EBS EFS FSx

Comparison Table

Feature	EBS	EFS	FSx
Type	Block Storage	File Storage (NFS)	File Storage (Various)
Attachment	Single EC2 (or Multi-Attach)	Multiple EC2s	Multiple EC2s
Availability	Single AZ	Multi-AZ	Depends on type
Use Case	Boot volumes, Databases	Shared files, Web content	Windows, HPC, ML
Protocol	Block device	NFSv4	SMB, Lustre, NFS
Scalability	Fixed size (manual resize)	Auto-scaling	Depends on type
Pricing	Per GB provisioned	Per GB used	Per GB provisioned

Visual Overview

```
┌─────────────────────────────────────────────────────────────────────┐
│                        AWS Storage Services                          │
├─────────────────────────────────────────────────────────────────────┤
│        EBS          │        EFS          │          FSx            │
│   (Block Storage)   │   (File Storage)    │    (File Storage)       │
├─────────────────────┼─────────────────────┼─────────────────────────┤
│                     │                     │                         │
│   ┌─────────┐       │   ┌─────────┐       │   ┌─────────────────┐   │
│   │   EC2   │       │   │   EC2   │       │   │ FSx for Windows │   │
│   └────┬────┘       │   └────┬────┘       │   │ FSx for Lustre  │   │
│        │            │        │            │   │ FSx for NetApp  │   │
│   ┌────▼────┐       │   ┌────▼────┐       │   │ FSx for OpenZFS │   │
│   │   EBS   │       │   │   EFS   │       │   └─────────────────┘   │
│   │ Volume  │       │   │  (NFS)  │       │                         │
│   └─────────┘       │   └────┬────┘       │                         │
│                     │        │            │                         │
│   • Single AZ       │   ┌────▼────┐       │   • Specialized         │
│   • Like a HDD/SSD  │   │   EC2   │       │   • High Performance    │
│   • Boot volumes    │   └─────────┘       │   • Enterprise features │
│                     │                     │                         │
│                     │   • Multi-AZ        │                         │
│                     │   • Shared storage  │                         │
└─────────────────────┴─────────────────────┴─────────────────────────┘
```

Amazon EBS (Elastic Block Store)

What is EBS?

EBS is like a **virtual hard drive** that you attach to your EC2 instance. Think of it as plugging a USB drive or external SSD into your computer, but in the cloud.

Key Characteristics

- **Block-level storage** - Works like a physical hard drive
- **Single AZ** - Exists in one Availability Zone
- **Persistent** - Data survives EC2 stop/start (but not termination by default)
- **Attachable** - Can detach from one EC2 and attach to another (same AZ)
- **Snapshots** - Point-in-time backups stored in S3

EBS Volume Types

Type	Name	IOPS	Throughput	Use Case
gp3	General Purpose SSD	16,000	1,000 MB/s	Most workloads ⭐

gp2	General Purpose SSD	16,000	250 MB/s	Legacy, use gp3 instead
io2	Provisioned IOPS SSD	256,000	4,000 MB/s	Critical databases
io1	Provisioned IOPS SSD	64,000	1,000 MB/s	High-performance DBs
st1	Throughput HDD	500	500 MB/s	Big data, log processing
sc1	Cold HDD	250	250 MB/s	Infrequent access, archives

EBS Volume Types - Deep Dive

gp3 (Recommended for Most Workloads)

```
┌────────────────────────────────────────┐
│              gp3 SSD                   │
├────────────────────────────────────────┤
│ Baseline: 3,000 IOPS, 125 MB/s         │
│ Max: 16,000 IOPS, 1,000 MB/s           │
│ Size: 1 GB - 16 TB                     │
│ Cost: ~$0.08/GB/month                  │
├────────────────────────────────────────┤
│ ✅ Independent IOPS & throughput       │
│ ✅ 20% cheaper than gp2                │
│ ✅ Best price-performance              │
└────────────────────────────────────────┘
```

io2 Block Express (Highest Performance)

```
┌────────────────────────────────────────┐
│         io2 Block Express              │
├────────────────────────────────────────┤
│ Max: 256,000 IOPS                      │
│ Max: 4,000 MB/s throughput             │
│ Size: 4 GB - 64 TB                     │
│ Durability: 99.999%                    │
├────────────────────────────────────────┤
│ ✅ Sub-millisecond latency             │
│ ✅ Multi-Attach support                │
│ ✅ Mission-critical databases          │
└────────────────────────────────────────┘
```

