Apache Hadoop Distributed File System (HDFS) migration to Azure

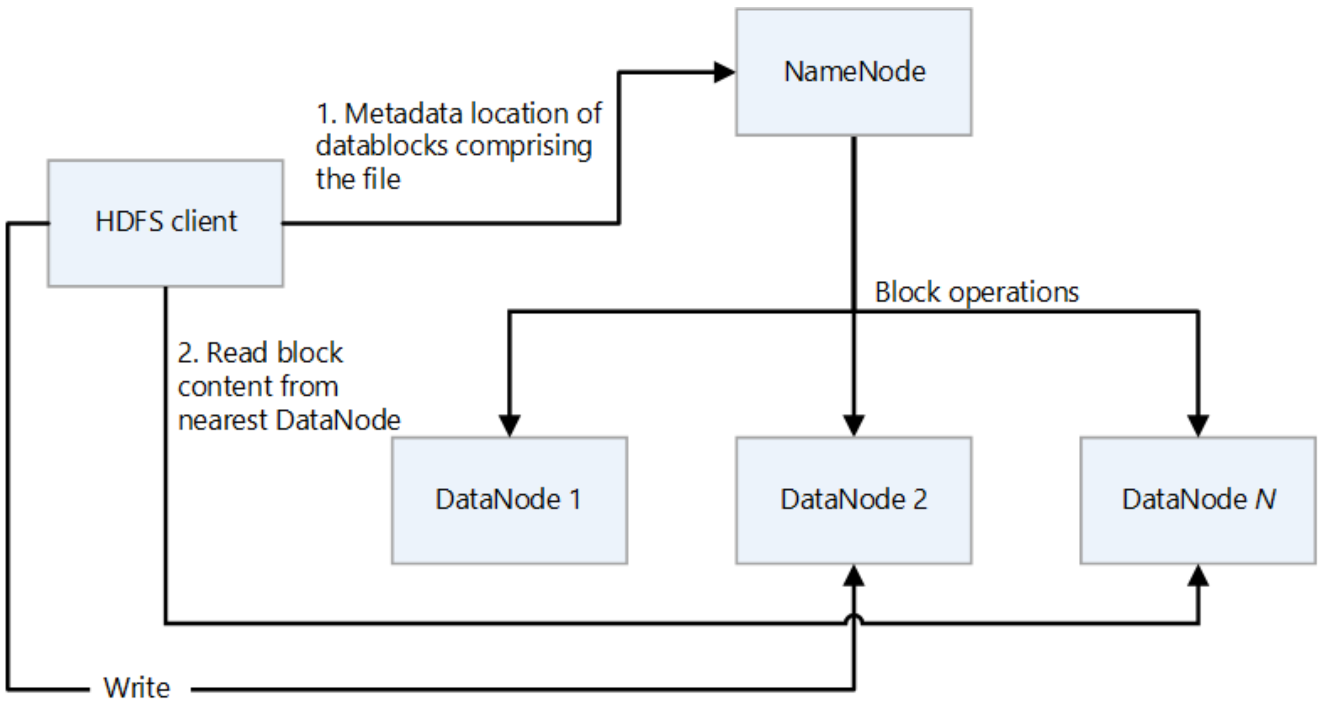
Azure HDInsight Azure Cosmos DB Azure Data Lake Storage Azure Synapse Analytics Azure Stream Analytics

The Hadoop Distributed File System (HDFS) is a Java-based distributed file system that provides reliable, scalable data storage that can span large clusters of commodity servers. This article provides an overview of HDFS and a guide to migrating it to Azure.

Apache ®, Apache Spark®, Apache Hadoop®, Apache Hive, and the flame logo are either registered trademarks or trademarks of the Apache Software Foundation in the United States and/or other countries. No endorsement by The Apache Software Foundation is implied by the use of these marks.

HDFS Architecture and Components

HDFS has a primary/secondary design. In the following diagram, NameNode is the primary and the DataNodes are the secondaries.


NameNode
1. Metadata location of datablocks comprising the file
HDFS client
Block operations
2. Read block content from nearest DataNode
DataNode 1
DataNode 2
DataNode N
Write

- **NameNode** manages access to files and to the file system namespace, which is a hierarchy of directories.
  - Files and directories are nodes on NameNode. They have attributes such as permissions, modification and access times, and size quotas for namespace and disk space.

  - A file comprises multiple blocks. The default block size is 128 megabytes. A non-default block size can be set for a cluster by modifying the hdfs-site.xml file.

