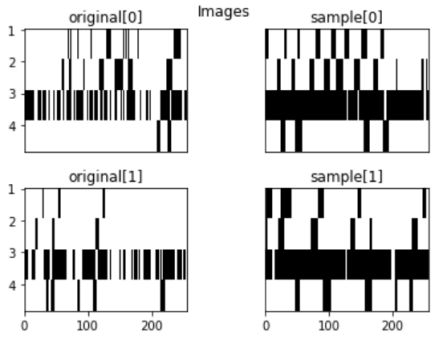


Figure 6.8: Evaluation for original and automatically generated signal pattern from test dataset using VAE algorithm [1] with P2S connection via SSH

6.4 Cost Analysis

In Figure 6.9 and Figure 6.10, the cost analysis [73] for one month of the proposed approach (discussed in Chapter 5) is given here to analyze the business cost [7, 74, 75] of the ITPower Solutions GmbH [1] if they will going to use this approach in the near future. Further, the cost analysis graph had been obtained from the **Cost Management** page of the Azure portal. However, this pricing and cost savings will change on several factors at *pay-as-you-go* upon the use of the cloud service [76].

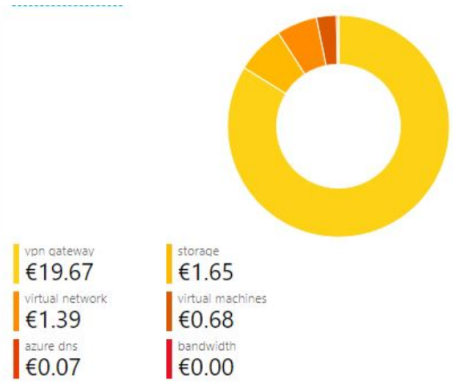


Figure 6.9: Cost Analysis by individual resource

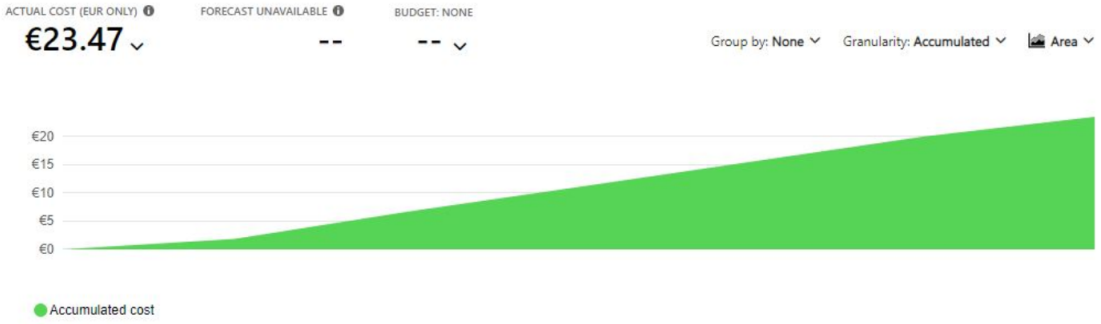


Figure 6.10: Accumulated cost for Cost analysis of Resource Group

7 Conclusion

In our problem statement, it was discussed that an adversary can change the training data during train an ML model and deceive the model with the testing dataset. Further, if we can think about a real-life threat model will be an attacker can magically transform the stop sign by adding an imperceptible noise into a Speed Limit (60km/h) in the eyes of a self-driving car [77, 78]. Therefore, an adversary of the 21st century will not necessarily need bombs, uranium, or biological weapons but he will need the only information. So, in the current world, information is the most valuable wealth of an organization which must have to be well protected from business loss.

We are living in the era of artificial intelligence. Nevertheless, the AI/ML algorithms are being called upon a problem in near future. Stéphane Mallat et al. explained that ML algorithms such as neural networks *"are black boxes"* [79]. That is, the way AI/ML model is learning with training datasets, an adversary can attack and control the prediction by testing the AI/ML model, as shown in Figure 7.1 [28].

