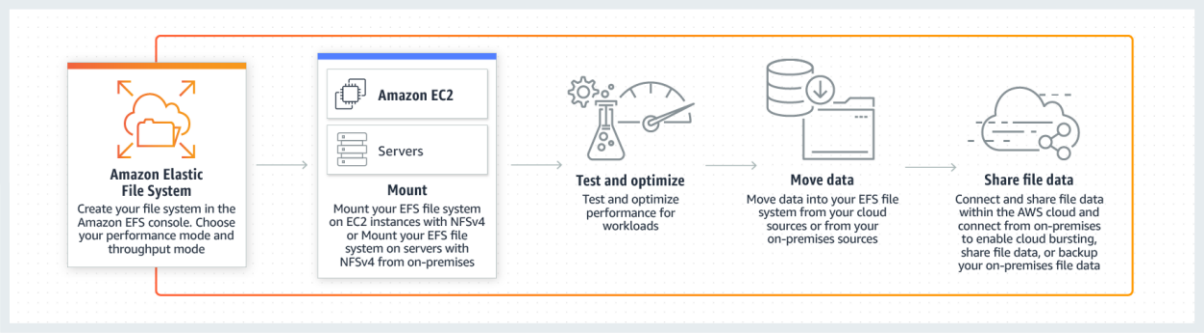
Getting Started with AWS - Amazon Elastic File System



Amazon Elastic File System (Amazon EFS) provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. It is built to scale on demand to petabytes without disrupting applications, growing and shrinking automatically as you add and remove files, eliminating the need to provision and manage capacity to accommodate growth.

This Scenario is for: **Amazon EC2 mount instructions (from local VPC)**

Step 1: Create your EC2 two AMI Linux instances (using default security groups) to check your Data Online with AWS DataSync

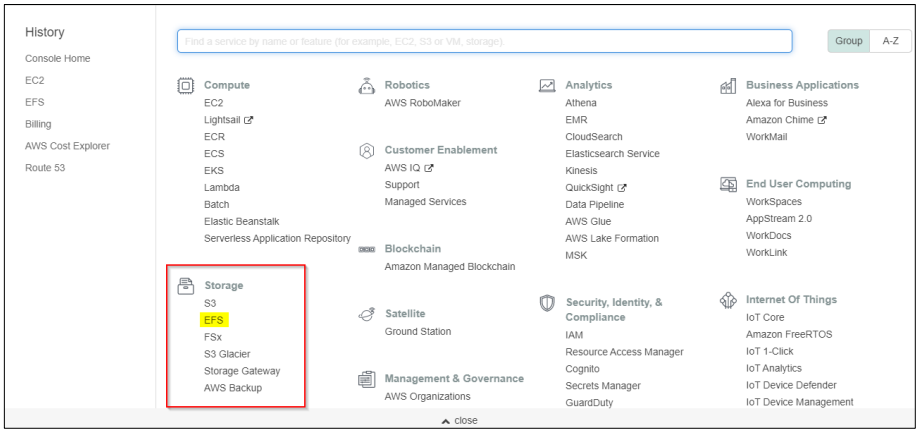
Step 2: Once your Instances are ready you can access using putty AMI Server 1 / AMI Server 2 and setup the below configuration:

Server 1
#sudo su
#yum update -y (To update the server packages)
#mkdir **efs** (create a "efs' directory to synch with EFS)

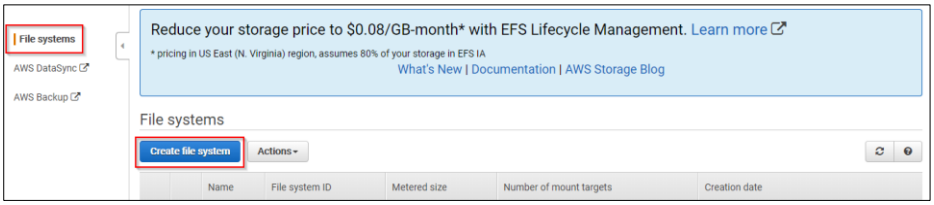
Server 2
#sudo su
#yum update -y (To update the server packages)
#mkdir **efs** (create a "efs' directory to synch with EFS)

Step 3: Click on EFS service in AWS to Create Your Amazon EFS File System.

