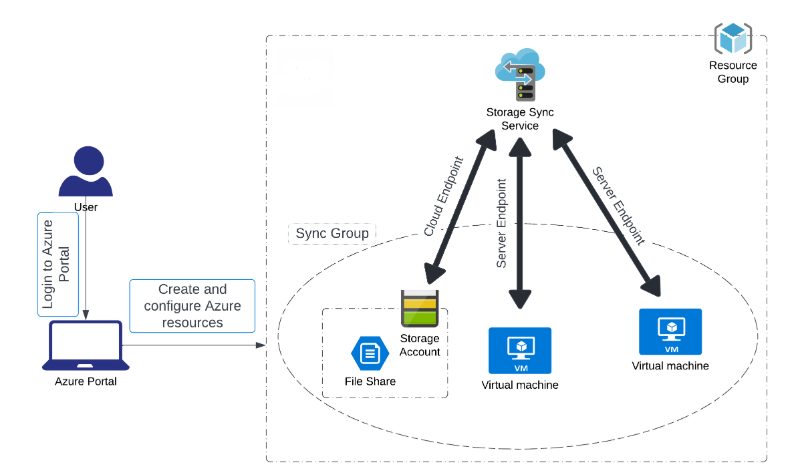
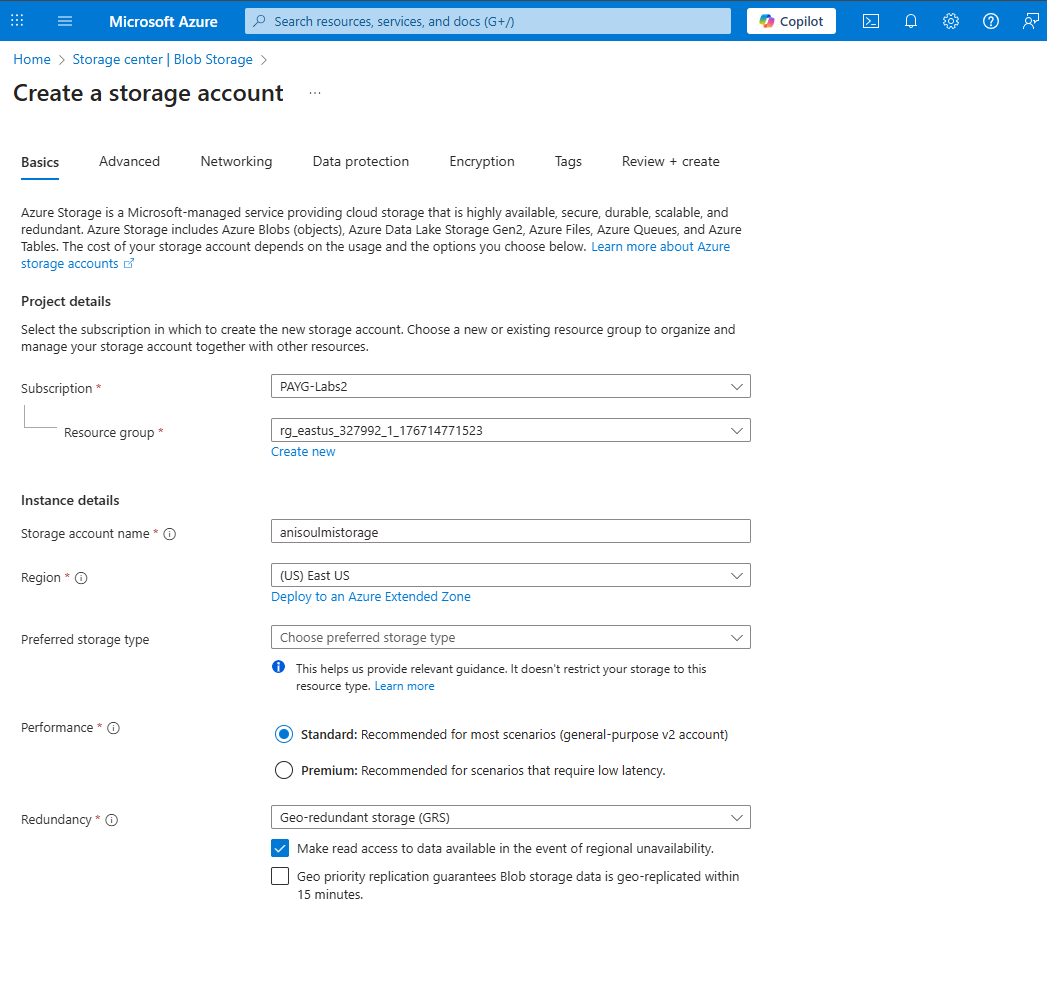
**Azure File Sync**