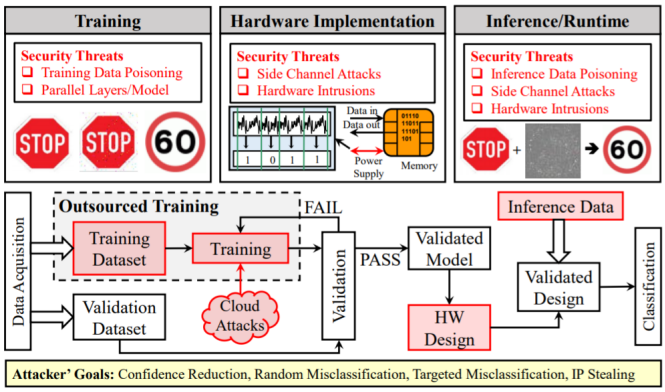


Figure 7.1: An Overview of Security Threats/Attacks and their respective payloads for Machine Learning Algorithms during Training, Inference, and their respective Hardware Implementations [28]

The main learning objective from this internship, as the AI/ML methods have vulnerabilities due to inherent limitations in the state-of-the-art model designs so that they are very dangerous and deceptive to leave and train the models in an open cloud environment which can be a possible cause of a devastating set of attacks.

Abbrevations and Acronyms

AI	artificial intelligent
ARM	Azure Resource Manager
DSVM	data science virtual machine
GB	Gigabyte
GPU	graphics processing unit
GPv2	general purpose v2
GUI	graphical user interface
HDD-based	hard disk-based
IaaS	infrastructure as a service
IDE	integrated development environment
LRS	locally-redundant storage
MITM	man-in-the-middle
ML	machine learning
OS	operating system
PaaS	platform as a service
P2S	point-to-site
PKI	public key infrastructure
QA	quality assurance
RAM	Random-access memory
SaaS	software as a service
S2S	site-to-site
SDK	software development kit
SMB	server message block
SSH	Secure Shell Protocol
UNC	universal naming convention
URL	uniform resource locator
VAE	variational autoencoder
VM	virtual machine
VNet	virtual network

VPN	virtual private network
VS	Visual Studio
WGAN	Wasserstein generative adversarial network

Bibliography

[1] Dr. Sadegh Sadeghipour. ITPower Solutions GmbH, Kolonnenstraße 26 10829 Berlin Germany. https://itpower.de/en/company/about-us/. 2000.

[2] Subodh Gangan. *A review of man-in-the-middle attacks*. In: *arXiv preprint arXiv:1504.-02115* (2015).

[3] Mauro Conti, Nicola Dragoni and Viktor Lesyk. *A Survey of Man In The Middle Attacks*. In: *IEEE Communications Surveys Tutorials* 18.3 (2016), pp. 2027–2051. DOI: 10.1109/COMST.2016.2548426.

[4] Netsparker Security Team. *Man-in-the-Middle Attacks and How To Avoid Them*. Web Security Blog. July 2019. URL: https://www.netsparker.com/blog/web-security/man-in-the-middle-attack-how-avoid/.

[5] Florent Gontharet. *Man-in-The-MiddleAttacks and Countermeasures Analysis*. https://www.researchgate.net/publication/340720434_Man-in-The-Middle_Attacks_Countermeasures_Analysis. MSc. Ethical Hacking and Computer Security. Baxter Building, Dundee Scotland, DD1 1HG: University of Abertay Dundee, Aug. 2015.

[6] Dor Bank, Noam Koenigstein and Raja Giryes. *Autoencoders*. In: *arXiv preprint arXiv:2003.05991* (2020).

[7] Adrian Rosebrock. *My review of Microsoft's data science virtual machine (DSVM) for deep learning*. PyImageSearch Blog. Mar. 2018. URL: https://www.pyimagesearch.com/2018/03/21/my-review-of-microsofts-deep-learning-virtual-machine/.

[8] Ashish Singh and Kakali Chatterjee. *Cloud security issues and challenges: A survey*. In: *Journal of Network and Computer Applications* 79 (2017), pp. 88–115.

[9] Deyan Chen and Hong Zhao. *Data security and privacy protection issues in cloud computing*. In: *2012 International Conference on Computer Science and Electronics Engineering*. Vol. 1. IEEE. 2012, pp. 647–651.

