DevOps

[AWS DataSync 2](#_Toc193686227)

[AWS DataSync – Accelerated Data Transfer Service 2](#_Toc193686228)

[AWS DataSync: On-Premises to AWS Data Transfer 4](#_Toc193686229)

[AWS CLI 6](#_Toc193686230)

# AWS DataSync

## AWS DataSync – Accelerated Data Transfer Service

**1. Overview of AWS DataSync**

* **AWS DataSync enables fast and secure data transfer** between:
  + **On-Premises → AWS (S3, EFS, FSx)**
  + **AWS Services → AWS Services (S3 ↔ EFS, S3 ↔ FSx, etc.)**
* **Key Benefits:**
  + **10x faster** than traditional data transfer methods.
  + **Automates** data transfer tasks like scripting, scheduling, and monitoring.
  + **Eliminates manual efforts** that slow down data migrations.
  + Supports **terabytes (TBs) to petabytes (PBs) of data** transfer.

**2. Example: Transferring Data from S3 to EFS Using DataSync**

**Source Location: S3 Bucket**

* **Private S3 bucket** contains some objects.

**DataSync Agent (EC2 Instance)**

* **Acts as an intermediary** between the source (S3) and destination (EFS).
* **Security Groups (SGs) allow NFS, SSH, and HTTP** for data transfer.

**Destination Location: EFS (Elastic File System)**

* **Final storage destination** where the transferred files are stored.

**3. Steps to Set Up AWS DataSync**

**Step 1: Create AWS Resources**

1. **Create an S3 Private Bucket** → Upload some sample objects.
2. **Create an EFS (Elastic File System).**
3. **Deploy a DataSync Agent:**
   * Launch an **EC2 instance with DataSync Agent AMI** (Follow AWS documentation for AMI details).

**Step 2: Configure Security & Task**

1. **Modify Security Group (SG)** to allow **NFS, SSH, and HTTP protocols**.
2. **Create a DataSync Task in AWS Console:**
   * **Set Source Location → S3**
   * **Set Destination Location → EFS**
3. **Run the DataSync Task** to start data migration.

**Step 3: Verify Data in EFS**

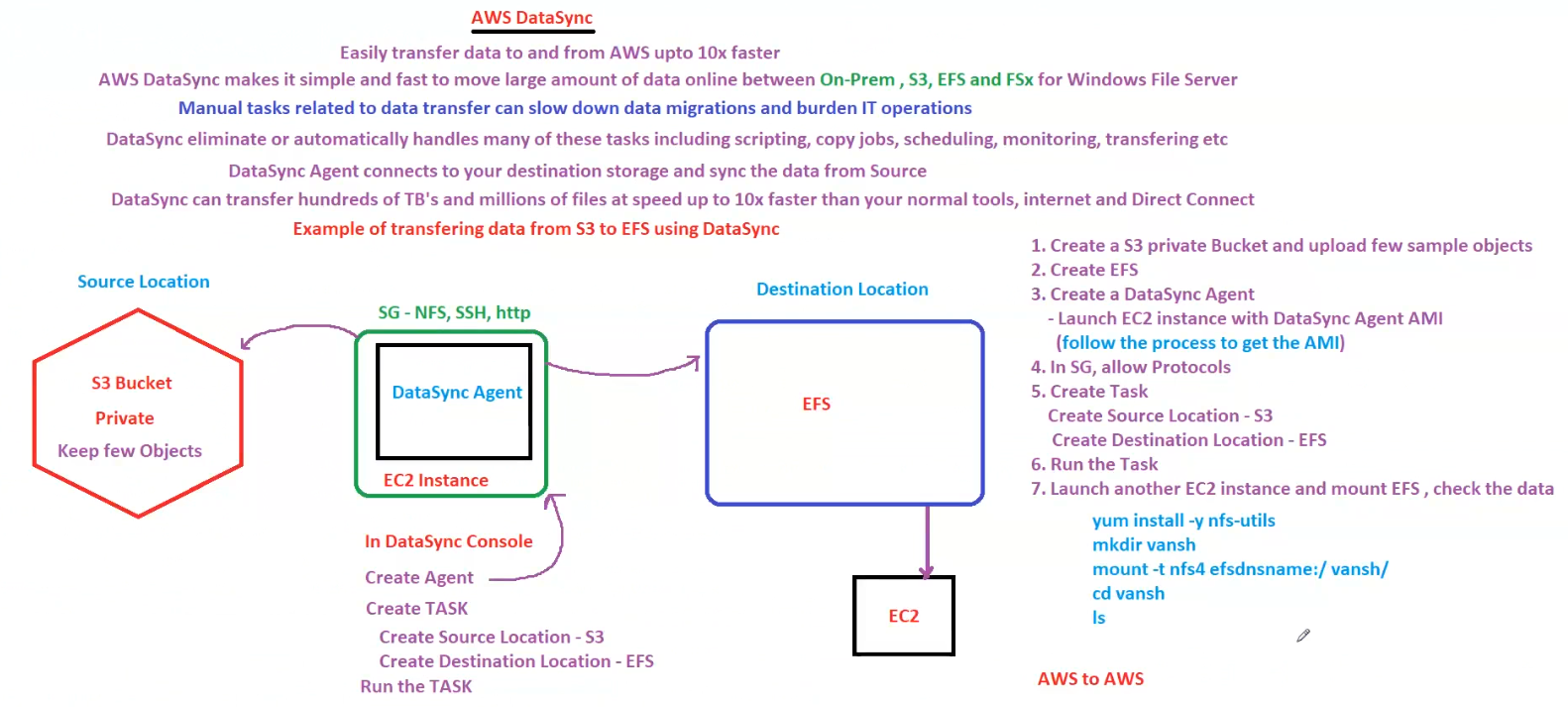
1. **Launch another EC2 instance and mount the EFS:**
2. yum install -y nfs-utils
3. mkdir vansh
4. mount -t nfs4 efsdnsname:/ vansh/
5. cd vansh
6. ls
   * **Checks if data has been successfully transferred.**