(Please note that AWS EFS service is chargeable for $0.08/GB-Month*)



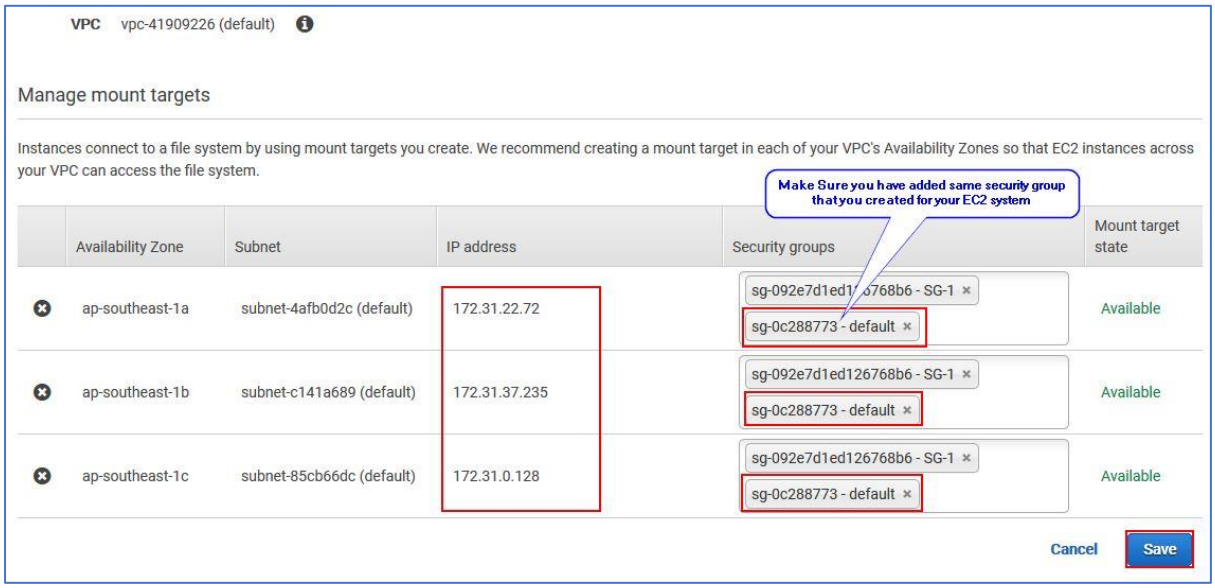
Step 4: Click on "Create File systems"



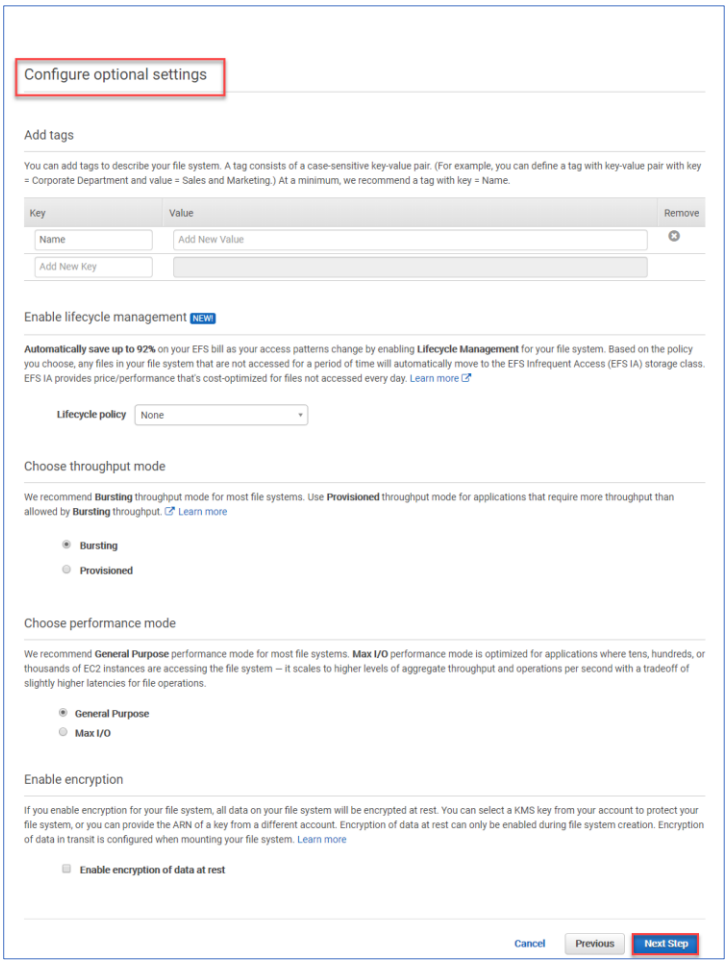
Step 5: Configure file system access click "Next"

Select the check boxes for all of the **Availability Zones**. Make sure that they all have the default subnets, automatic IP addresses, and the default security groups chosen. These are your mount targets. For more information, see Creating Mount Targets.