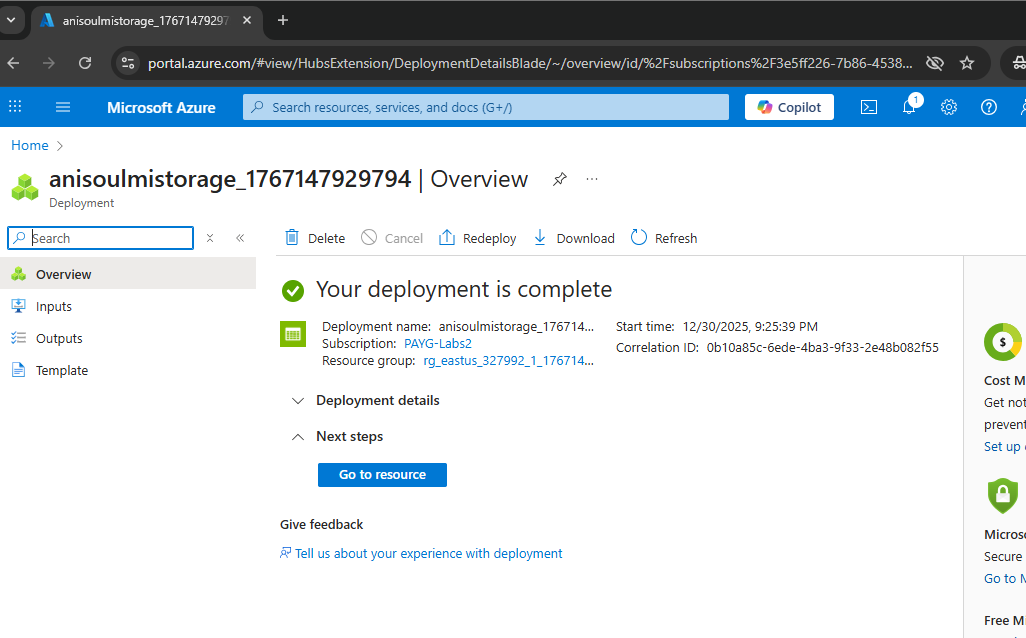
**Architecture Diagram**

****

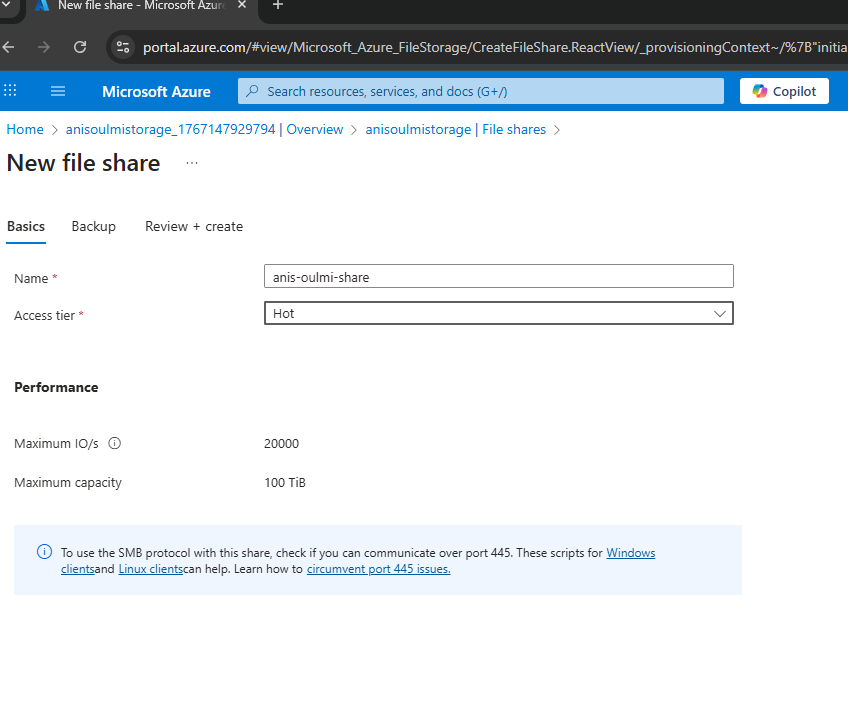
**Task 1: Create a Storage Account**

In today’s Azure lab, I explored how to set up cloud resources. I started by signing into the Azure portal, making sure to clear any cache to avoid login issues. Then, I created a storage account, which is basically a secure place to store data in the cloud. I placed it in a specific resource group, chose the East US region, standard performance, and kept the default redundancy settings to ensure my data is safe.

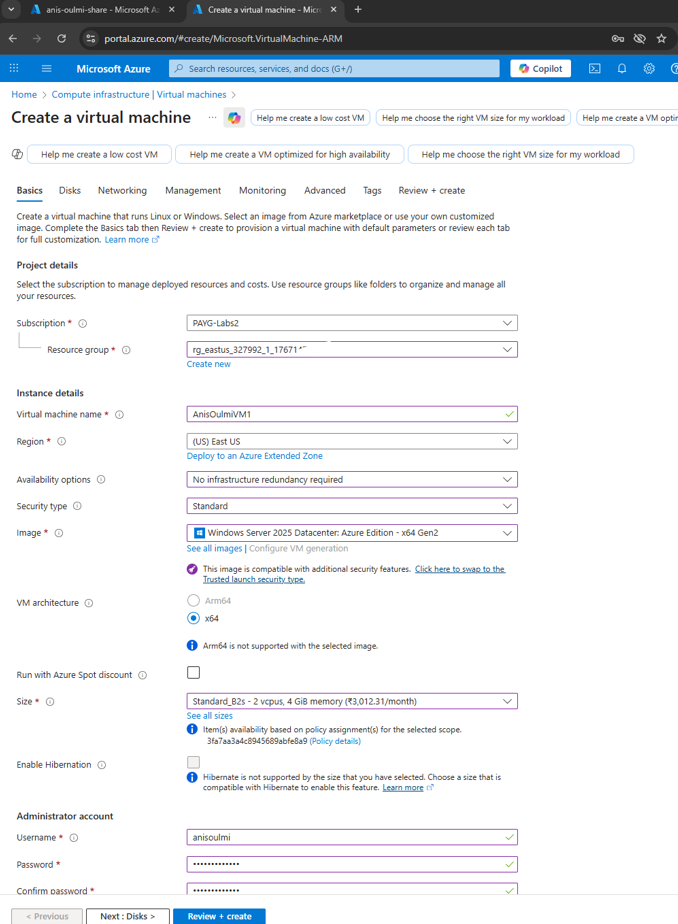


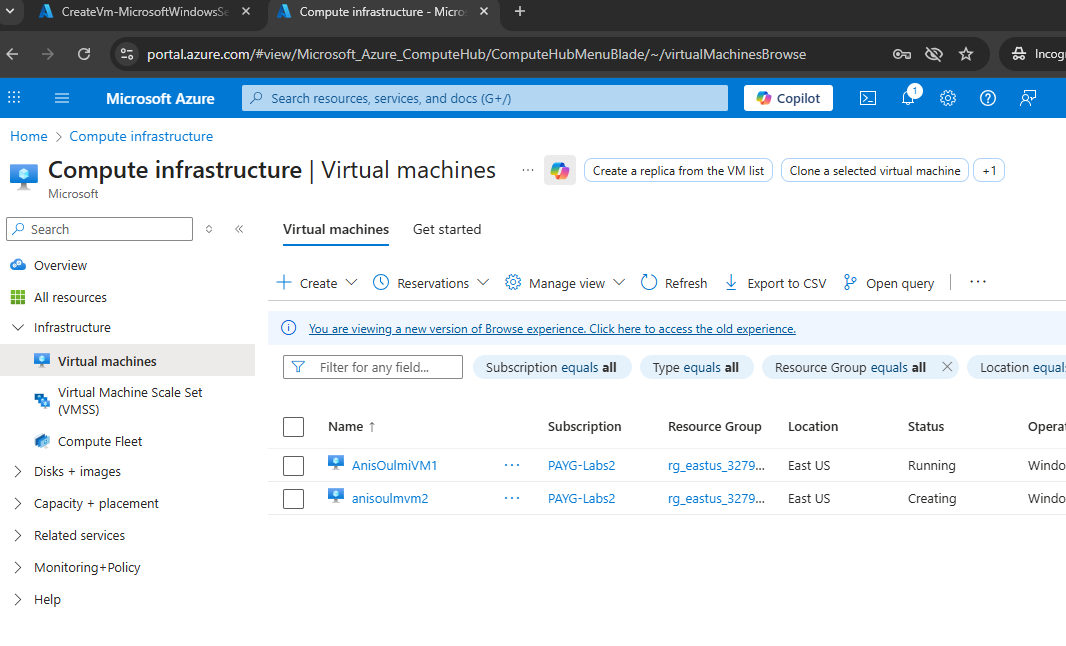


**Task 2: Create an SMB File Share**

Next, I created an SMB file share within my storage account. This is essentially a network-accessible folder in the cloud. I named it “anisoulmistorage” and selected the Hot tier, which is optimized for frequently accessed files