EBS Snapshots

Snapshots are **incremental backups** stored in S3:

```
Day 1: Full snapshot (10 GB)
       ┌──────────────┐
       │ Snapshot 1   │ ← All data
       │   (10 GB)    │
       └──────────────┘

Day 2: Changed 2 GB
       ┌──────────────┐
       │ Snapshot 2   │ ← Only changed blocks (2 GB)
       │   (2 GB)     │
       └──────────────┘

Day 3: Changed 1 GB
       ┌──────────────┐
       │ Snapshot 3   │ ← Only changed blocks (1 GB)
       │   (1 GB)     │
       └──────────────┘
```

EBS Snapshot Features

Feature	Description
Snapshot Archive	Move to archive tier (75% cheaper, 24-72hr restore)
Recycle Bin	Protect against accidental deletion (1 day - 1 year)
Fast Snapshot Restore	Instant full performance (costs extra)
Cross-Region Copy	Copy snapshots to other regions for DR

Cross-Account Sharing	Share snapshots with other AWS accounts

EBS Encryption

```
┌──────────────────────────────────────────────────────────────────┐
│                      EBS Encryption                              │
├──────────────────────────────────────────────────────────────────┤
│                                                                  │
│  Encrypted at rest using AWS KMS keys                            │
│                                                                  │
│  What's encrypted:                                               │
│  ✅ Data at rest inside the volume                               │
│  ✅ Data in transit between EC2 and EBS                          │
│  ✅ All snapshots                                                │
│  ✅ All volumes created from snapshots                           │
│                                                                  │
│  Key points:                                                     │
│  • Minimal latency impact                                        │
│  • Uses AES-256 encryption                                       │
│  • Can use default AWS key or custom CMK                         │
│  • Cannot directly encrypt an unencrypted volume                 │
│                                                                  │
└──────────────────────────────────────────────────────────────────┘
```

To encrypt an existing unencrypted volume:

```
# 1. Create snapshot of unencrypted volume
aws ec2 create-snapshot --volume-id vol-xxx

# 2. Copy snapshot with encryption
aws ec2 copy-snapshot --source-snapshot-id snap-xxx --encrypted

# 3. Create new volume from encrypted snapshot
aws ec2 create-volume --snapshot-id snap-encrypted --encrypted
```

EBS CLI Commands

```
# Create a volume
aws ec2 create-volume \
  --availability-zone us-east-1a \
  --size 100 \
  --volume-type gp3 \
  --iops 3000 \
  --throughput 125

# Attach volume to EC2
aws ec2 attach-volume \
  --volume-id vol-xxx \
  --instance-id i-xxx \
  --device /dev/sdf

# Create snapshot
aws ec2 create-snapshot \
  --volume-id vol-xxx \
  --description "My backup"

# List volumes
aws ec2 describe-volumes

# Modify volume (resize, change type)
aws ec2 modify-volume \
  --volume-id vol-xxx \
  --size 200 \
  --volume-type gp3
```

Using EBS on EC2 (Linux)

```
# 1. List block devices
lsblk

# 2. Check if volume has filesystem
sudo file -s /dev/xvdf

# 3. Create filesystem (if new volume)
sudo mkfs -t xfs /dev/xvdf

# 4. Create mount point
sudo mkdir /data

# 5. Mount the volume
```

```bash
sudo mount /dev/xvdf /data

# 6. Add to /etc/fstab for persistence
echo "/dev/xvdf /data xfs defaults,nofail 0 2" | sudo tee -a /etc/fstab

# 7. Verify
df -h
```

Amazon EFS (Elastic File System)

What is EFS?

