FILE SHARING IN MULTI CLOUD PLATFORM

TEAM NAME: KICKSTARTERS

TEAM MENTORS:

BOBBY SIR

SURYA ASHOK SIR

TEAM MEMBERS:

B.Poorna Gana Venkatesh

N.Suresh

Synopsis:

1. WHAT IS MEANT BY FILE SHARING?
2. WHAT IS MEANT BY FILE SHARING IN MULTI CLOUD PLATFORM?
3. WHAT ARE THE USES AND REAL TIME APPLICATION OF FILE SHARE IN MULTI CLOUD PLATFORM?
4. STEPS OF CONFIGURATION
5. PROOF OF CONCEPT
6. REPOSITORY DETAILS
7. OUTCOMES
8. CONCLUSION

PURPOSE:

The purpose of file sharing is to allow individuals or organizations to share files and information with others in a convenient, efficient, and secure manner. File sharing enables users to transfer large files or documents that may be too large to send via email or other traditional methods of file transfer. It also allows multiple users to collaborate on a single document or project by enabling them to access and edit files in real-time. File sharing is commonly used in various settings, such as businesses, schools, research institutions, and personal use, to facilitate the sharing of data and information among different individuals or groups.

TECHNOLOGIES USED:

- AWS
- AZURE
- MOBAXTERM

SERVICES USED:

- EC2
- VPC
- EFS
- SECURITY GROUPS
- VIRTUAL MACHINE
- STORAGE ACCOUNTS
- VIRTUAL NETWORK
- RESOURCE GROUP

I. OBJECTIVE: To share files securely from one cloud platform to the other cloud platform.

WHAT IS MEANT BY FILE SHARING?

File sharing refers to the practice of allowing access to digital files, such as documents, images, videos, music, and software, to other users over a network or the Internet. It involves the transfer of files from one computer or device to another, either for collaborative purposes or for personal use.

WHAT IS MEANT BY FILE SHARING IN MULTI CLOUD PLATFORM?

In a multi-cloud platform, file sharing refers to the practice of allowing users to share files across multiple cloud services. This means that users can access and share files stored in different cloud platforms, such as Google Drive, Dropbox, and Amazon Web Services, without the need to transfer them between platforms manually.

WHAT ARE THE USES AND REAL TIME APPLICATION OF FILE SHARE IN MULTI CLOUD PLATFORM?

1. Collaboration: Multi-cloud file sharing can be used for collaboration, enabling teams to work on shared documents, images, videos, and other files simultaneously, regardless of which cloud platform they are stored in.

2. Data Backup and Recovery: Multi-cloud file sharing can be used for data backup and recovery, allowing businesses to store copies of important files in multiple cloud platforms to ensure business continuity in the event of a disaster.

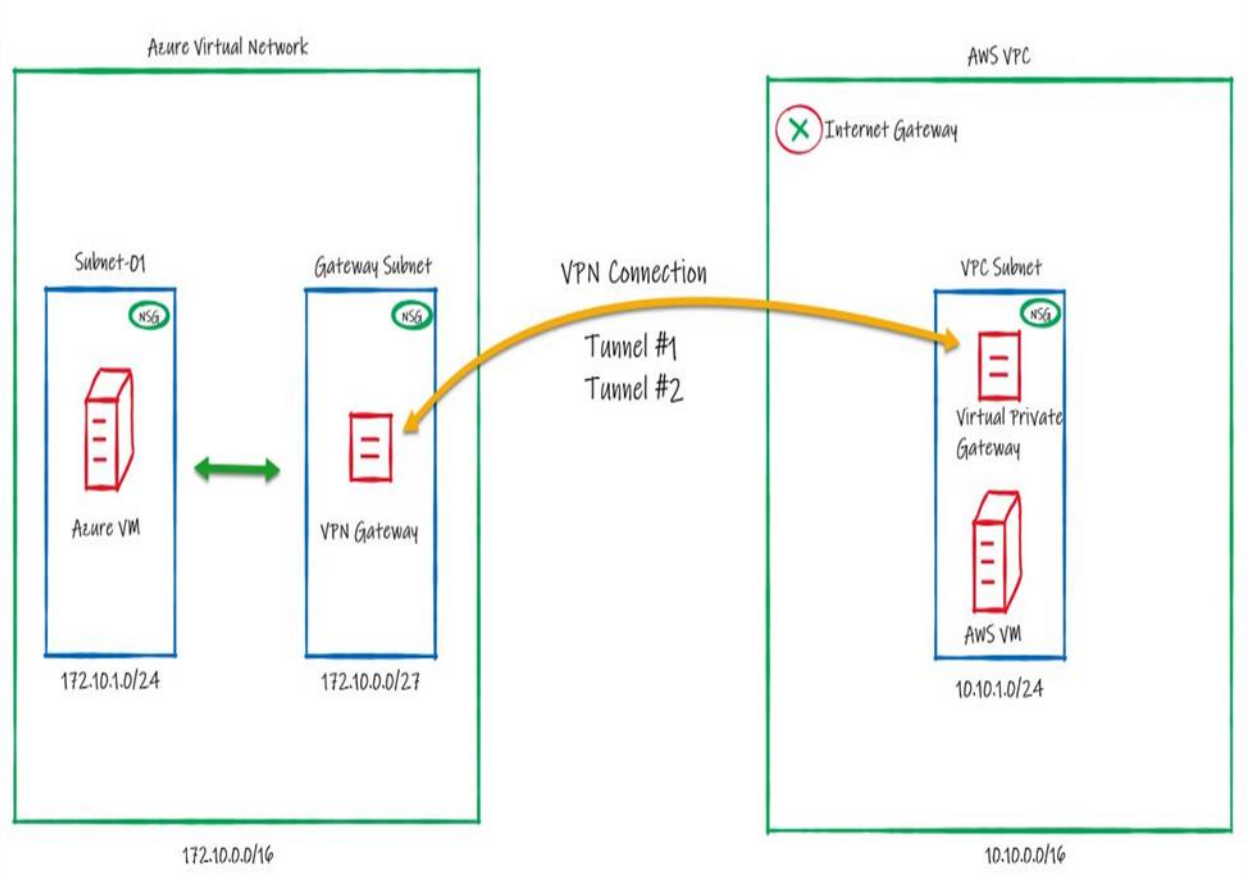
3. Cost savings: Multi-cloud file sharing can help businesses save costs by using different cloud platforms based on their specific needs, such as lower-cost storage options for less frequently accessed files, while using more expensive options for mission-critical files.