  - Each block of the file is independently replicated at multiple DataNodes. The default value for the replication factor is three, but every cluster can have its own non-default value. The replication factor can be changed at any time. A change causes a cluster re-balancing.
  - NameNode maintains the namespace tree and the mapping of file blocks to DataNodes (the physical locations of file data).
  - When an HDFS client reads a file:

    1. It contacts the NameNode for the locations of the data blocks of the file.
    2. It reads block contents from the nearest DataNode.

  - HDFS keeps the entire namespace in RAM.
- **DataNodes** are the secondary nodes that perform read and write operations on the file system, and perform block operations such as creation, replication, and deletion.
  - A DataNode contains metadata files that hold the checksums of the stored files. For each block replica that's hosted by a DataNode, there's a corresponding metadata file that contains metadata about the replica, including its checksum information. The metadata file has the same base name as the block file, and extension *.meta*.
  - The DataNode contains the data file that holds the block's data.
  - When a DataNode reads a file, it fetches the block locations and replica locations from the NameNode and tries to read from the nearest location.
  - An HDFS cluster can have thousands of DataNodes and tens of thousands of HDFS clients per cluster. Each DataNode can execute multiple application tasks concurrently.
  - An end-to-end checksum calculation is performed as part of the HDFS write pipeline when a block is written to DataNodes.
- **HDFS Client** is the client that applications use to access files.
  - It's a code library that exports the HDFS file system interface.
  - It supports operations to read, write, and delete files, and operations to create and delete directories.
  - It performs the following steps when an application reads a file:
    - It gets from the NameNode a list of DataNodes and locations that hold the file blocks. The list includes the replicas.
    - It uses the list to get the requested blocks from the DataNodes.
  - HDFS provides an API that exposes the locations of file blocks. This allows applications like the MapReduce framework to schedule a task to run where the data is, in order to optimize read performance.

Feature map

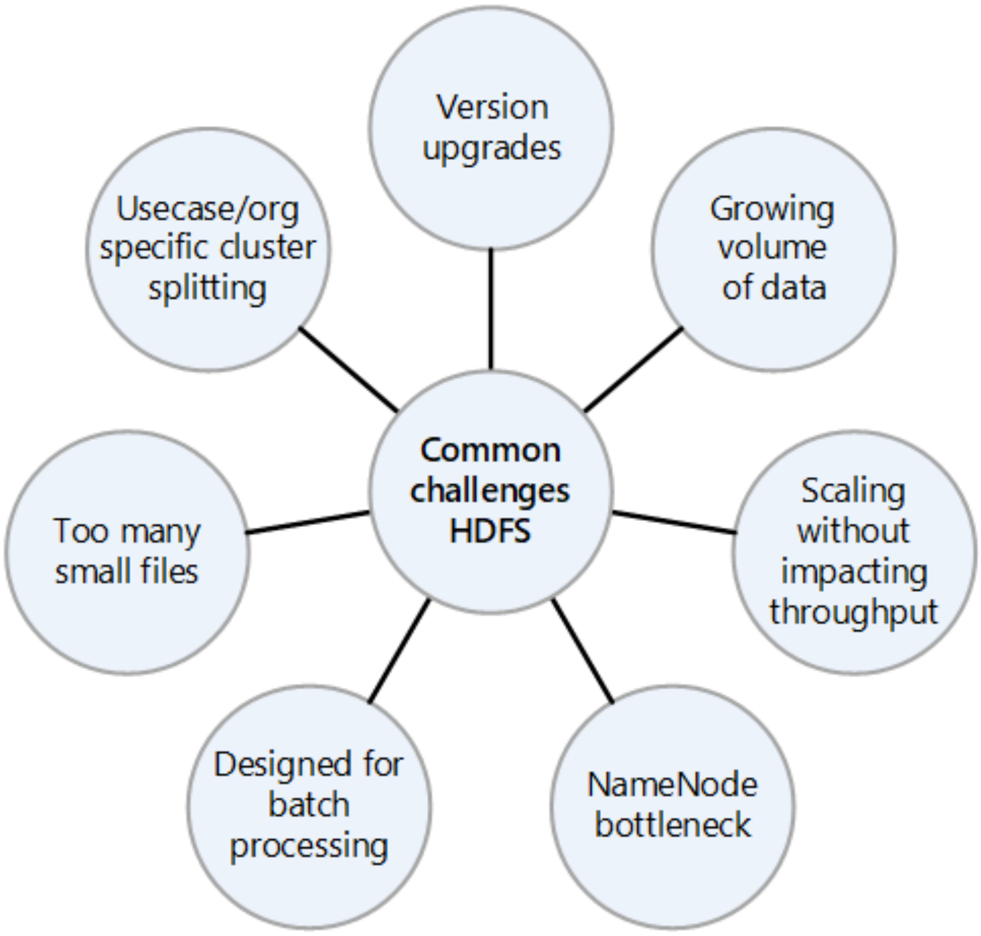
The Azure Blob Filesystem (ABFS) driver provides an interface that makes it possible for Azure Data Lake Storage to act as an HDFS file system. The following table compares the core functionality of the ABFS driver and Data Lake Storage to that of HDFS.