EFS is a **managed NFS (Network File System)** that can be mounted on multiple EC2 instances simultaneously. Think of it as a shared network drive in the cloud.

Key Characteristics

- **File-level storage** - Works like a network share
- **Multi-AZ** - Automatically replicated across AZs
- **Shared access** - Multiple EC2s can read/write simultaneously
- **Auto-scaling** - Grows and shrinks automatically
- **Pay per use** - Only pay for what you store

EFS vs EBS Comparison

```
┌─────────────────────────────────────────────────────────────────┐
│                   EBS vs EFS                                    │
├───────────────────────────┬─────────────────────────────────────┤
│          EBS              │              EFS                    │
├───────────────────────────┼─────────────────────────────────────┤
│                           │                                     │
│      ┌─────────┐          │      ┌─────────┐   ┌─────────┐      │
│      │   EC2   │          │      │   EC2   │   │   EC2   │      │
│      └────┬────┘          │      └────┬────┘   └────┬────┘      │
│           │               │           │             │           │
│      ┌────▼────┐          │           └──────┬──────┘           │
│      │   EBS   │          │                  │                  │
│      │ Volume  │          │             ┌────▼────┐             │
│      └─────────┘          │             │   EFS   │             │
│                           │             └────┬────┘             │
│  • One EC2 at a time      │                  │                  │
│  • Single AZ              │             ┌────▼────┐             │
│  • Fixed size             │             │   EC2   │             │
│  • Lower latency          │             └─────────┘             │
│  • Cheaper per GB         │                                     │
│                           │  • Multiple EC2s                    │
│                           │  • Multi-AZ                         │
│                           │  • Auto-scaling                     │
│                           │  • Higher latency                   │
│                           │  • More expensive per GB            │
└───────────────────────────┴─────────────────────────────────────┘
```

EFS Storage Classes

Storage Class	Description	Use Case
Standard	Frequently accessed	Active workloads
Standard-IA	Infrequent Access	Files accessed < 1x/month
One Zone	Single AZ	Dev/test, backups
One Zone-IA	Single AZ + Infrequent	Cost-sensitive, non-critical

EFS Lifecycle Management

Automatically move files between storage classes:

```
┌────────────────────────────────────────────────────────────────┐
│                  EFS Lifecycle Policy                          │
├────────────────────────────────────────────────────────────────┤
│                                                                │
│   File uploaded ──► Standard Storage                           │
│                          │                                      │
│                    (30 days no access)                          │
│                          │                                      │
│                          ▼                                      │
│                    Standard-IA (65% cheaper)                    │
│                          │                                      │
│                    (File accessed)                              │
│                          │                                      │
│                          ▼                                      │
│                    Back to Standard                             │
│                                                                 │
│   Configurable: 7, 14, 30, 60, or 90 days                      │
└────────────────────────────────────────────────────────────────┘
```

EFS Performance Modes

Mode	Description	Use Case
General Purpose	Low latency	Web serving, CMS, dev environments
Max I/O	Higher throughput, higher latency	Big data, media processing

EFS Throughput Modes

Mode	Description	Use Case
Bursting	Throughput scales with size	Most workloads
Provisioned	Fixed throughput	Consistent high throughput
Elastic	Auto-scales throughput	Unpredictable workloads

```
Bursting Throughput:
┌─────────────────┬─────────────────────┬─────────────────────┐
│ Storage Size    │ Baseline Throughput │ Burst Throughput    │
├─────────────────┼─────────────────────┼─────────────────────┤
│ 1 TB            │ 50 MB/s             │ 100 MB/s            │
│ 10 TB           │ 500 MB/s            │ 1 GB/s              │
│ 100 TB          │ 5 GB/s              │ 10 GB/s             │
└─────────────────┴─────────────────────┴─────────────────────┘
```