4. Data Governance: Multi-cloud file sharing can help businesses improve data governance by providing greater control and visibility over data sharing and access across different cloud platforms, ensuring compliance with regulations and data privacy laws.

5. Scalability: Multi-cloud file sharing can provide scalability by allowing businesses to adjust their cloud storage needs as per their requirements, scaling up or down as needed, to meet the changing needs of the business.

6. Improved Security: Multi-cloud file sharing can provide an added layer of security by allowing businesses to use different security features from multiple cloud providers, such as encryption and access controls, to protect their data from cyber threats.

II. FRAMEWORK:



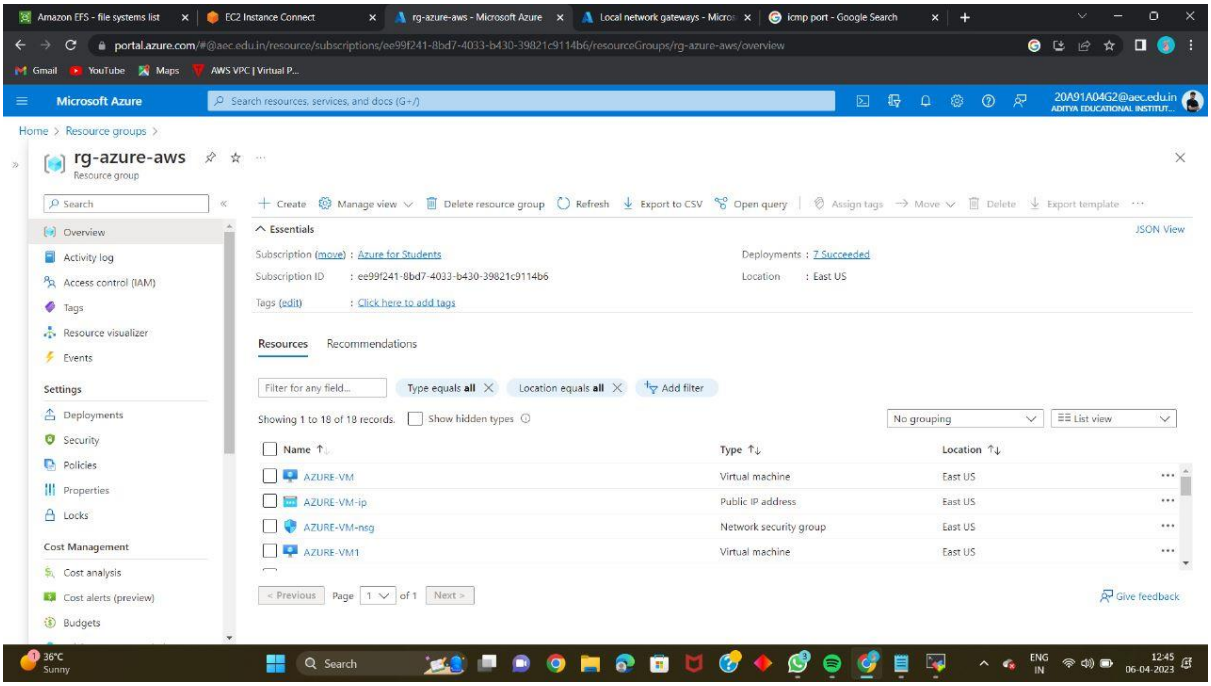
III. STEPS OF CONFIGURATION:

Step 1: Configuring Azure2

1. Create a resource group on Azure to deploy the resources on that:

   Resource Group Name: rg-azure-aws

   Region: East-US



2. Create Virtual Network

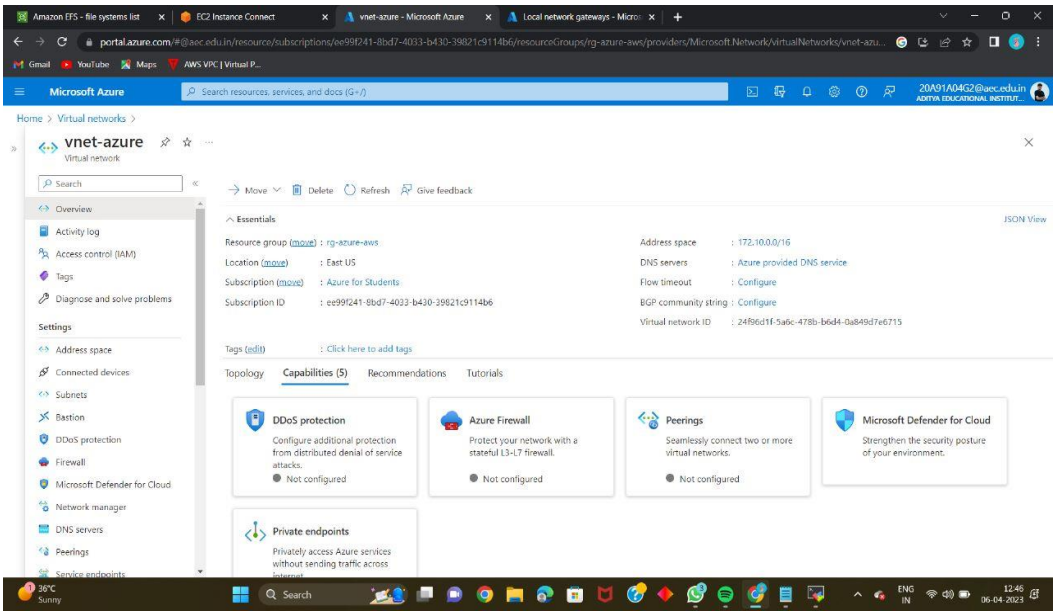
Resource Group Name: rg-azure-aws
Region: East-US

VNet Name: vnet-azure

VNet IPv4 Address Space: 172.10.0.0/16

Subnet Name: subnet-01

Subnet IPv4 Address Space: 172.10.1.0/24



3. Create the VPN Gateway

   VPN Gateway Name: vpn-azure-aws

   Region: East-US

   Gateway Type: VPN

   SKU: VpnGw1

   Generation: Generation 1

   Virtual Network: vnet-azure

   Public IP Address: pip-vpn-azure-aws

   Public IP Address Type: Basic

   Assignment: Dynamic

   Enable active-active mode: Disabled

   Configure BGP: Disabled

Step 2: Configuring AWS

4. Create the Virtual Private Cloud (VPC) in AWS

   Name: my-vpc-01

   IPv4 CIDR: 10.10.0.0/16

5. Create a subnet inside the VPC (Virtual Network)

   Name: my-subnet-01

   VPC Name: my-vpc-01

   VPC IPv4 CIDR: 10.10.0.0/16

   IPv4 CIDR: 10.10.1.0/24

6. Create a customer gateway pointing to the Public IP Address of Azure VPN Gateway

   IP address: Public IP Address of Azure VPN Gateway

   Rest keep everything as default

7. Create the Virtual Private Gateway then attach to the VPC

   Name: vpg-aws-azure

8. Create a site-to-site VPN Connection

   Name: vpn-aws-azure

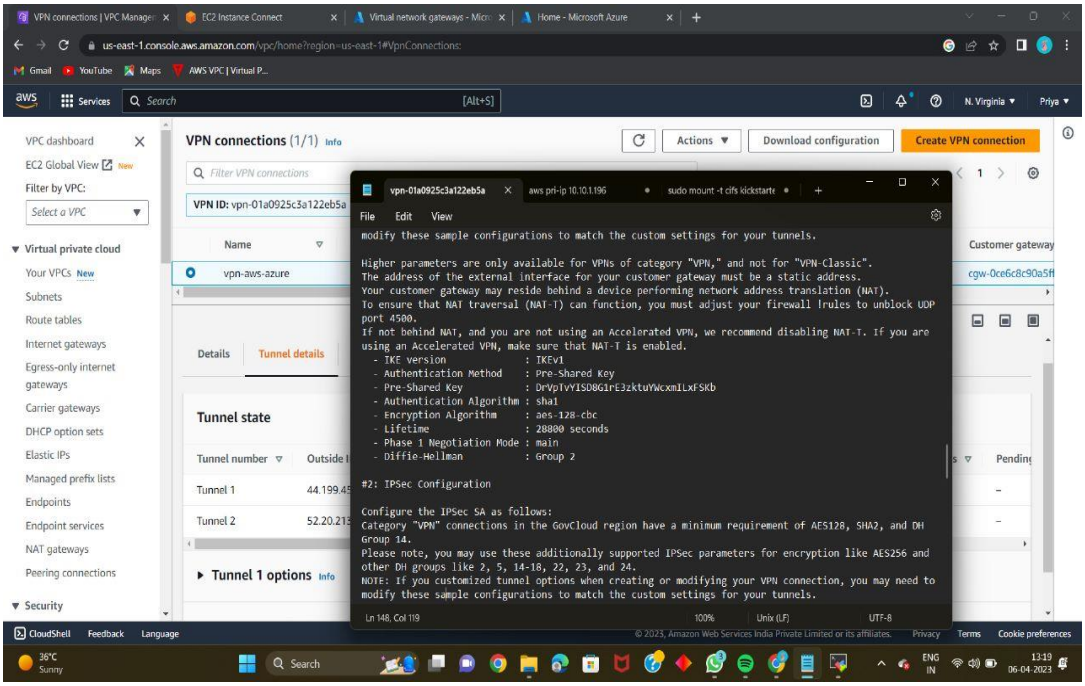
   Target gateway type: Virtual private gateway (Select your Virtual private gateway created in 7)