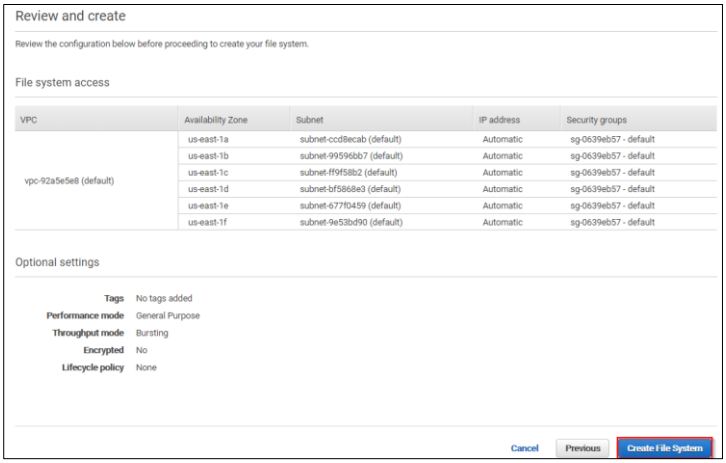
Make sure that you have to select the right Security Group where you would need to mount the EFS on that particular EC2 system.



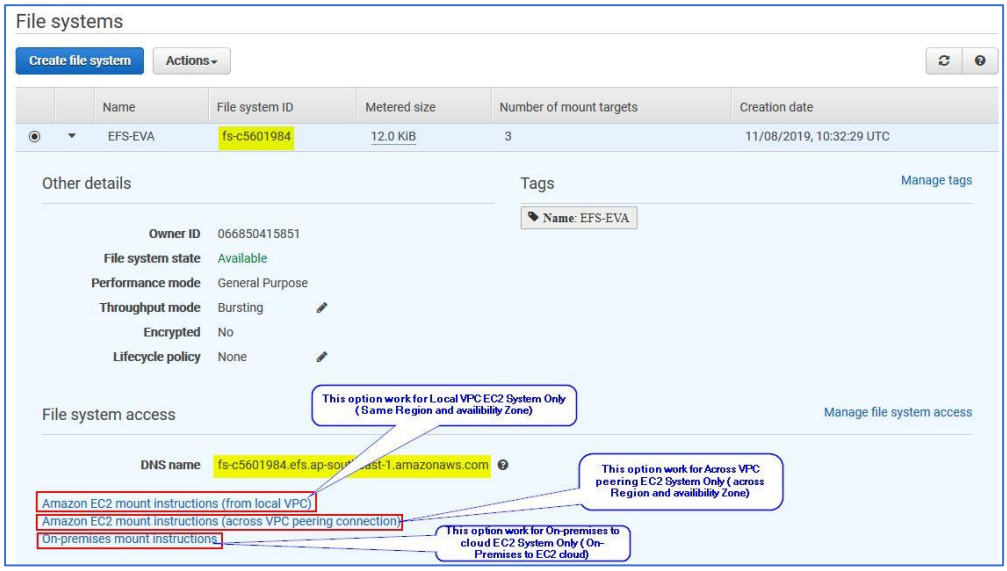
Step 6: Configure optional settings just click **"Next"**