[10] Micah Goldblum et al. *Data Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses*. In: *arXiv preprint arXiv:2012.10544* (2020).

[11] M. Jagielski et al. *Manipulating Machine Learning: Poisoning Attacks and Countermeasures for Regression Learning*. In: *2018 IEEE Symposium on Security and Privacy (SP)*. 2018, pp. 19–35. DOI: 10.1109/SP.2018.00057.

[12] Abdullah Abuhussein, Harkeerat Bedi and Sajjan Shiva. *Evaluating security and privacy in cloud computing services: A Stakeholder's perspective*. In: *2012 international conference for internet technology and secured transactions*. Ieee. 2012, pp. 388–395.

[13] Wenbo Jiang et al. *A flexible poisoning attack against machine learning*. In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–6.

[14] Functionize Team. *Automated Web Testing*. In: *Functionize, Inc.* (Jan. 2021). `https://www.functionize.com/automated_web-testing/`.

[15] testim Team. *Testim Automate*. In: *testim blog* (Jan. 2021). `https://www.testim.io/test-automation-tool/`.

[16] Gartner Inc. *Software Test Automation Reviews and Ratings*. `https://www.gartner.com/reviews/market/software-test-automation`. Mar. 2021.

[17] Martin Arjovsky, Soumith Chintala and Léon Bottou. *Wasserstein generative adversarial networks*. In: *International conference on machine learning*. PMLR. 2017, pp. 214–223.

[18] Ishaan Gulrajani et al. *Improved training of wasserstein gans*. In: *arXiv preprint arXiv:1704.00028* (2017).

[19] Diederik P Kingma and Max Welling. *Auto-encoding variational bayes*. In: *arXiv preprint arXiv:1312.6114* (2013).

[20] Danilo Jimenez Rezende, Shakir Mohamed and Daan Wierstra. *Stochastic backpropagation and approximate inference in deep generative models*. In: *International conference on machine learning*. PMLR. 2014, pp. 1278–1286.

[21] Wayne Hoggett. *Windows 10 Support for Azure File Sync*. `https://feedback.azure.com/forums/217298-storage/suggestions/35700508-windows-10-support-for-azure-file-sync`. Oct. 2018.

[22] Haidong Xia and José Carlos Brustoloni. *Hardening web browsers against man-in-the-middle and eavesdropping attacks*. In: *Proceedings of the 14th international conference on World Wide Web*. 2005, pp. 489–498.

[23] Ryan. *Encryption in-transit and Encryption at-rest – Definitions and Best Practices*. In: *Ryadel* (Mar. 2019). https://www.ryadel.com/en/data-encryption-in-transit-at-rest-definitions-best-practices-tutorial-guide/.

[24] Aniruddha Saha, Akshayvarun Subramanya and Hamed Pirsiavash. *Hidden trigger backdoor attacks*. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 11957–11965.

[25] Hai Huang et al. *Data Poisoning Attacks to Deep Learning Based Recommender Systems*. In: *arXiv preprint arXiv:2101.02644* ().

[26] Microsoft Azure. *Agent installation and server configuration*. `https://docs.microsoft.com/en-us/azure/storage/files/storage-files-release-notes`. Mar. 2021.

[27] Prabath Anuradha. *Enable immediate sync after changes on the Azure file share for Azure File Sync*. `https://feedback.azure.com/forums/217298-storage/suggestions/33072151-enable-immediate-sync-after-changes-on-the-azure-f`. Jan. 2018.

[28] Faiq Khalid et al. *Security for machine learning-based systems: Attacks and challenges during training and inference*. In: *2018 International Conference on Frontiers of Information Technology (FIT)*. IEEE. 2018, pp. 327–332.

[29] Microsoft Team. *Create a Site-to-Site connection using the Azure portal (classic)*. In: *Microsoft Azure Documentation* (Oct. 2020). https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-site-to-site-classic-portal.

[30] Flo Fox. *Should I be using Azure Files?* In: *Altaro Blog* (Jan. 2021). https://www.altaro.com/hyper-v/azure-files/.