   Customer gateway: Existing (Select your VCustomer gateway created in 6)

   Routing options: Static

   Static IP prefixes: 172.10.1.0/24

   Leave rest of them as default



9. Download the configuration file

   Vendor: Generic

   Platform: Generic

   Software: Vendor Agnostic

In this configuration file you will note that there are the Shared Keys and the Public Ip Address for each of one of the two IPSec tunnels created by AWS.

Step 3: Connecting Azure and AWS

10. Create the Local Network Gateway in Azure

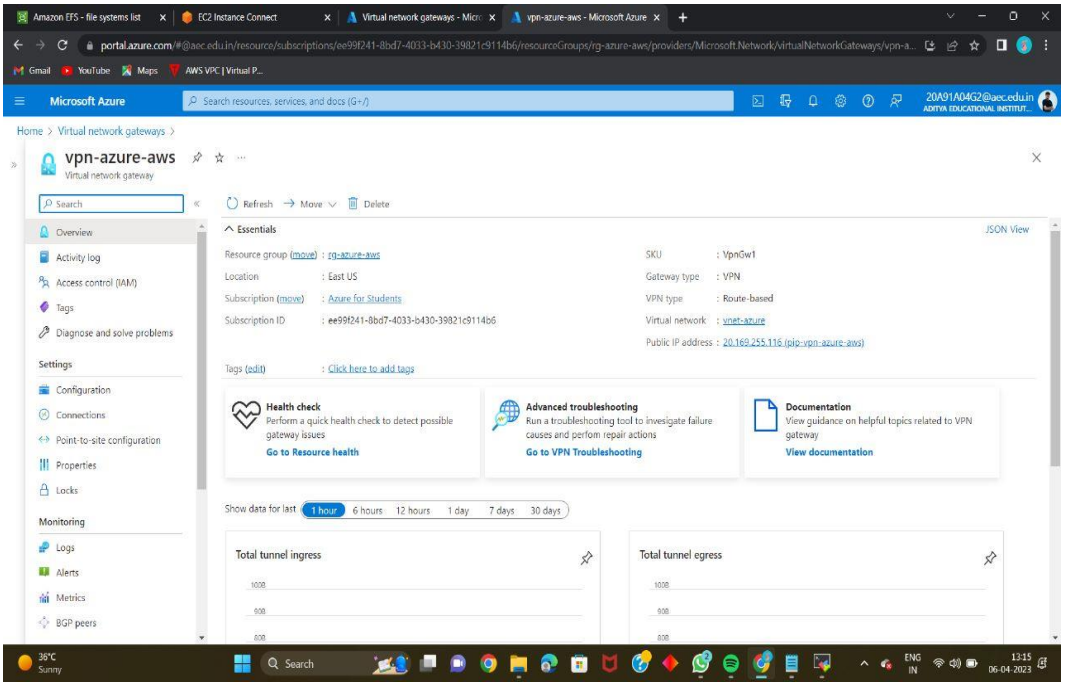
Name: lng-azure-aws

Resource Group Name: rg-azure-aws

Region: East-US

IP address: Get the Outside IP address from the configuration file downloaded in 9.

Address Space(s): 10.10.0.0/16



11. Create the connection on the Virtual Network Gateway in Azure

Name: connection-azure-aws

Connection Type: Site-to-Site

Local Network Gateway: Select the Local Network Gateway which you created in10.

Shared Key: Get the Shared Key from the configuration file downloaded in 9.

Wait till the Connection Status changes to - Connected

In the same way, check in AWS Console wheather the 1st tunnel of Virtual Private Gateway UP.

12. Create Internet Gateway and Attach it to VPC in AWS:

    Name: my-internet-gateway

13. Now let's edit the route table associated with our VPC

    Add the route to Azure subnet through the Virtual Private Gateway

    Destination: 172.10.1.0/24

    Target: Virtual Private Gateway that we created. also add,

    Destination: 0.0.0.0/0

    Target: Internet Gateway that we created in 12.

14. Create VMs in both Azure and AWS and Test the connection.

    Create security groups and Efs added to aws ec2 server.

    Create resource groups and file shares added to the ubuntu linux.