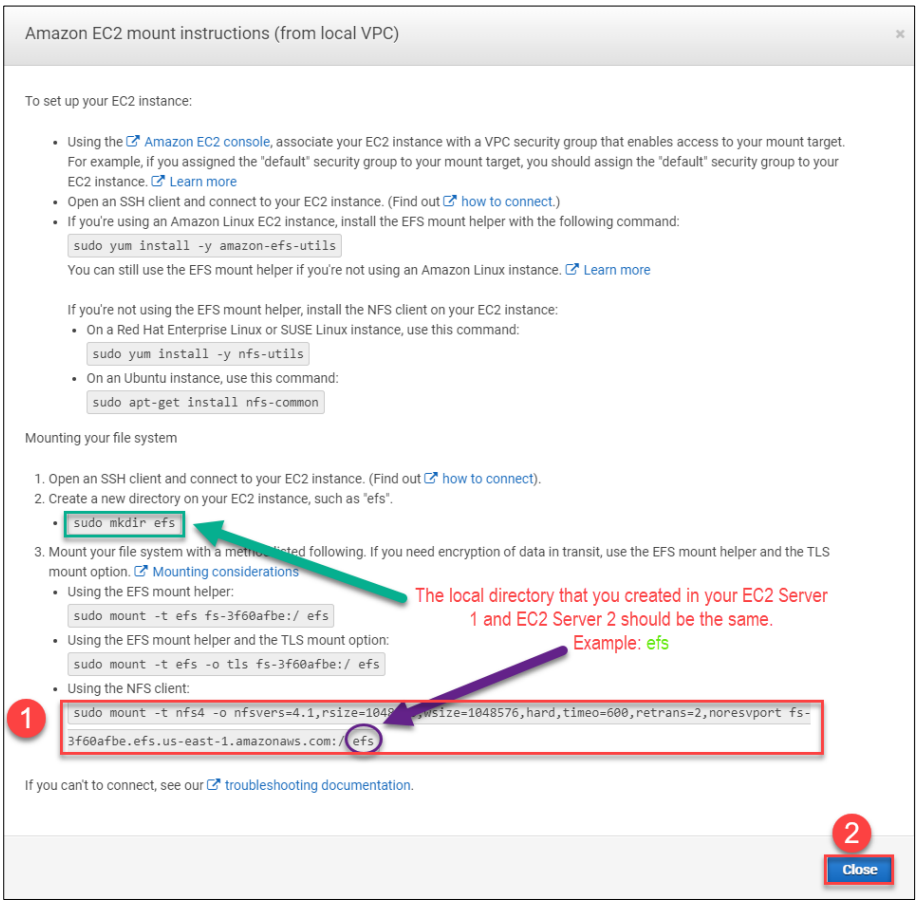
Step 7: Review the file system configuration, and then choose **"Create File System."**



Step 8: Click on **"File System ID"** and navigate settings for mount the Amazon EFS file system.



Step 9: Please connect to your Amazon EC2 instance **Server 1** and **server 2** and mount the Amazon EFS file system, **copy the command** and **mount target** for your Amazon EFS file system in your both the servers.



To update the /etc/fstab file in your EC2 instance

1. Connect to your EC2 instance, and open the `/etc/fstab` file in an editor.
2. Add the following line to the `/etc/fstab` file.

   ```
   fs-12345678:/ /mnt/efs efs defaults,_netdev 0 0
   ```

   Warning

   Use the `_netdev` option, used to identify network file systems, when mounting your file system automatically. If `_netdev` is missing, your EC2 instance might stop responding. This result is because network file systems need to be initialized after the compute instance starts its networking. For more information, see Automatic Mounting Fails and the Instance Is Unresponsive.

3. Save the changes to the file.

Your EC2 instance is now configured to mount the EFS file system whenever it restarts.

Note

If your Amazon EC2 instance needs to start regardless of the status of your mounted Amazon EFS file system, add the `nofail` option to your file system's entry in your `/etc/fstab` file.

The line of code you added to the /etc/fstab file does the following.

Field	Description
fs-12345678:/	The ID for your Amazon EFS file system. You can get this ID from the console or programmatically from the CLI or an AWS SDK.
/mnt/efs	The mount point for the EFS file system on your EC2 instance.
`efs`	The type of file system. When you're using the mount helper, this type is always `efs`.
`mount options`	Mount options for the file system. This is a comma-separated list of the following options: • `defaults` – This value tells the operating system to use the default mount options, which you can list after the file system has been mounted by viewing the output of the `mount` command. • `_netdev` – The value tells the operating system that the file system resides on a device that requires network access. This option prevents the instance from mounting the file system until the network has been enabled on the client. • You can replace `defaults` here with `tls` to enable encryption of data in transit.
`0`	A nonzero value indicates that the file system should be backed up by `dump`. For EFS, this value should be `0`.
`0`	The order in which `fsck` checks file systems at boot. For EFS file systems, this value should be `0` to indicate that `fsck` should not run at startup.