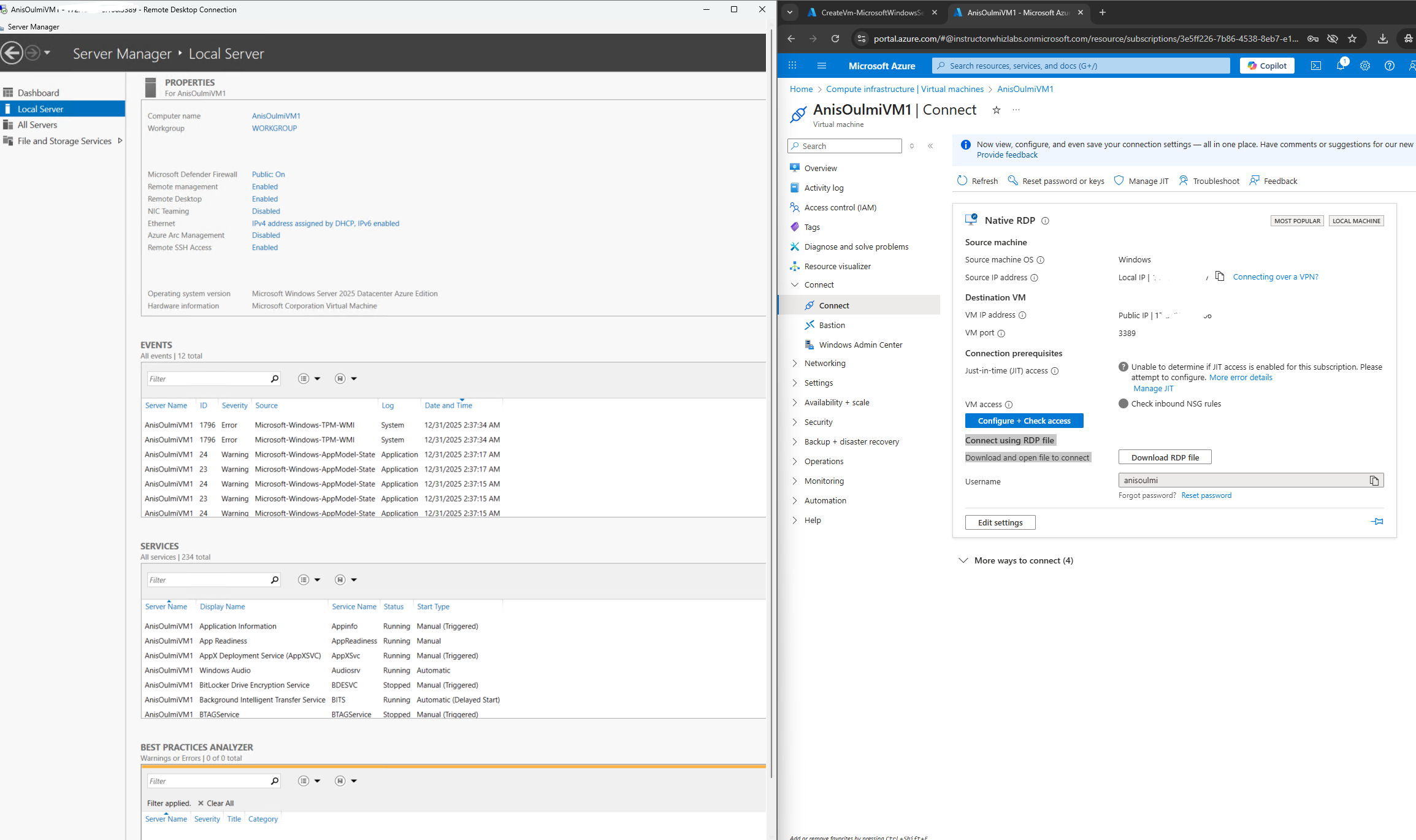
**Task 3: Deploy two Windows Servers**

After setting up storage, I deployed two Windows servers in Azure. Both virtual machines were configured similarly, using Windows Server 2025, the East US region, standard security settings, and a Standard SSD for the OS disk. I set up administrator accounts and enabled important ports like HTTP and RDP for remote access 



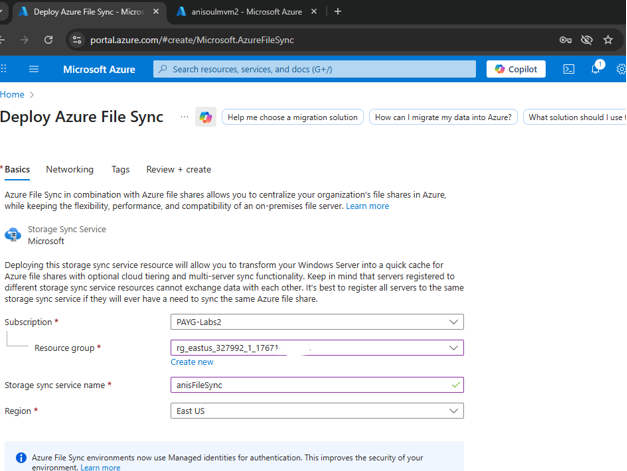
**Task 4: Prepare Windows Server to use with Azure File Sync**

After deploying the virtual machines, I prepared both servers to work with Azure File Sync. I connected to each VM remotely using RDP, logged in with the administrator accounts, and made some essential configurations, like turning off the Internet Explorer Enhanced Security settings to allow smoother management.