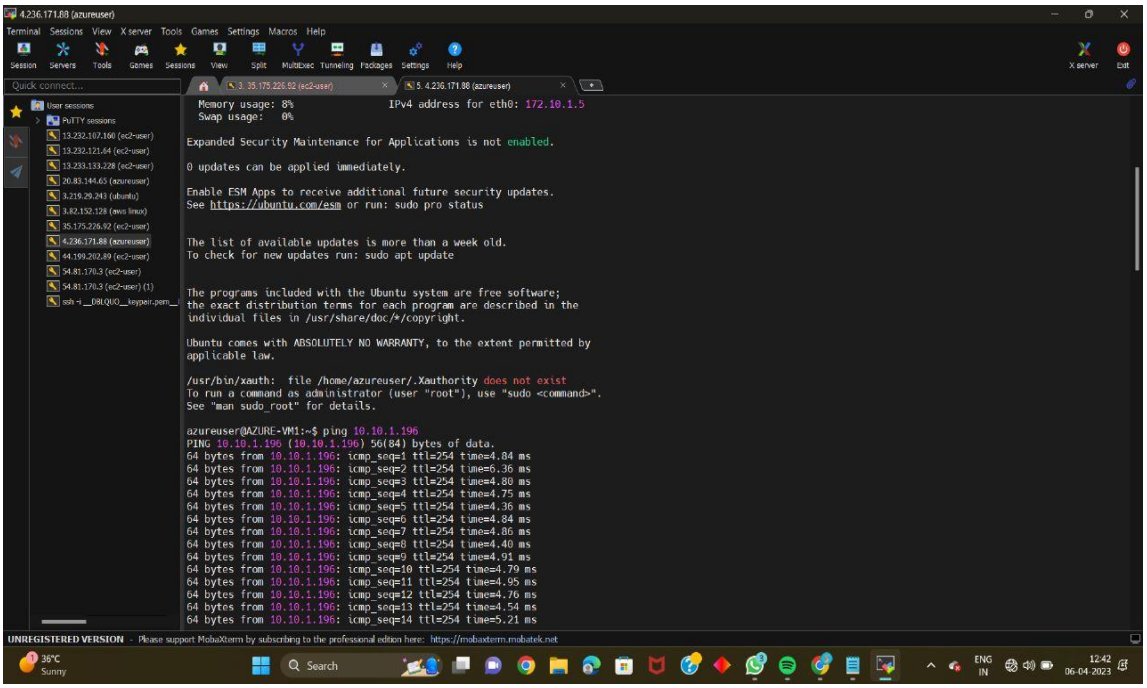
    Ping the both aws ec2 and azure virtual machine.

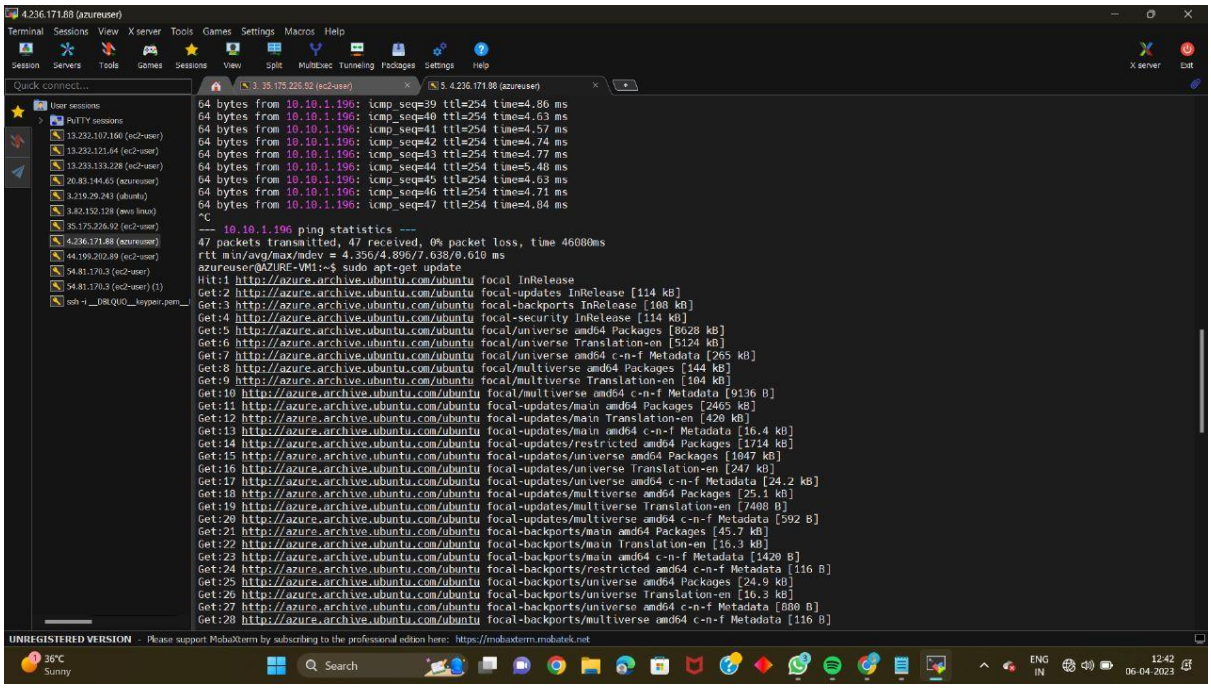
15. Commands used for Sharing the files from Azure to aws
    - sudo yum update
    - sudo yum install cifs-utils
    - sudo mkdir /mnt/azure_mount_point