Step 10: AMI Server 1
[root@ip-172-31-85-194 ec2-user]# sudo mount -t nfs4 -o nfsvers=4.1,rsize=104857
6,wsize=1048576,hard,timeo=600,retrans=2,noresvport **fs-3f60afbe.efs.us-east-1.am**
azonaws.com:/ efs

```
Complete!
[root@ip-172-31-85-194 ec2-user]# mkdir efs
[root@ip-172-31-85-194 ec2-user]# sudo mount -t nfs4 -o nfsvers=4.1,rsize=104857
6,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-3f60afbe.efs.us-east-1.am
azonaws.com:/ efs
```

Step 11: AMI Server 2

[root@ip-172-31-89-235 ec2-user]# sudo mount -t nfs4 -o nfsvers=4.1,rsize=104857
6,wsize=1048576,hard,timeo=600,retrans=2,noresvport **fs-3f60afbe.efs.us-east-1.am**
azonaws.com:/ efs

Now our EC2 Server 1 and Server 2 instances is now configured to mount the EFS file system.

Step 12: Now you can create a directory or two files on Server 1 under "efs" directory

#touch abc pqr
#mkdir ethans

Step 13: Now you can check on EC2 Server 2 under "efs" directory If directory and files have been synced automatically from Server 1

FYI – See the output

Server 1

```
drwxr-xr-x 3 root root 6144 Oct 16 04:15 efs
[root@ip-172-31-85-194 ec2-user]# cd efs/
[root@ip-172-31-85-194 efs]# ll
total 16
-rw-r--r-- 1 root root    0 Oct 16 04:14 abc
drwxr-xr-x 2 root root 6144 Oct 16 04:15 ethans
-rw-r--r-- 1 root root    0 Oct 16 04:14 pqr
-rw-r--r-- 1 root root   14 Oct 16 04:15 temp.txt
[root@ip-172-31-85-194 efs]#
```

Server2

```
drwxr-xr-x 3 root root 6144 Oct 16 04:15 efs
[ec2-user@ip-172-31-84-96 ~]$ cd efs/
[ec2-user@ip-172-31-84-96 efs]$ ll
total 16
-rw-r--r-- 1 root root    0 Oct 16 04:14 abc
drwxr-xr-x 2 root root 6144 Oct 16 04:15 ethans
-rw-r--r-- 1 root root    0 Oct 16 04:14 pqr
-rw-r--r-- 1 root root   14 Oct 16 04:15 temp.txt
[ec2-user@ip-172-31-84-96 efs]$
```

Done: Transfer Files to Amazon EFS Using AWS DataSync Successfully

To transfer files from a source location to a destination location using AWS DataSync.
Now that we have created a functioning Amazon EFS file system, we can use AWS DataSync to transfer files from an existing file system to Amazon EFS. AWS DataSync is a data transfer service that simplifies, automates, and accelerates moving and replicating data between on-premises storage systems and AWS storage services over the internet or AWS Direct Connect. AWS DataSync can transfer our file data, and also file system metadata such as ownership, time stamps, and access permissions.
Done

Amazon EC2 mount instructions (across a VPC peering connection)

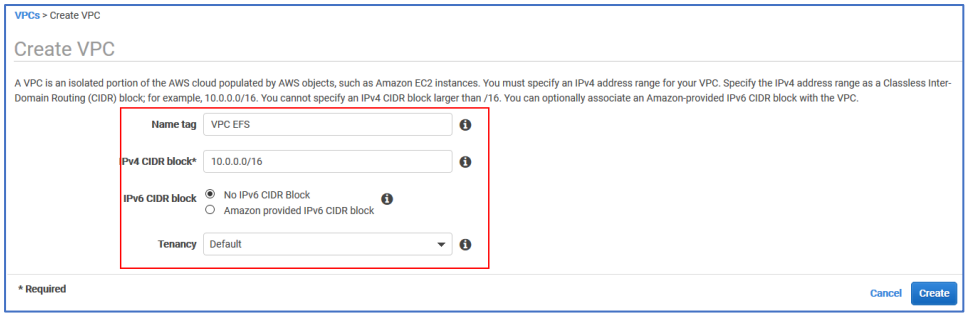
You can mount an EFS file system on an Amazon EC2 instance over a VPC peering connection. Learn more

You can use an Amazon EFS file system in one VPC based on the Amazon VPC service at a time. That is, you create mount targets in a VPC for your file system, and use those mount targets to provide access to the file system.