**4. AWS DataSync Console Workflow**

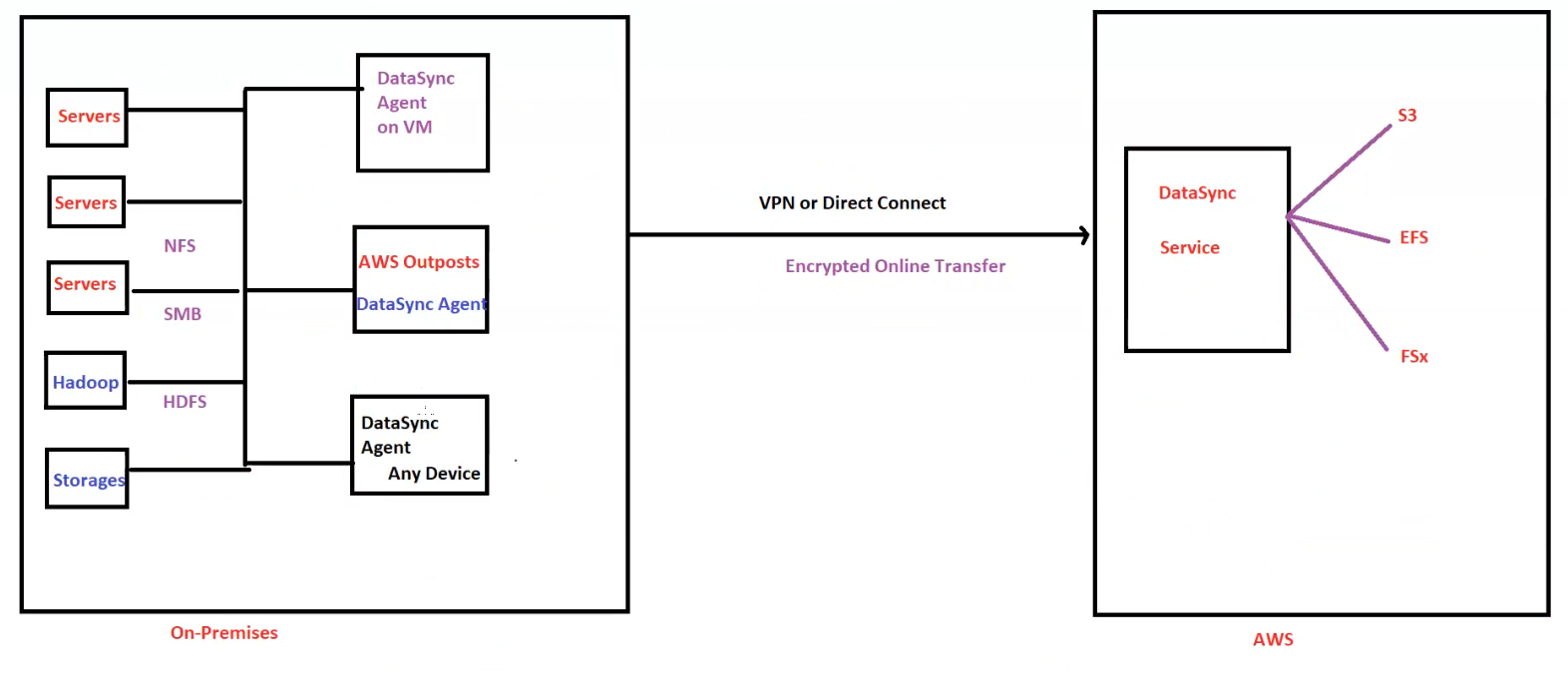
| **Step** | **Action** |
| --- | --- |
| **1** | Create DataSync Agent |
| **2** | Create DataSync Task |
| **3** | Set **Source Location** (S3) |
| **4** | Set **Destination Location** (EFS) |
| **5** | Run the Task |
| **6** | Verify data in EFS |