[31] Microsoft Team. *Microsoft Azure portal*. In: *Microsoft* (Apr. 2021). https://azure.microsoft.com/en-us/features/azure-portal/.

[32] Arron Fu. *7 Different Types of Cloud Computing Structures*. In: *UniPrint.net* (Mar. 2017). https://www.uniprint.net/en/7-types-cloud-computing-structures/.

[33] Microsoft Team. *Azure documentation*. In: *Microsoft* (Apr. 2021). https://docs.microsoft.com/en-us/azure/?product=featured.

[34] Dave Cutler. *Microsoft Azure*. In: *Wikipedia* (Apr. 2021). https://en.wikipedia.org/wiki/Microsoft_Azure.

[35] Jim Dowling. *Introduction to Cloud Computing*. In: *KTH Royal Institute of Technology* (Aug. 2020). https://www.kth.se/social/files/554fa451f276544829be2e5e/9-cloud-computing.pdf.

[36] Bijay Kumar. *How to Create Azure Free Account (Step by Step tutorial)*. In: *Azure Lessons* (Apr. 2020). https://azurelessons.com/create-azure-free-account/.

[37] Microsoft Team. *Create your Azure free account today*. In: *Microsoft Azure* (Apr. 2021). https://azure.microsoft.com/en-us/free/.

[38] Microsoft Team. *Create your Azure free account today*. In: *Microsoft Azure* (Apr. 2021). https://azure.microsoft.com/en-in/free/students/.

[39] Francisco Navarro. *How to Set up Azure File Sync*. In: *Adam the Automator* (Nov. 2020). https://adamtheautomator.com/azure-file-sync/.

[40] Dishan M. Francis. *Step-by-Step guide to create Azure file share and Map it in Windows 10*. In: *Rebel Admin* (Mar. 2018). https://www.rebeladmin.com/2018/03/step-step-guide-create-azure-file-share-map-windows-10/.

[41] Joel Guerra. *How to: SSH to your Azure Linux VMs with username and password from Windows, Linux or Mac*. In: *Medium Blog* (Mar. 2018). https://joelatwar.medium.com/how-to-ssh-to-your-azure-linux-vms-with-username-and-password-from-windows-linux-or-mac-df7d07ea3be1.

[42] Sana Ajani. *Remote SSH with Visual Studio Code*. In: *Visual Studio Code Blog* (June 2019). https://code.visualstudio.com/blogs/2019/07/25/remote-ssh.

[43] Microsoft Team. *How to use SSH keys with Windows on Azure*. In: *Microsoft Azure Document* (Oct. 2020). https://docs.microsoft.com/en-us/azure/virtual-machines/linux/ssh-from-windows.

[44] Kalyan Bandarupalli. *Connecting On-Premises Networks with AZURE Virtual Networks*. In: *Tech bubbles blog* (Feb. 2019). http://www.techbubbles.com/azure/connecting-on-premises-networks-with-azure-virtual-networks/.

[45] Dishan Francis. *Step-By-Step: Creating an Azure Point-to-Site VPN*. In: *Microsoft tech community* (May 2019). https://techcommunity.microsoft.com/t5/itops-talk-blog/step-by-step-creating-an-azure-point-to-site-vpn/ba-p/326264.

[46] Microsoft Team. *Configure a Point-to-Site VPN connection to a VNet using native Azure certificate authentication: Azure portal*. In: *Microsoft Azure Documentation* (Oct. 2021). https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-howto-point-to-site-resource-manager-portal.

[47] Microsoft Team. *About Point-to-Site VPN*. In: *Microsoft Azure Document* (Sept. 2020). https://docs.microsoft.com/en-us/azure/vpn-gateway/point-to-site-about.

[48] Microsoft Team. *MakeCert*. In: *Microsoft Azure Document* (May 2018). https://docs.microsoft.com/en-us/windows/win32/seccrypto/makecert.

[49] Microsoft Developer Team. *Windows 10 SDK*. In: *Microsoft Azure Document* (Dec. 2020). https://developer.microsoft.com/en-us/windows/downloads/windows-10-sdk/.

[50] Wikipedia Team. *X.509*. In: *Wikipedia Documents* (Apr. 2021). https://en.wikipedia.org/wiki/X.509.