Prerequisites

Before starting, ensure you have:

Requirement	Details
EC2 Instances	Running in a VPC
VPC ID	Note your VPC ID (vpc-xxx)
Subnet IDs	Note subnet IDs where EC2s are running
EC2 Security Group ID	Note your EC2's security group (sg-xxx)
AWS CLI	Configured with appropriate permissions
SSH Access	Ability to SSH into your EC2 instances

Step 1: Create EFS Security Group

We'll create a dedicated security group for EFS that allows NFS traffic (port 2049) from your EC2 instances.

Using AWS Console

1. Navigate to **EC2** → **Security Groups** → **Create security group**
2. **Basic details:**
   - Security group name: `efs-mount-sg`
   - Description: `Security group for EFS mount targets - allows NFS`

  - VPC: Select your VPC
3. **Outbound rules:** Leave default (Allow all)
4. Click **Create security group**
5. **Note the Security Group ID** (e.g., `sg-0abc123def456`)

Using AWS CLI

```
# Set variables (replace with your values)
VPC_ID="vpc-XXXXXXXXX"
EC2_SG_ID="sg-XXXXXXXXX"  # Your existing EC2 security group

# Create security group for EFS
EFS_SG_ID=$(aws ec2 create-security-group \
  --group-name efs-mount-sg \
  --description "Security group for EFS mount targets - allows NFS" \
  --vpc-id $VPC_ID \
  --query 'GroupId' \
  --output text)

echo "Created EFS Security Group: $EFS_SG_ID"

# Add inbound rule to allow NFS (port 2049) from EC2 security group
aws ec2 authorize-security-group-ingress \
  --group-id $EFS_SG_ID \
  --protocol tcp \
  --port 2049 \
  --source-group $EC2_SG_ID

echo "Added inbound rule: Allow NFS from $EC2_SG_ID"

# Add Name tag
aws ec2 create-tags \
  --resources $EFS_SG_ID \
  --tags Key=Name,Value=efs-mount-sg
```

Verify Security Group

```
# View the security group rules
aws ec2 describe-security-groups \
  --group-ids $EFS_SG_ID \
  --query 'SecurityGroups[0].IpPermissions'
```

Step 2: Create EFS File System

Using AWS Console

1. Navigate to **EFS** → **Create file system**
2. Click **Customize** (for full control)
3. **Step 1 - General:**
   - Name: `my-efs-filesystem`
   - Storage class:
     - **Regional** (Multi-AZ) - for production
     - **One Zone** - for dev/test (cheaper)
   - Automatic backups: Enable (recommended)
   - Lifecycle management:
     - Transition into IA: 30 days after last access
     - Transition out of IA: On first access
   - Encryption: ✅ Enable encryption of data at rest
   - Click **Next**
4. **Step 2 - Network:**
   - VPC: Select your VPC
   - Mount targets: **We'll add these in the next step**
   - Click **Next**
5. **Step 3 - File system policy:** (Optional)

- Skip for now, click **Next**

6. **Step 4 - Review and create:**
   - Review settings
   - Click **Create**
7. **Note the File System ID** (e.g., `fs-0f34b6bab45a59c57`)

Using AWS CLI

```bash
# Set variables
REGION="us-east-1"

# Create EFS file system
EFS_ID=$(aws efs create-file-system \
  --performance-mode generalPurpose \
  --throughput-mode bursting \
  --encrypted \
  --tags Key=Name,Value=my-efs-filesystem \
  --region $REGION \
  --query 'FileSystemId' \
  --output text)

echo "Created EFS File System: $EFS_ID"

# Wait for file system to be available
echo "Waiting for EFS to be available..."
aws efs describe-file-systems \
  --file-system-id $EFS_ID \
  --query 'FileSystems[0].LifeCycleState'

# Keep checking until "available"
while true; do
  STATUS=$(aws efs describe-file-systems \
    --file-system-id $EFS_ID \
    --query 'FileSystems[0].LifeCycleState' \
    --output text)
  if [ "$STATUS" = "available" ]; then
    echo "EFS is available!"
    break
  fi
  echo "Status: $STATUS - waiting..."
  sleep 5
done
```

Step 3: Create Mount Targets

Mount targets allow EC2 instances in each Availability Zone to access EFS.