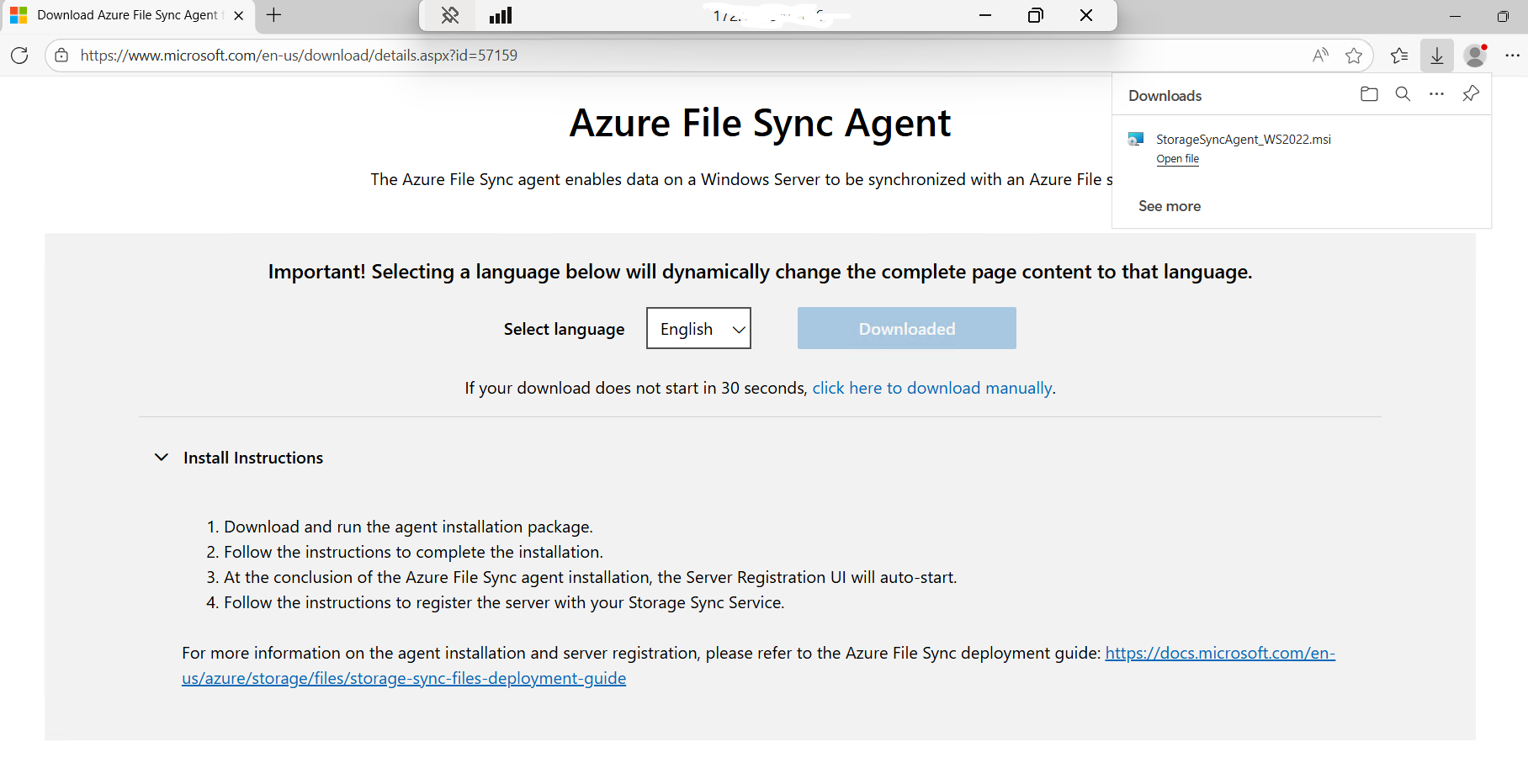


**Task 5: Deploy the Storage Sync Service:**

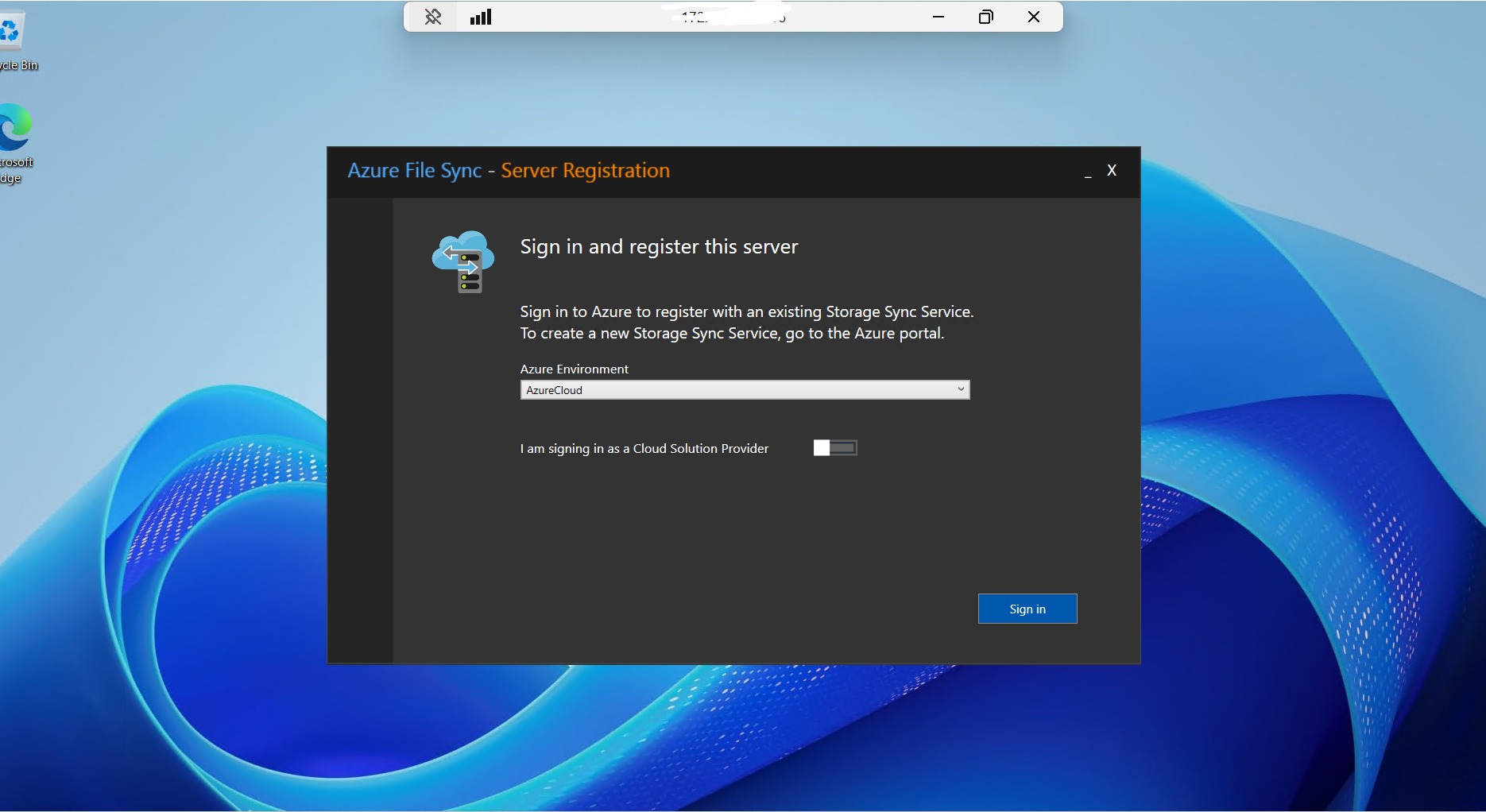
Once the servers were ready, I deployed the Storage Sync Service in Azure. This service acts as the bridge between the cloud file share and the on-premises-like servers, enabling seamless synchronization of files. I configured it within the same resource group, gave it a unique name, and selected the East US region, completing the setup needed to keep the cloud and server files in sync.

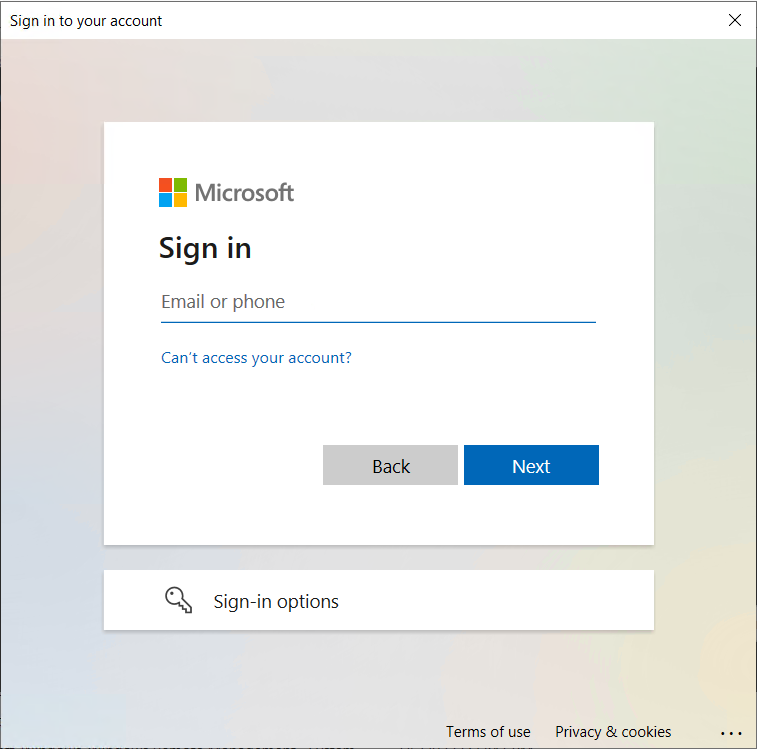


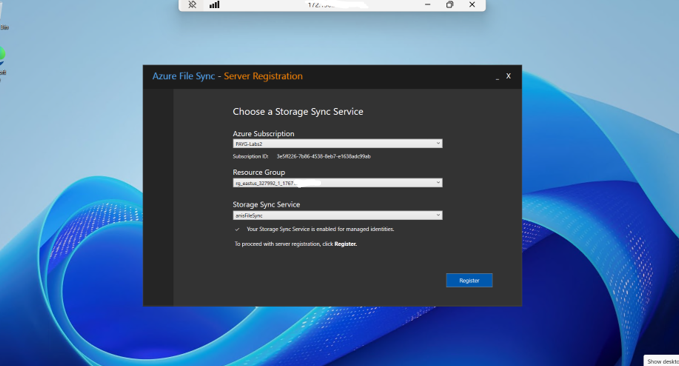
**Task 6: Install the Azure File Sync Agent**The next step was to install the Azure File Sync agent on the first Windows server. This agent is what allows the server to communicate with the Azure File Sync service. I downloaded the installer from Microsoft, ran it, and completed the installation