**5. Key Takeaways**

✅ **AWS DataSync automates and accelerates large-scale data transfers.**  
✅ **Useful for migrating data from S3 to EFS, FSx, or On-Premises to AWS.**  
✅ **Security Groups must allow necessary protocols for smooth data transfer.**  
✅ **Data can be verified by mounting EFS on an EC2 instance.**



## AWS DataSync: On-Premises to AWS Data Transfer



**AWS DataSync: On-Premises to AWS Data Transfer**

**1. Overview of AWS DataSync**

* **AWS DataSync enables secure and fast transfer of large data sets** from **on-premises storage to AWS services**.
* **Supports file transfers between:**
  + **On-Premises → AWS (S3, EFS, FSx)**
  + **On-Premises → AWS Outposts**
  + **Between AWS services**
* **Key Benefits:**
  + **Automates data transfers**, reducing manual effort.
  + **Encrypts data in transit** via **VPN or AWS Direct Connect**.
  + **10x faster** than traditional transfer tools.

**2. DataSync Architecture**

**On-Premises Components**

* **Data Sources:**
  + **Servers (NFS, SMB)**
  + **Hadoop (HDFS)**
  + **Storage Devices**
* **DataSync Agent:**
  + Runs on **a Virtual Machine (VM) or AWS Outposts**.
  + Connects to on-premises storage.
  + Transfers data securely to AWS.

**Data Transfer Process**

* **Data moves through an encrypted online transfer mechanism** (VPN or Direct Connect).
* Ensures **secure transmission to AWS**.

**AWS Destination Services**

* **AWS DataSync Service** receives data and transfers it to:
  + **S3 (Object Storage)**
  + **EFS (Elastic File System)**
  + **FSx (Windows File Server or Lustre File System)**