Using AWS Console

1. Go to **EFS** → Select your file system → **Network** tab
2. Click **Create mount target**
3. For each Availability Zone where you have EC2 instances:
   - **Availability Zone:** Select AZ (e.g., us-east-1a)
   - **Subnet ID:** Select a subnet in that AZ
   - **Security groups:** Select `efs-mount-sg` (the one we created)
4. Repeat for each AZ
5. Click **Save**
6. Wait for mount target status to show **Available**

Using AWS CLI

```bash
# Set variables (replace with your values)
EFS_ID="fs-XXXXXXXXX"
EFS_SG_ID="sg-XXXXXXXXX"  # The EFS security group we created

# Get subnet IDs from your VPC
aws ec2 describe-subnets \
  --filters "Name=vpc-id,Values=$VPC_ID" \
  --query 'Subnets[*].[SubnetId,AvailabilityZone]' \
  --output table

# Create mount target in each subnet (replace subnet IDs)
# Subnet 1 (AZ-1a)
aws efs create-mount-target \
  --file-system-id $EFS_ID \
  --subnet-id subnet-XXXXXXXX \
  --security-groups $EFS_SG_ID

# Subnet 2 (AZ-1b)
aws efs create-mount-target \
  --file-system-id $EFS_ID \
  --subnet-id subnet-YYYYYYYY \
  --security-groups $EFS_SG_ID

# Subnet 3 (AZ-1c) - if needed
aws efs create-mount-target \
  --file-system-id $EFS_ID \
  --subnet-id subnet-ZZZZZZZZ \
  --security-groups $EFS_SG_ID
```

Wait until all mount targets show `available` status (1-2 minutes)

Step 4: Install EFS Utils on EC2

SSH into your EC2 instance(s) and install the required packages.

Note - You have 2 options to mount, using EFS-utils or you can use NFS

Amazon Linux 2

```
# Update packages
sudo yum update -y

# Install amazon-efs-utils
sudo yum install -y amazon-efs-utils

# Verify installation
mount.efs --version
```

Amazon Linux 2023

```
# Update packages
sudo dnf update -y

# Install amazon-efs-utils
sudo dnf install -y amazon-efs-utils

# Verify installation
mount.efs --version
```

Ubuntu 20.04 / 22.04 / 24.04

```
# Update packages
sudo apt-get update

# Install dependencies
sudo apt-get install -y git binutils stunnel4 nfs-common

# Clone the efs-utils repository
cd /tmp
git clone https://github.com/aws/efs-utils

# Build the .deb package
cd efs-utils
./build-deb.sh

# Install the package
sudo apt-get install -y ./build/amazon-efs-utils*deb

# Verify installation
mount.efs --version
```

If You Get "Go is required for FIPS" Error on Ubuntu:

```
# Install Go
sudo apt-get install -y golang

# Rebuild
cd /tmp/efs-utils
./build-deb.sh

# Install
```

```
sudo apt-get install -y ./build/amazon-efs-utils*deb
```

Alternative: Install Only NFS Client (Without efs-utils)

If you don't need the EFS mount helper:

```
# Amazon Linux
sudo yum install -y nfs-utils

# Ubuntu
sudo apt-get install -y nfs-common
```

Step 5: Mount EFS

Create Mount Point

```
# Create directory for mounting EFS
sudo mkdir -p /efs

# Set permissions (optional)
sudo chown ec2-user:ec2-user /efs   # Amazon Linux
sudo chown ubuntu:ubuntu /efs        # Ubuntu
```