[51] Microsoft Team. *Generate and export certificates for Point-to-Site connections using MakeCert*. In: *Microsoft Azure Document* (Feb. 2020). https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-certificates-point-to-site-makecert.

[52] Microsoft Team. *Install client certificates for P2S certificate authentication connections*. In: *Microsoft Azure Documentation* (Feb. 2020). https://docs.microsoft.com/en-us/azure/vpn-gateway/point-to-site-how-to-vpn-client-install-azure-cert.

[53] Microsoft Team. *Deploy resources with ARM templates and Azure portal*. In: *Microsoft Azure Documentation* (Oct. 2020). https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/deploy-portal.

[54] Juniper Networks Team. *Create a Resource Group*. In: *TechLibrary Blog, Juniper Networks* (Nov. 2020). https://www.juniper.net/documentation/us/en/software/vsrx/vsrx-azure/topics/task/security-vsrx-azure-marketplace-resource-group.html.

[55] Microsoft Team. *What is Azure Resource Manager?* In: *Microsoft Azure Document* (Mar. 2021). https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview.

[56] Microsoft Team. *About VPN Gateway configuration settings: Gateway SKUs*. In: *Microsoft Azure Documentation* (Oct. 2020). https://docs.microsoft.com/en-us/azure/vpn-gateway/vpn-gateway-about-vpn-gateway-settings.

[57] Microsoft Team. *Create and install VPN client configuration files for native Azure certificate authentication P2S configurations*. In: *Microsoft Azure Documentation* (Nov. 2020). https://docs.microsoft.com/en-us/azure/vpn-gateway/point-to-site-vpn-client-configuration-azure-cert.

[58] Microsoft Team. *Create an Azure file share*. In: *Microsoft Azure Documentation* (Apr. 2021). https://docs.microsoft.com/en-us/azure/storage/files/storage-how-to-create-file-share?tabs=azure-portal.

[59] Microsoft Team. *Use an Azure file share with Windows*. In: *Microsoft Azure Documentation* (Apr. 2021). https://docs.microsoft.com/en-us/azure/storage/files/storage-how-to-use-files-windows.

[60] Microsoft Team. *Naming Files, Paths, and Namespaces*. In: *Microsoft Azure Documentation* (Sept. 2020). https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file.

[61] Microsoft Team. *What is Azure Private Endpoint?* In: *Microsoft Azure Documentation* (Sept. 2020). https://docs.microsoft.com/en-us/azure/private-link/private-endpoint-overview.

[62] Microsoft Team. *What is Azure Private Link service?* In: *Microsoft Azure Documentation* (Sept. 2020). https://docs.microsoft.com/en-us/azure/private-link/private-link-service-overview.

[63] Microsoft Team. *What is Azure Private Link?* In: *Microsoft Azure Documentation* (Mar. 2021). https://docs.microsoft.com/en-us/azure/private-link/private-link-overview.

[64] Microsoft Team. *Use private endpoints for Azure Storage*. In: *Microsoft Azure Documentation* (Mar. 2021). https://docs.microsoft.com/en-us/azure/storage/common/storage-private-endpoints.

[65] Microsoft Team. *Quickstart: Create a Private Endpoint using the Azure portal*. In: *Microsoft Azure Documentation* (Oct. 2020). https://docs.microsoft.com/en-us/azure/private-link/create-private-endpoint-portal.

[66] Microsoft Team. *Tutorial: Connect to a storage account using an Azure Private Endpoint*. In: *Microsoft Azure Documentation* (Sept. 2020). https://docs.microsoft.com/en-us/azure/private-link/tutorial-private-endpoint-storage-portal.

[67] Wikipedia Team. *Integrated development environment*. What is an IDE? Jan. 2021. URL: https://en.wikipedia.org/wiki/Integrated_development_environment.

[68] Microsoft Team. *Azure virtual machine extensions and features*. Azure Virtual Machines. Jan. 2021. URL: https://docs.microsoft.com/en-us/azure/virtual-machines/extensions/overview.