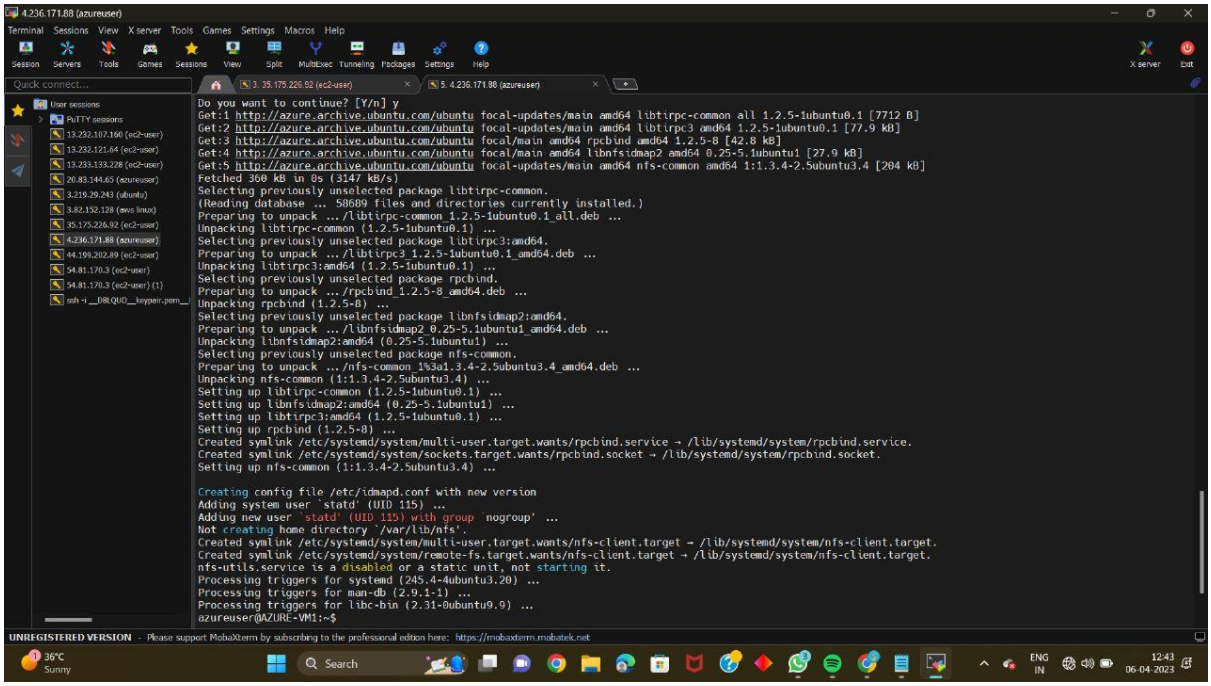
    sudo mount -t cifs //<storage-account-name>.file.core.windows.net/<file-share-name> /<mount-point> -o vers=3.0,username=<storage-account-name>,password=<storage-account-access-key>,dir_mode=0777,file_mode=0777

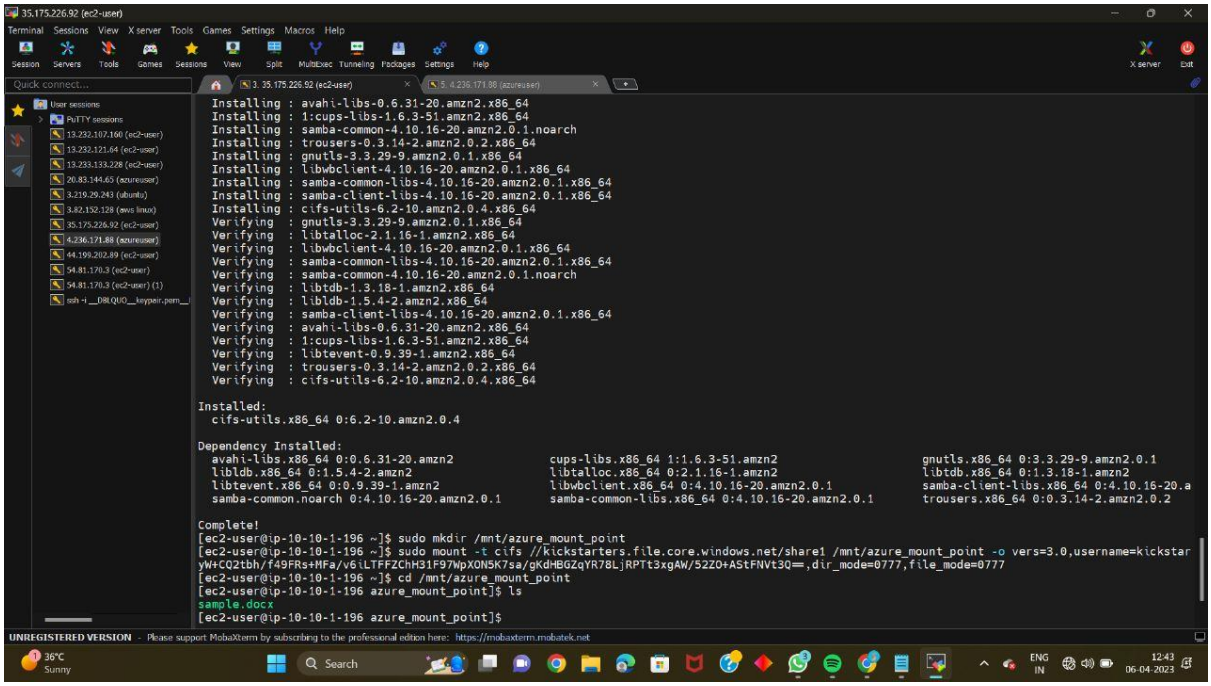
    - cd /mnt/azure_mount_point
    - ls