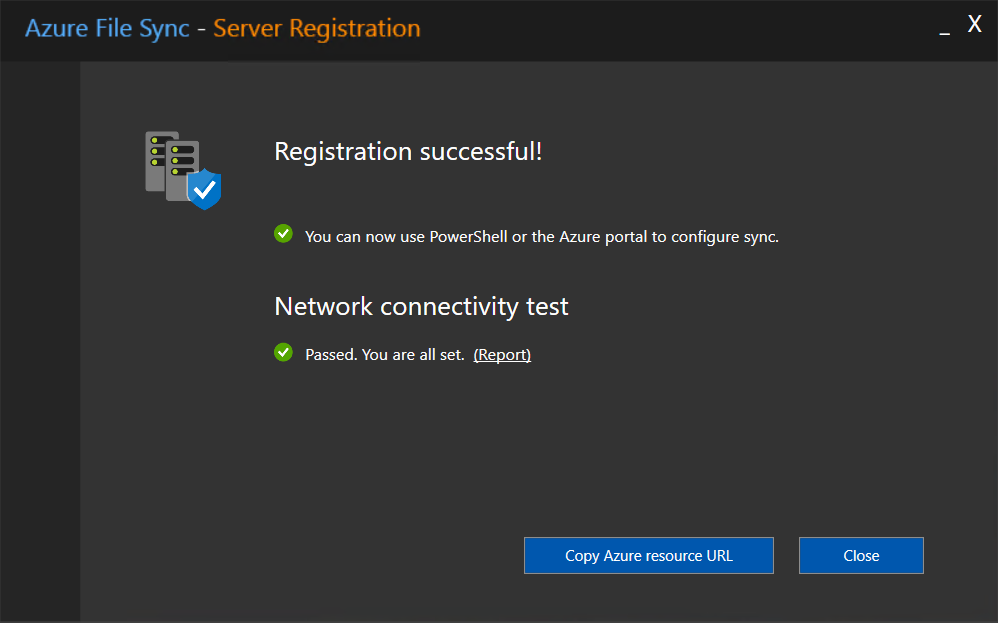


**Task 7: Register Windows Server with Storage Sync Service:**After installing the Azure File Sync agent, I registered the Windows server with the Storage Sync Service. This involved signing in with my Azure credentials and linking the server to the specific subscription, resource group, and the File Sync service I had previously created. Once registered, the server became officially connected to Azure File Sync, allowing it to start synchronizing files with the cloud.



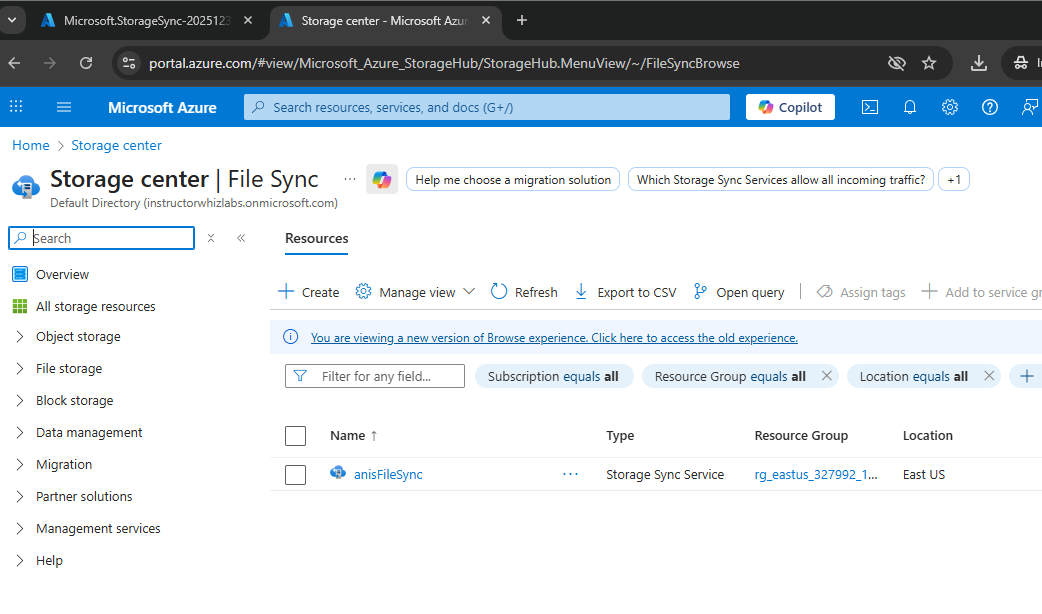


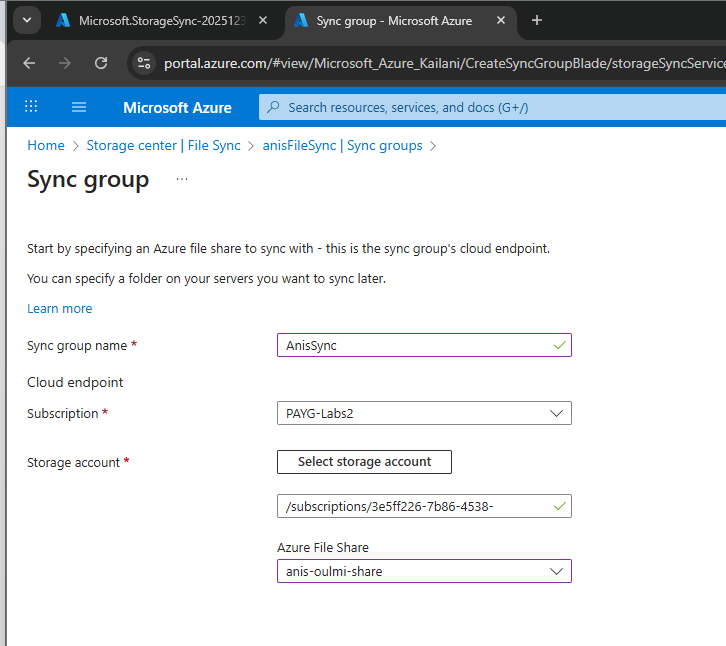


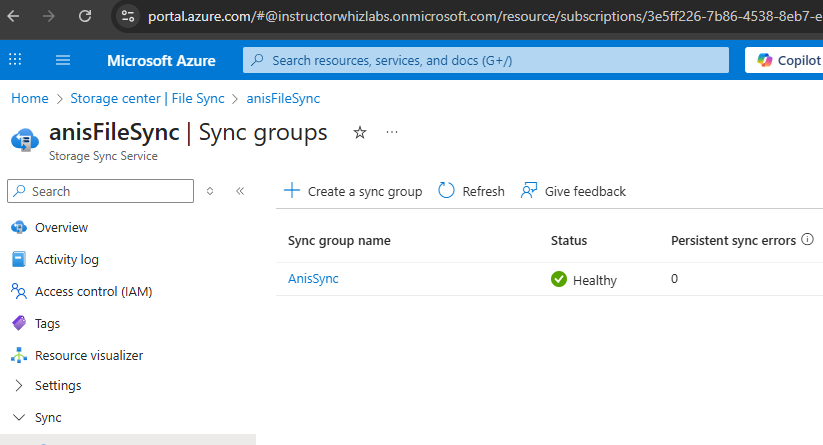


**Task 8: Create a Sync Group and a Cloud Endpoint**

Next, I created a Sync Group and connected it to a cloud endpoint. A Sync Group is essentially a container that manages file synchronization between servers and Azure. I linked it to the storage account and the file share I had created earlier, which set up the pathway for files to stay synchronized between the cloud and the on-premises-like servers. This step completed the configuration needed for seamless file management across environments.