[69] Visual Studio Code Team. *Azure Virtual Machines*. Azure Virtual Machines extension. Jan. 2021. URL: https://marketplace.visualstudio.com/items?itemName=ms-azuretools.vscode-azurevirtualmachines.

[70] Microsoft Team. *Remote - SSH*. Visual Studio Code Remote - SSH. Jan. 2021. URL: https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-ssh.

[71] Visual Studio Code Team. *Remote development over SSH*. Connect using SSH. Jan. 2021. URL: https://code.visualstudio.com/docs/remote/ssh-tutorial.

[72] Microsoft Azure Team. *Use Azure Files with Linux*. Azure Storage Files. May 2021. URL: https://docs.microsoft.com/en-us/azure/storage/files/storage-how-to-use-files-linux?tabs=smb311.

[73] Microsoft Team. *Azure Private Link pricing*. In: *Microsoft Azure Documentation* (Jan. 2021). https://azure.microsoft.com/en-us/pricing/details/private-link/.

[74] Jay Chapel. *Cloud Storage Cost Comparison: AWS vs. Azure vs. Google*. Medium blog. July 2019. URL: https://jaychapel.medium.com/cloud-storage-cost-comparison-aws-vs-azure-vs-google-844dfff3d324.

[75] ParkMyCloud Team. *Cloud Pricing Comparison in 2020*. ParkMyCloud blog. Jan. 2020. URL: https://www.parkmycloud.com/the-cloud-pricing-comparison/.

[76] Microsoft Team. *Azure pricing*. Microsoft Azure document blog. Jan. 2021. URL: https://azure.microsoft.com/en-us/pricing/.

[77] Marcus Comiter. *Attacking Artificial Intelligence*. In: *Belfer Center Paper* (2019).

[78] Jeff Jun Zhang et al. *Building robust machine learning systems: Current progress, research challenges, and opportunities*. In: *Proceedings of the 56th Annual Design Automation Conference 2019*. 2019, pp. 1–4.

[79] John Zarka et al. *Deep network classification by scattering and homotopy dictionary learning*. In: *arXiv preprint arXiv:1910.03561* (2019).

A Appendix: Windows CLI Command

The user and password of *"itpowerfileshare"* to map a storage file share with localhost PC for later configuration on Windows CLI command, as shown in Figure A.1

```
$connectTestResult = Test-NetConnection -ComputerName
itpowerstorage.file.core.windows.net -Port 445
if ($connectTestResult.TcpTestSucceeded) {
    # Save the password so the drive will persist on reboot
    cmd.exe /C "cmdkey /add:`"itpowerstorage.file.core.windows.net`"
/user:`"Azure\itpowerstorage`"
/pass:`"N7VTh5+S/mwUbwzNqE84a6TTELJ0vpwnQDuCibyekw+U+6oCLzksceQDkwr8q+W3y1YRWraSQktaA
NBV6gOaaA==`""
    # Mount the drive
    New-PSDrive -Name Z -PSProvider FileSystem -Root
"\\itpowerstorage.file.core.windows.net\itpowerfileshare" -Persist
} else {
    Write-Error -Message "Unable to reach the Azure storage account via port 445.
Check to make sure your organization or ISP is not blocking port 445, or use Azure
P2S VPN, Azure S2S VPN, or Express Route to tunnel SMB traffic over a different
port."
}
```

Figure A.1: Mount the Share File *itpowerstorage* on Windows 10 localhost PC

B Appendix: Use Azure Files with Linux

To mount the Azure file share with VS code, it was copied the following command [72] and pasted to the bash command prompt.