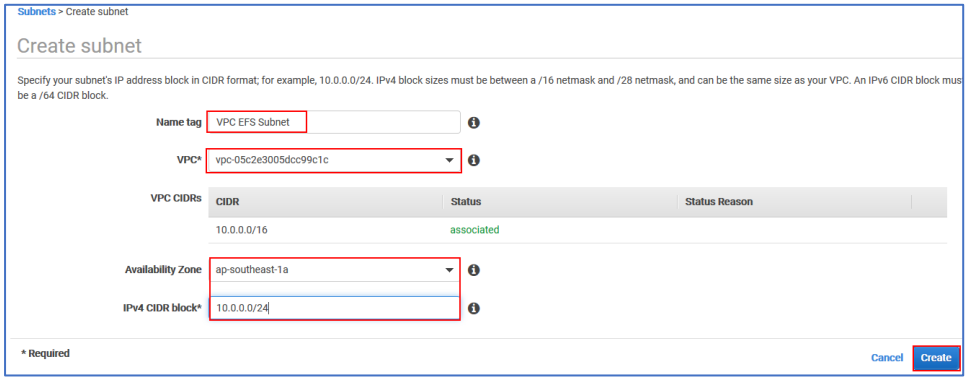
SIDE A - ACCOUNT ID: 786125941515 (EFS SYSTEM) SINGAPORE REGION VPC ID: VPC-05C2E3005DCC99C1C

SIDE B - ACCOUNT ID: 268776714427 (EC2 SYSTEM) SINGAPORE REGION

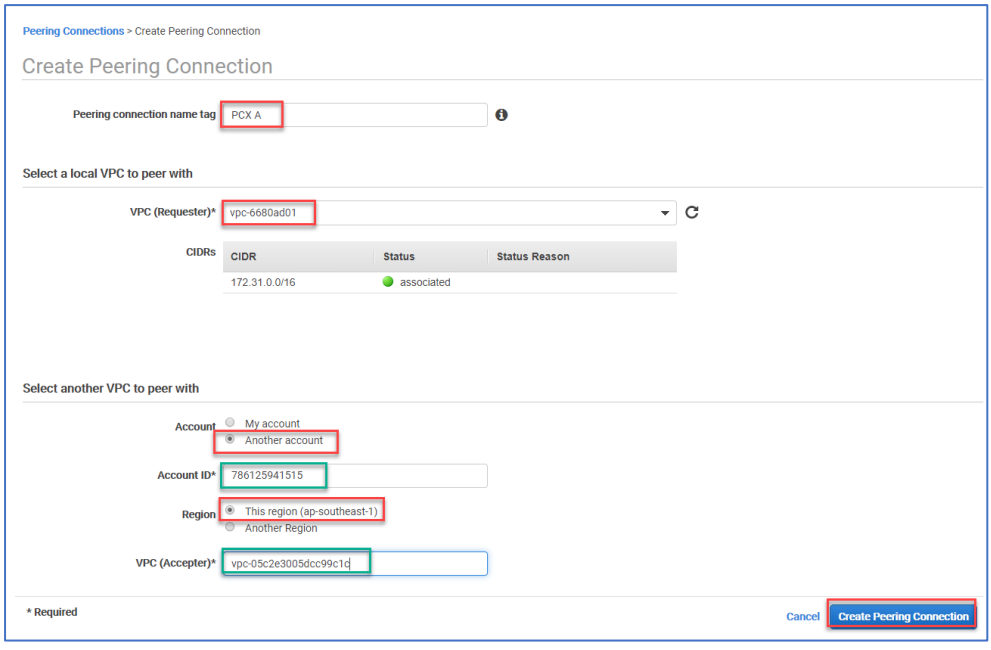
1. **Create a new VPC Side A ESF System. IPv4 CIDR (10.0.0.0/16)**