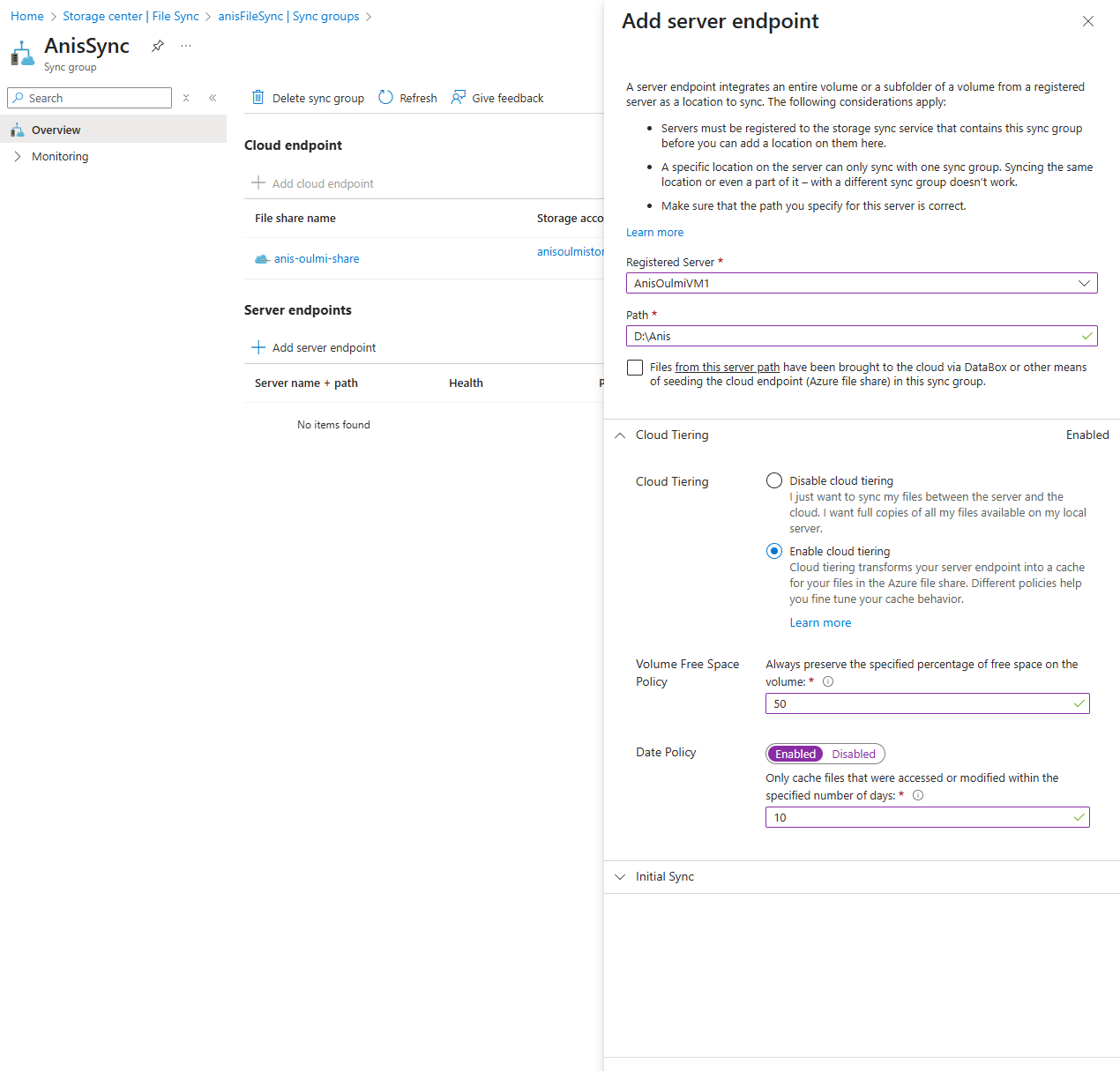






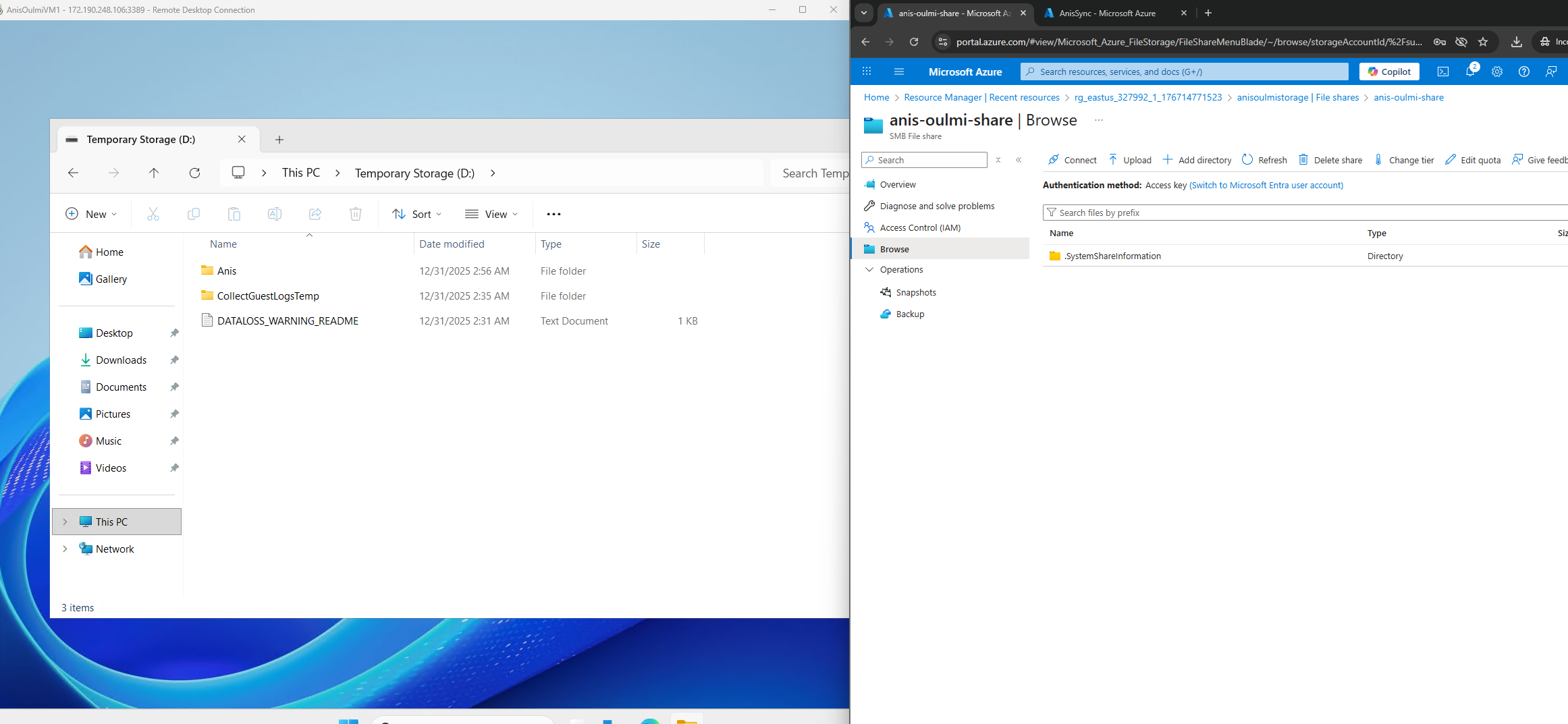
**Task 9: Create a Server Endpoint in the Sync Group**