PROOF OF CONCEPT:






4.236.171.88 (azureuser)
Do you want to continue? [Y/n] y
Get:1 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 libtirpc-common all 1.2.5-1ubuntu0.1 [7712 B]
Get:2 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 libtirpc3 amd64 1.2.5-1ubuntu0.1 [77.9 kB]
Get:3 http://azure.archive.ubuntu.com/ubuntu focal/main amd64 rpcbind amd64 1.2.5-8 [42.8 kB]
Get:4 http://azure.archive.ubuntu.com/ubuntu focal/main amd64 libnfsidmap2 amd64 0.25-5.1ubuntu1 [27.9 kB]
Get:5 http://azure.archive.ubuntu.com/ubuntu focal-updates/main amd64 nfs-common amd64 1:1.3.4-2.5ubuntu3.4 [204 kB]
Fetched 360 kB in 0s (3147 kB/s)
Selecting previously unselected package libtirpc-common.
(Reading database ... 50609 files and directories currently installed.)
Preparing to unpack .../libtirpc-common_1.2.5-1ubuntu0.1_all.deb ...
Unpacking libtirpc-common (1.2.5-1ubuntu0.1) ...
Selecting previously unselected package libtirpc3:amd64.
Preparing to unpack .../libtirpc3_1.2.5-1ubuntu0.1_amd64.deb ...
Unpacking libtirpc3:amd64 (1.2.5-1ubuntu0.1) ...
Selecting previously unselected package rpcbind.
Preparing to unpack .../rpcbind_1.2.5-8_amd64.deb ...
Unpacking rpcbind (1.2.5-8) ...
Selecting previously unselected package libnfsidmap2:amd64.
Preparing to unpack .../libnfsidmap2_0.25-5.1ubuntu1_amd64.deb ...
Unpacking libnfsidmap2:amd64 (0.25-5.1ubuntu1) ...
Selecting previously unselected package nfs-common.
Preparing to unpack .../nfs-common_1%3a1.3.4-2.5ubuntu3.4_amd64.deb ...
Unpacking nfs-common (1:1.3.4-2.5ubuntu3.4) ...
Setting up libtirpc-common (1.2.5-1ubuntu0.1) ...
Setting up libnfsidmap2:amd64 (0.25-5.1ubuntu1) ...
Setting up libtirpc3:amd64 (1.2.5-1ubuntu0.1) ...
Setting up rpcbind (1.2.5-8) ...
Created symlink /etc/systemd/system/multi-user.target.wants/rpcbind.service → /lib/systemd/system/rpcbind.service.
Created symlink /etc/systemd/system/sockets.target.wants/rpcbind.socket → /lib/systemd/system/rpcbind.socket.
Setting up nfs-common (1:1.3.4-2.5ubuntu3.4) ...
Creating config file /etc/idmapd.conf with new version
Adding system user `statd' (UID 115) ...
Adding new user `statd' (UID 115) with group `nogroup' ...
Not creating home directory `/var/lib/nfs'.
Created symlink /etc/systemd/system/multi-user.target.wants/nfs-client.target → /lib/systemd/system/nfs-client.target.
Created symlink /etc/systemd/system/remote-fs.target.wants/nfs-client.target → /lib/systemd/system/nfs-client.target.
nfs-utils.service is a disabled or a static unit, not starting it.
Processing triggers for systemd (245.4-4ubuntu3.20) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu9.9) ...
azureuser@AZURE-VM1:~$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net