Method 1: Using EFS Mount Helper (Recommended)

```
# Replace fs-XXXXXXXXX with your EFS File System ID

# Mount with TLS encryption (RECOMMENDED)
sudo mount -t efs -o tls fs-XXXXXXXXX:/ /efs

# OR mount without TLS (faster but unencrypted)
sudo mount -t efs fs-XXXXXXXXX:/ /efs
```

Method 2: Using NFS4 Directly

```
# Replace fs-XXXXXXXXX with your EFS ID
# Replace us-east-1 with your region

sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-0f34b6bab45a59c57.efs.us-east-1.amazonaws.com:/ /efs
```

Mount Options Explained:

Option	Value	Purpose
`-t nfs4`	•	Use NFS version 4
`nfsvers=4.1`	4.1	NFS protocol version
`rsize=1048576`	1 MB	Read buffer size
`wsize=1048576`	1 MB	Write buffer size
`hard`	•	Keep retrying on failure
`timeo=600`	60 sec	Timeout before retry
`retrans=2`	2	Number of retries
`noresvport`	•	Allow reconnection

Verify Mount

```
# Check if mounted
df -h | grep efs

# Or use mount command
mount | grep efs

# Check mount point
ls -la /efs
```

Step 6: Make Mount Persistent

Add entry to `/etc/fstab` so EFS mounts automatically after reboot.

Using EFS Mount Helper

```
# Add to /etc/fstab (with TLS)
echo "fs-XXXXXXXXX:/ /efs efs _netdev,tls 0 0" | sudo tee -a /etc/fstab

# OR without TLS
echo "fs-XXXXXXXXX:/ /efs efs _netdev 0 0" | sudo tee -a /etc/fstab
```

Using NFS4

```
# Add to /etc/fstab
echo "fs-XXXXXXXXX.efs.us-east-1.amazonaws.com:/ /efs nfs4
nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport,_netdev 0 0" | sudo tee -a /etc/fstab
```

Important fstab Options

Option	Purpose
`_netdev`	Wait for network before mounting
`tls`	Enable encryption in transit (efs mount helper only)
`0 0`	Don't dump, don't fsck

Troubleshooting

Error: "Connection timed out" or "mount.nfs4: Connection timed out"

Cause: Security group not allowing NFS traffic

Solution:

```
# Verify EFS security group has correct inbound rule
aws ec2 describe-security-groups \
  --group-ids $EFS_SG_ID \
  --query 'SecurityGroups[0].IpPermissions[?FromPort==`2049`]'

# The source should be your EC2 security group
```

Fix: Add inbound rule for port 2049 from EC2 security group

Error: "mount.nfs4: access denied by server"

Cause: Mount target not available or incorrect file system ID

Solution:

```
# Check mount targets are available
aws efs describe-mount-targets --file-system-id $EFS_ID

# Verify file system ID is correct
aws efs describe-file-systems --query 'FileSystems[*].[FileSystemId,Name]'
```

Error: "mount.nfs4: No such file or directory"

Cause: Mount point doesn't exist

Solution:

```
sudo mkdir -p /efs
```

Error: "mount.efs: Failed to resolve" or DNS resolution failed

Cause: VPC DNS settings or mount target not ready

Solution:

```
# Test DNS resolution
nslookup fs-XXXXXXXXX.efs.us-east-1.amazonaws.com

# Check VPC DNS settings
aws ec2 describe-vpc-attribute --vpc-id $VPC_ID --attribute enableDnsSupport
aws ec2 describe-vpc-attribute --vpc-id $VPC_ID --attribute enableDnsHostnames

# Both should return "Value": true
```

Cleanup

When you're done, clean up resources to avoid charges:

Using Console

1. **Unmount EFS** on all EC2 instances:

   ```
   sudo umount /efs
   ```

2. **Remove fstab entry:**

   ```
   sudo nano /etc/fstab
   # Remove the EFS line
   ```

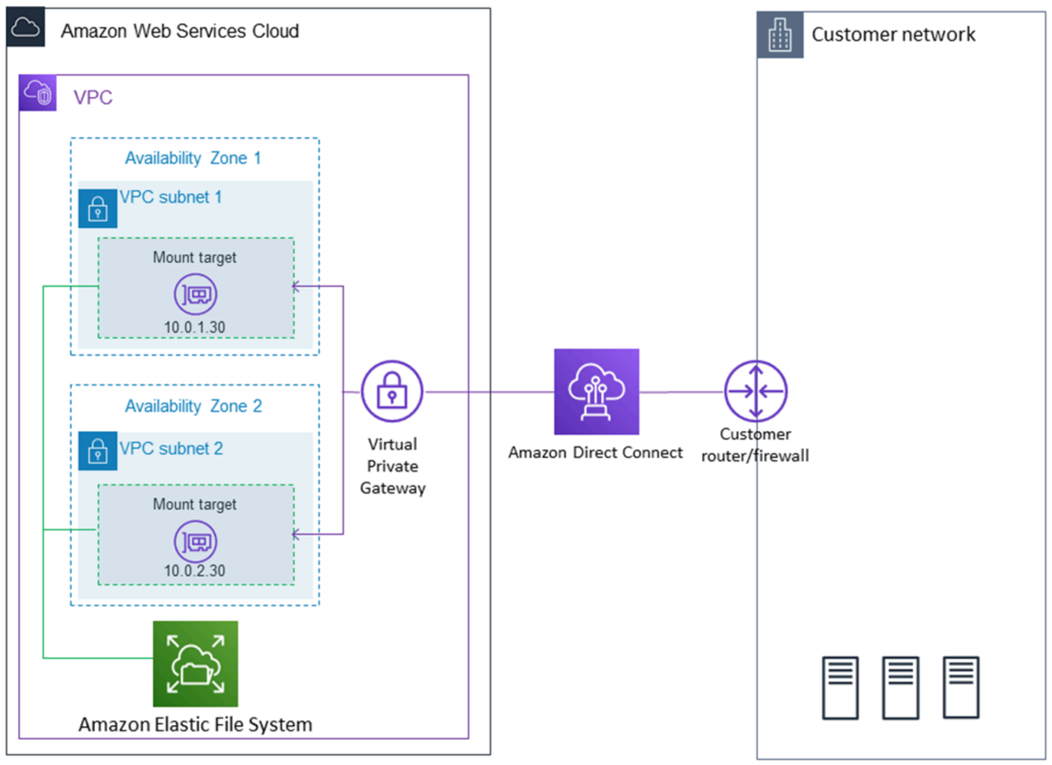
3. **Delete Mount Targets:**
   - EFS → Select file system → Network → Delete all mount targets
   - Wait for deletion to complete
4. **Delete EFS File System:**
   - EFS → Select file system → Delete
5. **Delete EFS Security Group:**
   - EC2 → Security Groups → Delete `efs-mount-sg`

Using CLI

```
# 1. Get mount target IDs
MOUNT_TARGETS=$(aws efs describe-mount-targets \
  --file-system-id $EFS_ID \
  --query 'MountTargets[*].MountTargetId' \
  --output text)

# 2. Delete each mount target
for MT in $MOUNT_TARGETS; do
  echo "Deleting mount target: $MT"
  aws efs delete-mount-target --mount-target-id $MT
done

# 3. Wait for mount targets to be deleted (1-2 minutes)
echo "Waiting for mount targets to be deleted..."
sleep 60

# 4. Delete EFS file system
aws efs delete-file-system --file-system-id $EFS_ID
echo "Deleted EFS: $EFS_ID"

# 5. Delete security group
aws ec2 delete-security-group --group-id $EFS_SG_ID
echo "Deleted Security Group: $EFS_SG_ID"

```

To mount EFS to on prem servers →

Reference - Tutorial: Mounting with on-premises Linux clients - Amazon Elastic File System



Amazon FSx

What is FSx?