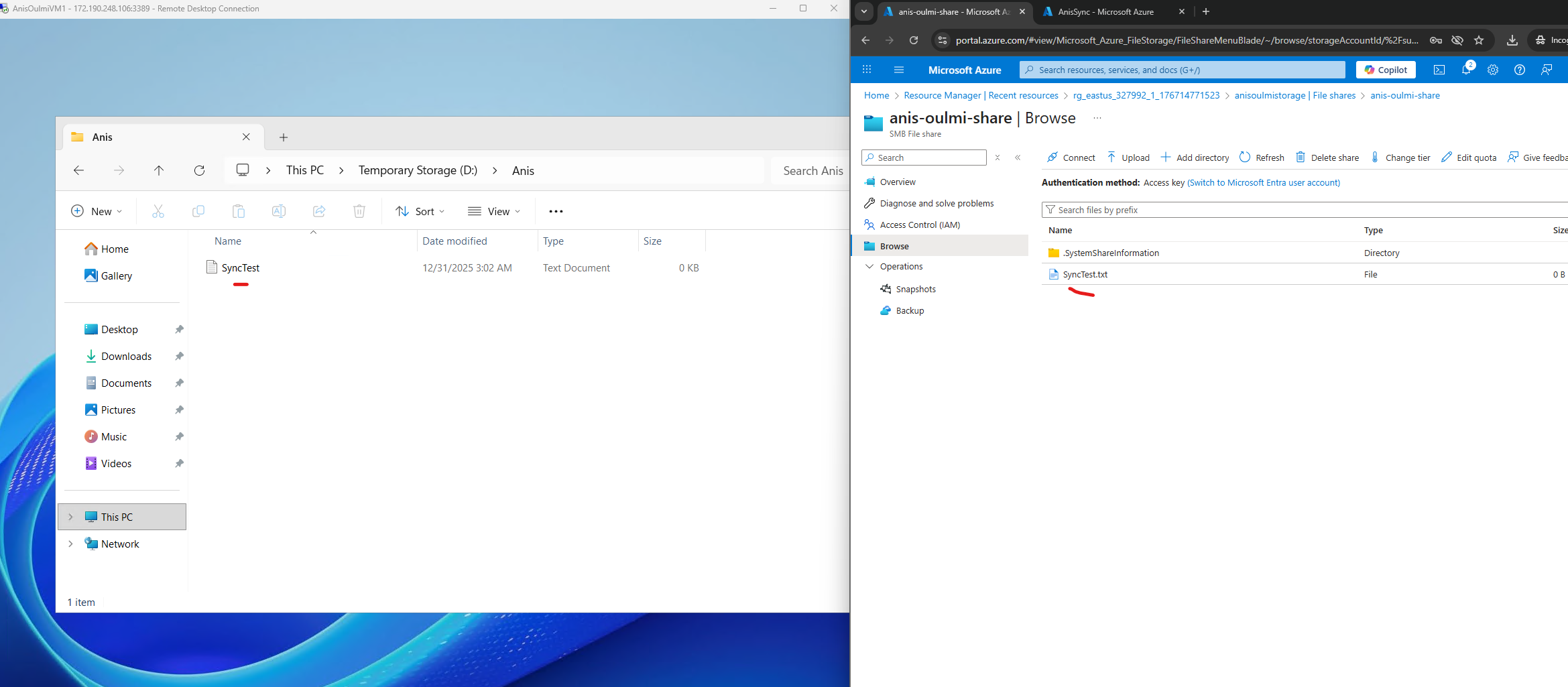
Finally, I created a server endpoint within the Sync Group. This endpoint links a specific folder on the Windows server to the cloud file share, enabling the actual file synchronization. I specified the folder path on the server, enabled cloud tiering to optimize storage usage, and configured policies for data retention and free space. After a short setup, the endpoint was active, indicated by a green checkmark, showing that the server and cloud were successfully connected and ready to sync files.



**Task 10: Test the Sync Group with One Server Endpoint**

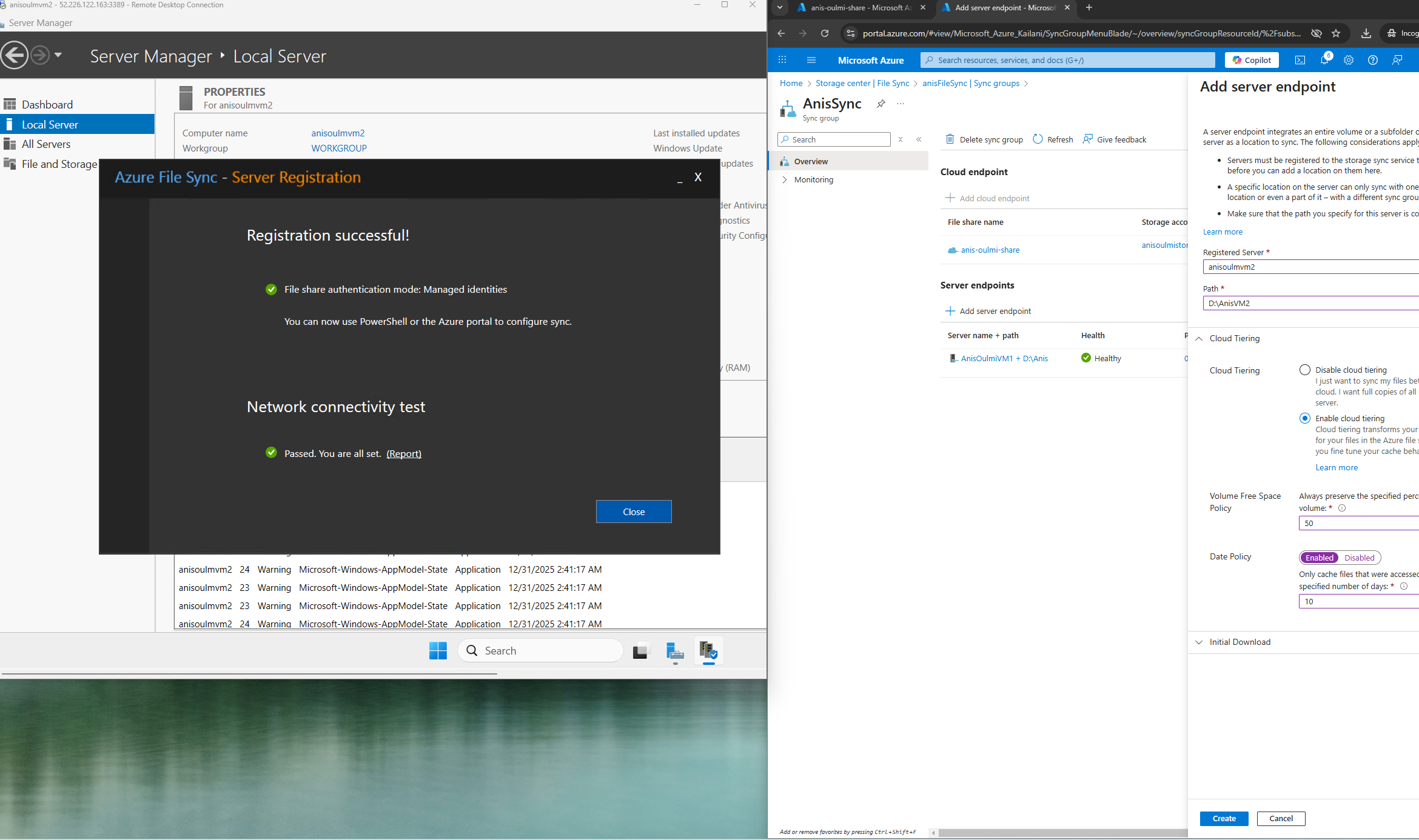
To verify that the synchronization was working, I tested the Sync Group with the first server endpoint. I navigated to the synchronized folder on the Windows server, created a new text file, and saved some data in it. Then, I checked the corresponding Azure file share in the portal and saw the file appear almost instantly. This confirmed that changes on the server were being reflected in the cloud in real time, demonstrating that the Azure File Sync setup was functioning as intended.