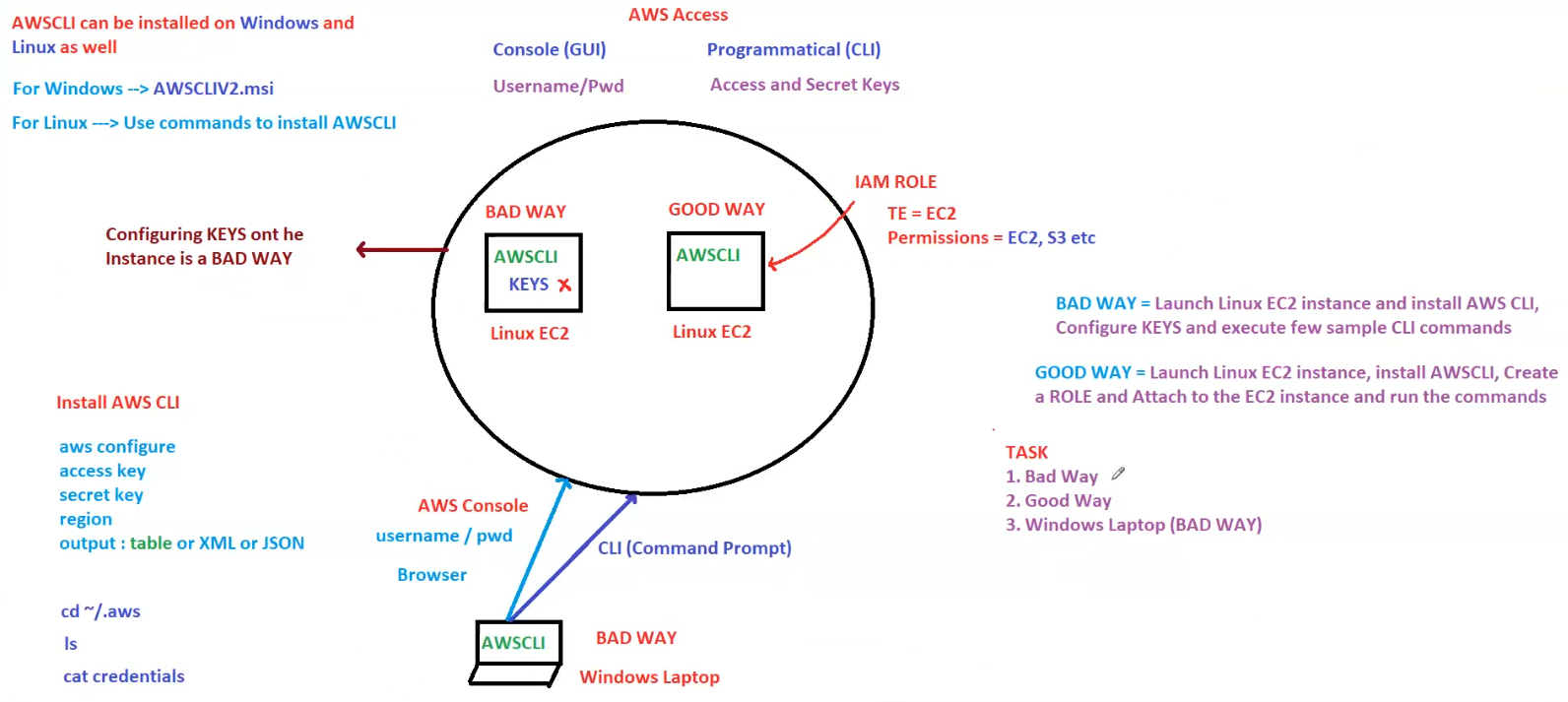
**3. Key Features of AWS DataSync**

| **Feature** | **Description** |
| --- | --- |
| **Automated Transfers** | Eliminates manual work by automating data migration. |
| **Secure Encryption** | Uses **VPN or Direct Connect** for **secure** transfers. |
| **Scalability** | Transfers **terabytes to petabytes** of data efficiently. |
| **Multi-Protocol Support** | Supports **NFS, SMB, HDFS** storage types. |

**4. Summary of Key Takeaways**

✅ **AWS DataSync simplifies on-premises to AWS data migration.**  
✅ **Supports multiple storage types (NFS, SMB, HDFS, AWS Outposts).**  
✅ **Transfers data securely using VPN or Direct Connect.**  
✅ **AWS DataSync Service integrates with S3, EFS, and FSx.**

# AWS CLI



**AWS CLI (Command Line Interface) – Best Practices & Installation**

**1. AWS CLI Installation**

* **AWS CLI can be installed on both Windows & Linux**.
  + **For Windows** → Install **AWSCLIV2.msi**.
  + **For Linux** → Use package managers like yum, apt-get, or pip.

**AWS CLI Setup Commands**

1. **Install AWS CLI**
2. **Configure AWS CLI** using:
3. aws configure
   * **Enter Access Key, Secret Key, Region, and Output Format** (Table, JSON, XML).
4. **Verify Configuration**
5. cd ~/.aws
6. ls
7. cat credentials

**2. AWS Access Methods**

| **Access Type** | **Description** |
| --- | --- |
| **Console (GUI)** | Access AWS via browser with **Username/Password**. |
| **CLI (Command Line Interface)** | Access AWS via **programmatic commands** using **AWS CLI**. |

**3. AWS CLI Authentication – Good vs. Bad Practices**

**❌ BAD WAY (Insecure Practice)**

* **Manually configuring AWS Access Keys on an EC2 instance**.
* **Saving Access & Secret Keys in AWS CLI configuration (~/.aws/credentials)**.
* **Risk:** If the instance is compromised, **AWS credentials can be stolen**.

**✅ GOOD WAY (Best Practice)**

* **Use IAM Roles instead of Access Keys**:
  + **Attach an IAM Role** to the EC2 instance.
  + **Grant permissions dynamically** (e.g., access to EC2, S3, etc.).
  + No need to store keys on the instance.
* **How to set up IAM Role for EC2:**
  + Create an **IAM Role** with required **permissions**.
  + Attach the IAM Role **to the EC2 instance**.
  + **Run AWS CLI commands without needing Access Keys**.

**4. Common AWS CLI Setup Scenarios**

| **Scenario** | **Good or Bad?** | **Reason** |
| --- | --- | --- |
| **EC2 with AWS CLI & IAM Role** | ✅ **Good** | Uses IAM Role for authentication. |
| **EC2 with AWS CLI & Access Keys** | ❌ **Bad** | Credentials are stored in the instance. |
| **AWS CLI on Windows Laptop with Access Keys** | ❌ **Bad** | If the laptop is lost, credentials can be stolen. |

**5. Summary of Key Takeaways**

✅ **Use IAM Roles for EC2 authentication instead of Access Keys.**  
✅ **Avoid storing AWS Access Keys on EC2 instances or personal devices.**  
✅ **Install AWS CLI properly on Windows & Linux for secure access.**  
✅ **AWS Console (GUI) and CLI can both be used for AWS access, but CLI requires authentication best practices.**