FSx provides **fully managed third-party file systems** optimized for specific workloads.

FSx Family

```
┌──────────────────────────────────────────────────────────────────┐
│                     Amazon FSx Family                             │
├──────────────────────────────────────────────────────────────────┤
│                                                                   │
│  ┌─────────────────┐  ┌─────────────────┐  ┌─────────────────┐  │
│  │ FSx for Windows │  │ FSx for Lustre  │  │ FSx for NetApp  │  │
│  │   File Server   │  │                 │  │     ONTAP       │  │
│  ├─────────────────┤  ├─────────────────┤  ├─────────────────┤  │
│  │ • SMB protocol  │  │ • HPC workloads │  │ • NFS & SMB     │  │
│  │ • Windows apps  │  │ • ML training   │  │ • Multi-protocol│  │
│  │ • Active Dir    │  │ • S3 integration│  │ • Snapshots     │  │
│  │ • DFS namespaces│  │ • Sub-ms latency│  │ • Cloning       │  │
│  └─────────────────┘  └─────────────────┘  └─────────────────┘  │
│                                                                   │
│  ┌─────────────────┐                                             │
│  │ FSx for OpenZFS │                                             │
│  ├─────────────────┤                                             │
│  │ • NFS protocol  │                                             │
│  │ • Linux/macOS   │                                             │
│  │ • Snapshots     │                                             │
│  │ • Compression   │                                             │
│  └─────────────────┘                                             │
│                                                                   │
└──────────────────────────────────────────────────────────────────┘
```

FSx for Windows File Server

A fully managed Windows native file system.

Key Features

Feature	Description

Protocol	SMB (Server Message Block)
Integration	Active Directory (self-managed or AWS Managed AD)
Features	DFS, shadow copies, user quotas
Storage	SSD or HDD options
Size	32 GB - 64 TB (single AZ), up to 64 TB (multi-AZ)

Use Cases

- Windows application migrations
- Home directories
- SharePoint
- SQL Server databases
- Media workflows

Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│               FSx for Windows Architecture                       │
├─────────────────────────────────────────────────────────────────┤
│                                                                  │
│        On-Premises                          AWS                  │
│                                                                  │
│    ┌──────────────┐                   ┌──────────────┐          │
│    │   Active     │◄─────────────────►│ AWS Managed  │          │
│    │  Directory   │    AD Trust       │     AD       │          │
│    └──────────────┘                   └──────────────┘          │
│                                              │                   │
│    ┌──────────────┐                   ┌──────▼───────┐          │
│    │   Windows    │◄──────  VPN/DX ──►│  FSx for     │          │
│    │   Clients    │         SMB       │  Windows     │          │
│    └──────────────┘                   └──────────────┘          │
│                                              │                   │
│                                       ┌──────▼───────┐          │
│                                       │   Windows    │          │
│                                       │  EC2 (SMB)   │          │
│                                       └──────────────┘          │
│                                                                  │
└─────────────────────────────────────────────────────────────────┘
```

Quick Reference

Need	Best Choice
Boot volume for EC2	EBS (gp3)
Database storage	EBS (io2) or FSx ONTAP
Shared storage for Linux	EFS
Shared storage for Windows	FSx for Windows
ML/HPC with S3 data	FSx for Lustre
Enterprise NAS features	FSx for NetApp ONTAP
Migrating ZFS workloads	FSx for OpenZFS

Cost Comparison (Approximate)

Service	Price (per GB/month)	Notes
EBS gp3	$0.08	Provisioned capacity
EBS io2	$0.125 + IOPS cost	High performance
EFS Standard	$0.30	Pay for usage

EFS IA	$0.016	Infrequent access
EFS One Zone	$0.16	Single AZ
FSx Windows	$0.13 (SSD)	• throughput cost
FSx Lustre	$0.14 (Persistent)	Per GB provisioned