35.175.226.92 (ec2-user)
Installing : avahi-libs-0.6.31-20.amzn2.x86_64
Installing : 1:cups-libs-1.6.3-51.amzn2.x86_64
Installing : samba-common-4.10.16-20.amzn2.0.1.noarch
Installing : trousers-0.3.14-2.amzn2.0.2.x86_64
Installing : gnutls-3.3.29-9.amzn2.0.1.x86_64
Installing : libwbclient-4.10.16-20.amzn2.0.1.x86_64
Installing : samba-common-libs-4.10.16-20.amzn2.0.1.x86_64
Installing : samba-client-libs-4.10.16-20.amzn2.0.1.x86_64
Installing : cifs-utils-6.2-10.amzn2.0.4.x86_64
Verifying : gnutls-3.3.29-9.amzn2.0.1.x86_64
Verifying : libtalloc-2.1.16-1.amzn2.x86_64
Verifying : libwbclient-4.10.16-20.amzn2.0.1.x86_64
Verifying : samba-common-libs-4.10.16-20.amzn2.0.1.x86_64
Verifying : samba-common-4.10.16-20.amzn2.0.1.noarch
Verifying : libtdb-1.3.18-1.amzn2.x86_64
Verifying : libldb-1.5.4-2.amzn2.x86_64
Verifying : samba-client-libs-4.10.16-20.amzn2.0.1.x86_64
Verifying : avahi-libs-0.6.31-20.amzn2.x86_64
Verifying : 1:cups-libs-1.6.3-51.amzn2.x86_64
Verifying : libtevent-0.9.39-1.amzn2.x86_64
Verifying : trousers-0.3.14-2.amzn2.0.2.x86_64
Verifying : cifs-utils-6.2-10.amzn2.0.4.x86_64
Installed:
cifs-utils.x86_64 0:6.2-10.amzn2.0.4
Dependency Installed:
avahi-libs.x86_64 0:0.6.31-20.amzn2
cups-libs.x86_64 1:1.6.3-51.amzn2
gnutls.x86_64 0:3.3.29-9.amzn2.0.1
libldb.x86_64 0:1.5.4-2.amzn2
libtalloc.x86_64 0:2.1.16-1.amzn2
libtdb.x86_64 0:1.3.18-1.amzn2
libtevent.x86_64 0:0.9.39-1.amzn2
libwbclient.x86_64 0:4.10.16-20.amzn2.0.1
samba-client-libs.x86_64 0:4.10.16-20.a
samba-common.noarch 0:4.10.16-20.amzn2.0.1
samba-common-libs.x86_64 0:4.10.16-20.amzn2.0.1
trousers.x86_64 0:0.3.14-2.amzn2.0.2
Complete!
[ec2-user@ip-10-10-1-196 ~]$ sudo mkdir /mnt/azure_mount_point
[ec2-user@ip-10-10-1-196 ~]$ sudo mount -t cifs //kickstarters.file.core.windows.net/share1 /mnt/azure_mount_point -o vers=3.0,username=kickstar
yW+CQ2tbh/f49FRs+MFa/v6iLTFFZChH31F97WpXON5K7sa/gKdHBGZqYR78LjRPTt3xgAW/52ZO+AStFNVt3Q==,dir_mode=0777,file_mode=0777
[ec2-user@ip-10-10-1-196 ~]$ cd /mnt/azure_mount_point
[ec2-user@ip-10-10-1-196 azure_mount_point]$ ls
sample.docx
[ec2-user@ip-10-10-1-196 azure_mount_point]$
UNREGISTERED VERSION - Please support MobaXterm by subscribing to the professional edition here: https://mobaxterm.mobatek.net

Repository Details:

- Cloud storage accounts:

  Multi-cloud file sharing typically involves using cloud storage accounts on multiple cloud platforms. For example, an organization may use AWS S3 for storage on one project, and Azure Blob Storage for another project.

- Access controls:

  Access controls are typically used to restrict access to files stored in cloud storage accounts. These controls can be set up using tools such as AWS IAM or Azure Active Directory.

- File transfer protocols:

  File transfer protocols such as NFS, SMB, FTP, or SFTP may be used to transfer files between different cloud storage accounts. These protocols can be used to move files from one cloud storage account to another, or to synchronize files between cloud storage accounts.

- File synchronization tools:

  File synchronization tools such as Rclone, Syncthing, or GoodSync can be used to keep files synchronized across multiple cloud storage accounts. These tools can be set up to automatically sync files between different cloud storage accounts, ensuring that the most up-to-date files are always available.

- Monitoring and reporting:

  Monitoring and reporting tools such as CloudWatch, Azure Monitor, or Grafana can be used to monitor the performance of file sharing in a multi-cloud platform. These tools can provide real-time monitoring of file transfer activities, as well as historical reports and analytics.

Outcomes:

- Increased flexibility:

  File sharing in a multi-cloud platform can allow organizations to leverage the strengths of multiple cloud providers, which can provide greater flexibility in terms of selecting the right cloud platform for different use cases.

- Improved agility:

  Multi-cloud file sharing can allow organizations to quickly and easily move data between different cloud providers, which can improve agility in terms of responding to changing business needs.

- Cost savings:

  File sharing in a multi-cloud platform can help organizations to avoid vendor lock-in and negotiate better pricing with different cloud providers, which can result in cost savings.

- Improved data resilience:

  By storing data across multiple cloud providers, multi-cloud file sharing can provide improved data resilience in case of outages or other issues with any particular cloud provider.

- Enhanced security:

  Multi-cloud file sharing can allow organizations to implement a defense-in-depth security strategy, which involves using multiple layers of security controls to protect data.

CONCLUSION:

In conclusion, file sharing in a multi-cloud platform can provide many benefits to organizations, including increased flexibility, agility, cost savings, improved data resilience, and enhanced security. By leveraging multiple cloud providers, organizations can avoid vendor lock-in and tailor their cloud environment to specific use cases. Additionally, multi-cloud file sharing can provide a more resilient and secure environment for data storage and transfer.

However, it is important to carefully plan and implement a multi-cloud file sharing strategy that takes into account the specific needs and requirements of the organization. This includes selecting the appropriate cloud providers and technologies, ensuring proper access controls and security measures are in place, and monitoring performance and usage to ensure optimal operation. With proper planning and execution, file sharing in a multi-cloud platform can provide significant benefits to organizations looking to optimize their cloud environment.