How to Mount File Storage on Linux:

```
sudo mkdir /mnt/itpowerfileshare
if [ ! -d "/etc/smbcredentials" ]; then
sudo mkdir /etc/smbcredentials
fi
if [ ! -f "/etc/smbcredentials/itpowerstorage.cred" ]; then
    sudo bash -c 'echo "username=itpowerstorage" >>
/etc/smbcredentials/itpowerstorage.cred'
    sudo bash -c 'echo
"password=N7VTh5+S/mwUbwzNqE84a6TTELJ0vpwnQDuCibyekw+U+6oCLzksceQDkwr8q+W3y1YRWraSQkt
aANBV6gOaaA==" >> /etc/smbcredentials/itpowerstorage.cred'
fi
sudo chmod 600 /etc/smbcredentials/itpowerstorage.cred

sudo bash -c 'echo "//itpowerstorage.file.core.windows.net/itpowerfileshare
/mnt/itpowerfileshare cifs
nofail,vers=3.0,credentials=/etc/smbcredentials/itpowerstorage.cred,dir_mode=0777,fil
e_mode=0777,serverino" >> /etc/fstab'
sudo mount -t cifs //itpowerstorage.file.core.windows.net/itpowerfileshare
/mnt/itpowerfileshare -o
vers=3.0,credentials=/etc/smbcredentials/itpowerstorage.cred,dir_mode=0777,file_mode=
0777,serverino
```

Figure B.1: Mount Azure File Storage on Linux DSVM i.e., $vm1$

C Appendix: Test Environment Creation

It can be used the following values to create a test environment, or refer to these values to better understand the examples in this report:

Subscription If user have one subscription, verify and sign-in with using the correct user name and password.

Resource Group syncrg (sync resource group).

Location East US.

VNet Name syncvnet.

Address space 10.1.0.0/16.

Note: we can have more than one address space for your VNet

Subnet name default.

Subnet address range 10.1.0.0/24.

Subnet name storagesubnet.

Subnet address range 10.1.1.0/24.

Subnet name GateWaySubnet.

Virtual network gateway name sync-vng.

Gateway type VPN.

VPN type Route-based.

Public IP address name vng-ip.

Gate Way Subnet 10.1.2.0/24.

SKU Basic (But in my case I selected VpnGw1, which was deducted some cost form my subscription student account).

Note: The Basic SKU does not support IKEv2 or RADIUS authentication.

Connection type Point-to-site.

Client address pool 72.16.0.0/24.

Root certificate file name P2SRootCert_Base64.cer.

Client certificate file name P2SChildCert.pfx.

Root Certificate Name P2SRootCert.

Name of Client VPN syncvnet.

Storage account Name itpowerstorage.

Name file share itpowerfileshare.

Name private endpoints syncpvtendpoint.

Private endpoint (private IP address) 10.1.1.4/24.

Reserve public IP address for DSVM 52.170.22.119.

Visual Studio Code plugin Remote – SSH and Azure Virtual Machines extension.

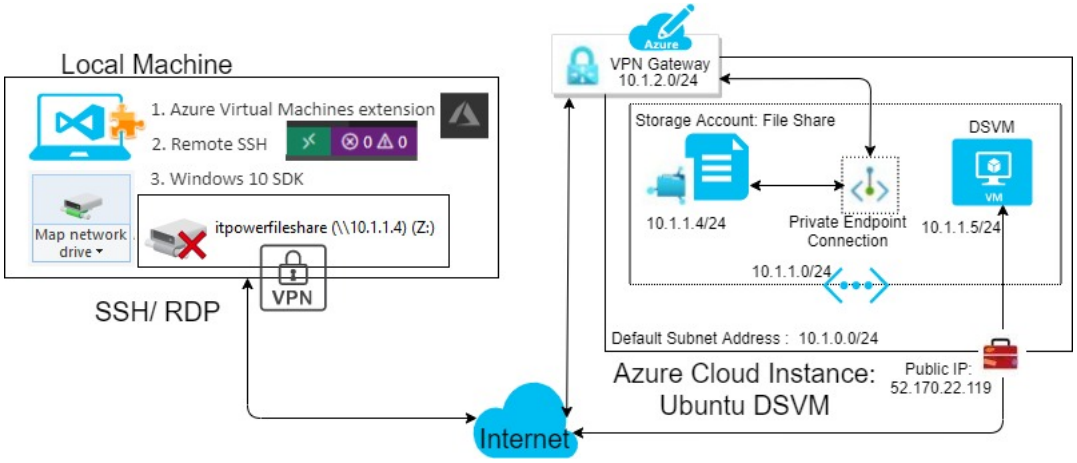


Figure C.1: Securing network topology for P2S connection from Windows 10 to Azure DSVM and mount the Azure file share.