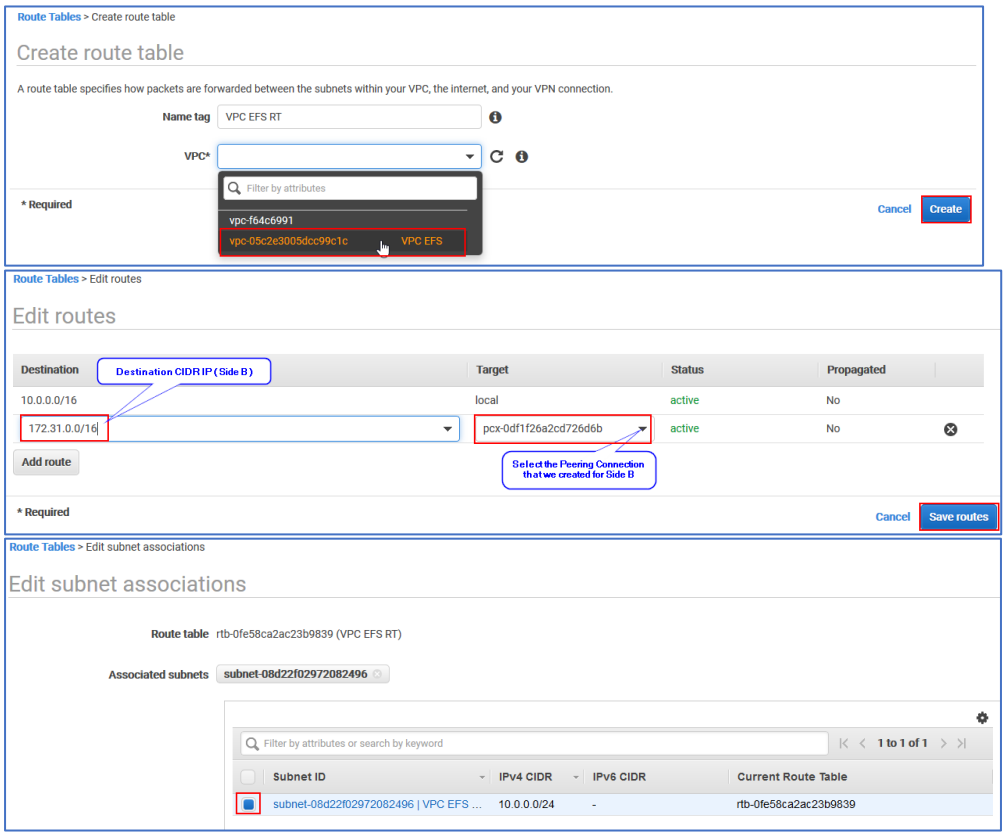
2. Create subnet and associated with VPC that I created above Side A.



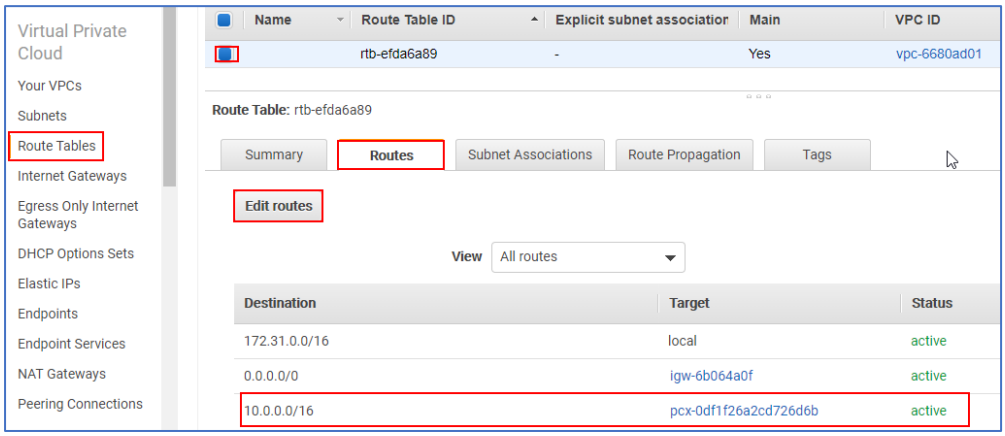
3. Go to **side B** and create a **VPC** Peering request as below: **Provide A/C ID & VPC ID A Side.**



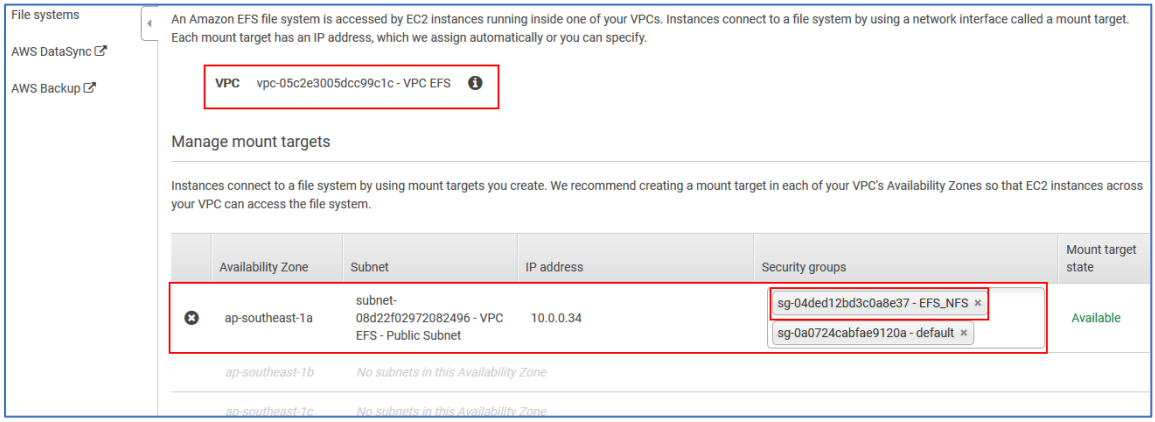
4. **Go To Side A 'Peering Connection and accept the request.**
5. **Go To Side A 'Route Table' and create a Route as below and edit the route as below**