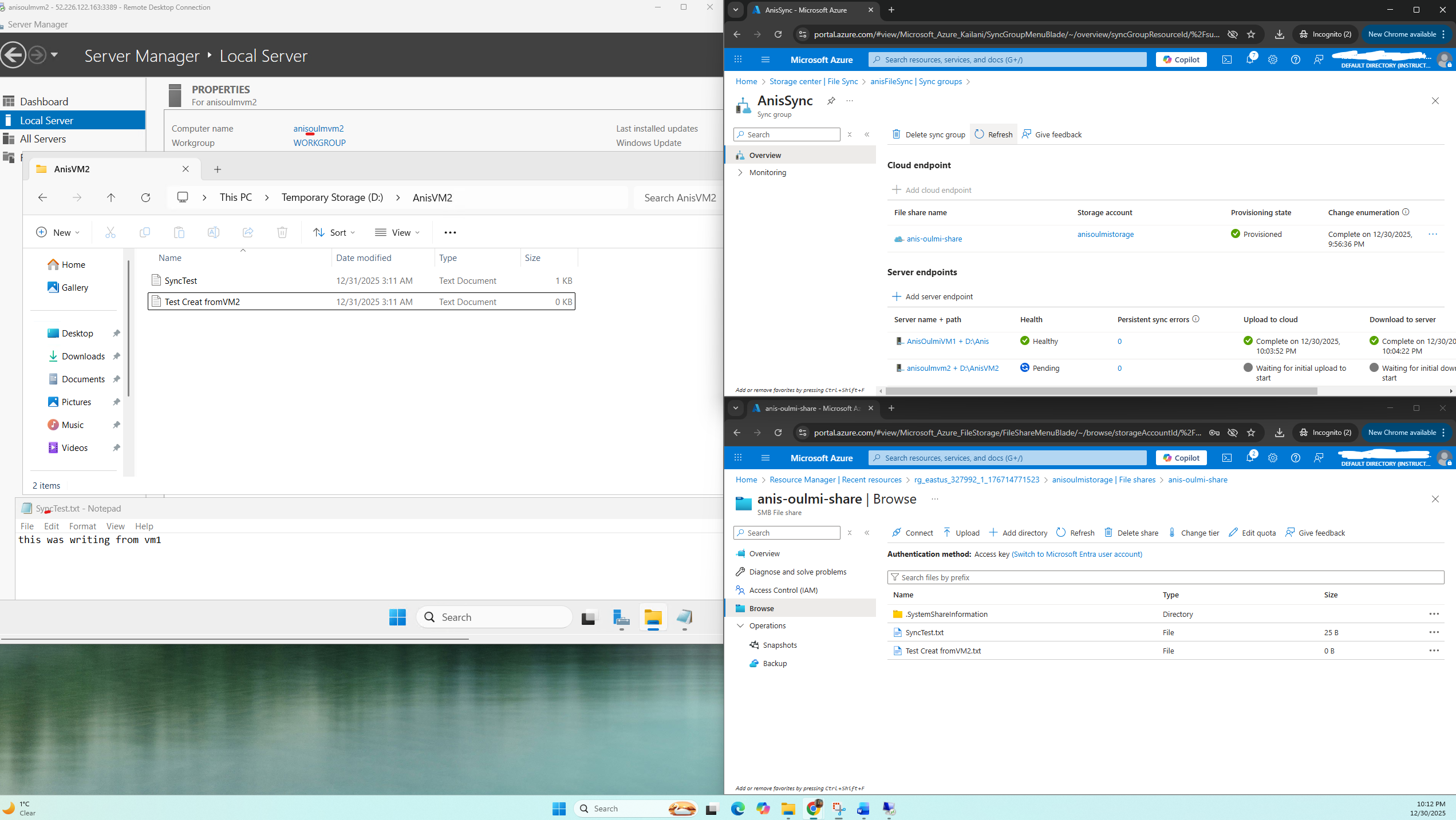


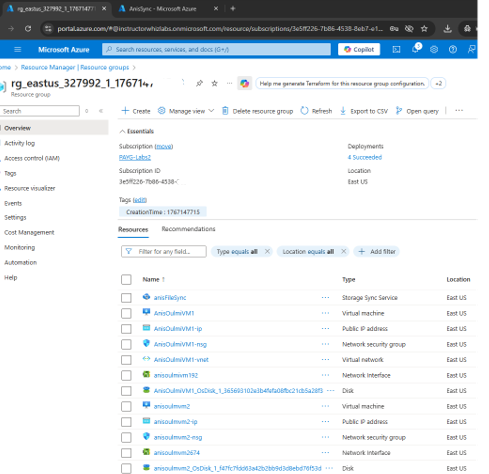


**Task 11: Test the Sync Group with Two Server Endpoints**

Finally, I tested the Sync Group with both Windows servers. I set up the second server, just like I did with VM1, installing the Azure File Sync agent, registering it with the Storage Sync Service, and creating a server endpoint linked to a dedicated folder. When I created a new file on VM2, it appeared in the Azure file share almost immediately. Then, checking VM1, I could see that the same file had synced there as well. This demonstrated that multiple servers could stay in sync through Azure File Sync, ensuring consistency across different endpoints connected to the same cloud storage.





As a final step, it’s important to clean up all the resources you created in Azure. Deleting the virtual machines, storage accounts, and sync services prevents any unexpected charges, helping you manage cloud costs responsibly.

In conclusion, this lab gave me hands-on experience with deploying and managing core Azure services. From creating storage accounts and file shares to setting up Windows servers and configuring Azure File Sync, I learned how to seamlessly connect on-premises-like servers to cloud storage. Testing with multiple server endpoints demonstrated the power of real-time synchronization, ensuring data consistency across environments.