Now Go To Side B and add the 'Route table' as below (We need to add the destination IP for side A) i.e 10.0.0.0/16



Now Go to **Side A** and Create a **new EFS adding** under new VPC that we created for EFS. Add Security group.



Now access EFS file sys from Side B EC2 system :

Access my EC2 Instance **Side B**

Create a new directory on your EC2 instance, such as "appserver_efs"

Mount your file system.

sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 10.0.0.34:/ appserver_efs

root@ip-172-31-17-127:/# sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2 10.0.0.34:/ appserver_efs

Done

How to set up auto rsync backups in AWS EC2 using ssh.

Rsync (Remote Sync) is a most commonly used command for copying and synchronizing files and directories remotely as well as locally in Linux/Unix systems. With the help of rsync command you can copy and synchronize your data remotely and locally across directories, across disks and networks, perform data backups and mirroring between two Linux machines.

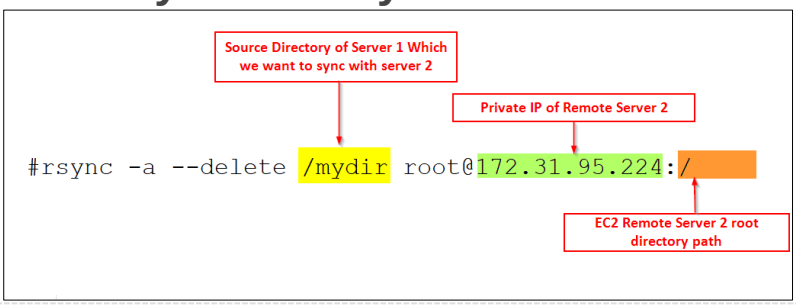
Step 1: Create your EC2 two AMI Linux instances to sync your Data One server to another server Online with rsync command.

Step 2: Server 1 configuration with the steps below
#rm -rf .ssh
#ssh-keygen -t rsa
#cd .ssh/
#vi id_rsa.pub (**Copy the ssh code and paste it over at the EC2 server 2**)
#cd
#ssh 172.31.95.224 (Private IP of your **server 2** Instance)
(Check the ssh permitted server 1 to access the server 2 using ssh putty)
If yes, then go ahead.
#Exit
#mkdir /mydir
#cd /mydir/
#touch abc pqr
#ls -lart

Step 3: Server 2 configuration with the steps below
#rm -rf .ssh
#ssh-keygen -t rsa
#cd .ssh/
#vi authorized_keys **(Paste the ssh code here which you copied from EC2 server 1**)
Save and Exit
#chmod 600 authorized_keys
#mkdir **/mydir**
#ls -lart

Step 4: Use the following command on Server 1 and will sync files on Server 1 to Server 2.
#rsync -a --delete /mydir **root@172.31.95.224:/**

Basic syntax of rsync command



Some common options used with rsync commands

-v : verbose
-r : copies data recursively (but don't preserve timestamps and permission while transferring data
-a : archive mode, archive mode allows copying files recursively and it also preserves symbolic links, file permissions, user & group ownerships and timestamps
-z : compress file data
-h : human-readable, output numbers in a human-readable format

Step 5: You can schedule "crontab" for auto sync
If you need the backup script run at a specific time daily at 7am, you'll have to manually create a cron job by issuing the command crontab -e and then adding a line such as:
00 07 * * * rsync -a --delete /mydir root@172.31.95.224:/

Output: Now you can check at Server 2 under "mydir" directory If files and directory have been synced automatically from Server 1

Done!