Expand table

Feature	The ABFS driver and Data Lake Storage	HDFS
Access that's compatible with Hadoop	You can manage and access data just as you would with HDFS. The ABFS driver is available in all Apache Hadoop environments, including Azure HDInsight and Azure Databricks.	A MapR cluster can access an external HDFS cluster with the hdfs:// or webhdfs:// protocols
POSIX permissions	The security model for Data Lake Gen2 supports access control list (ACL) and POSIX permissions along with some extra granularity that's specific to Data Lake Storage Gen2. Settings can be configured by using admin tools or frameworks like Apache Hive and Apache Spark.	Jobs that require file system features like strictly atomic directory renames, fine-grained HDFS permissions, or HDFS symlinks can only work on HDFS.
Cost effectiveness	Data Lake Storage offers low-cost storage capacity and transactions. Azure Blob Storage lifecycles help to lower costs by adjusting billing rates as data moves through its lifecycle.	
Optimized driver	The ABFS driver is optimized for big data analytics. The corresponding REST APIs are provided through the distributed file system (DFS) endpoint, dfs.core.windows.net.	
Block Size	**Blocks** is equivalent to a single **Append** API invocation (the **Append** API creates a new block) and is limited to 100 MB per invocation. However, the write pattern supports calling **Append** many times per file (even in parallel) to a maximum of 50,000 and then calling **Flush** (equivalent to **PutBlockList**).	HDFS stores the data in a data block. You set the block size by setting a value in the hdfs-site.xml file in the Hadoop directory. The default size is 128 MB.

	This is the way the maximum files size of 4.75TB is achieved.	
Default ACLS	Files don't have default ACLs and aren't enabled by default.	Files don't have default ACLs.
Binary Files	Binary files can be moved to Azure Blob Storage in a non-hierarchical namespace. Objects in Blob Storage are accessible via the Azure Storage REST API, Azure PowerShell, Azure CLI, or an Azure Storage client library. Client libraries are available for different languages, including .NET, Java, Node.js, Python, Go, PHP, and Ruby	Hadoop provides the ability to read and write binary files. SequenceFile is a flat file that consists of a binary key and value pairs. The SequenceFile provides Writer, Reader, and Sorter classes for writing, reading, and sorting. Convert the image or video file into a SequenceFile and store it in HDFS. Then use the HDFS SequenceFileReader/Writer methods or the put command: `bin/hadoop fs –put /src_image_file /dst_image_file`
Permission inheritance	Data Lake Storage uses the POSIX-style model and behaves the same as Hadoop if ACLs control access to an object. For more information, see Access control lists (ACLs) in Data Lake Storage Gen2.	Permissions for an item are stored on the item itself, not inherited after the item exists. Permissions are only inherited if default permissions are set on the parent item before the child item is created.
Data replication	Data in an Azure Storage account is replicated three times in the primary region. Zone-redundant storage is the recommended replication option. It synchronously replicates across three Azure availability zones in the primary region.	By default a file’s replication factor is three. For critical files or files that are accessed often, a higher replication factor improves fault tolerance and increases read bandwidth.
Sticky bit	In the context of Data Lake Storage, it's unlikely that the sticky bit is required. Briefly, if the sticky bit is enabled on a directory, a child item can only be deleted or renamed by the user that owns the child item. The sticky bit isn't shown in the Azure portal.	The sticky bit can be set on directories to prevent anyone except the superuser, directory owner, or file owner from deleting or moving files within the directory. Setting the sticky bit for a file has no effect.

Common challenges of an on-premises HDFS


Version upgrades
Usecase/org specific cluster splitting
Growing volume of data
Common challenges HDFS
Too many small files
Scaling without impacting throughput
Designed for batch processing
NameNode bottleneck

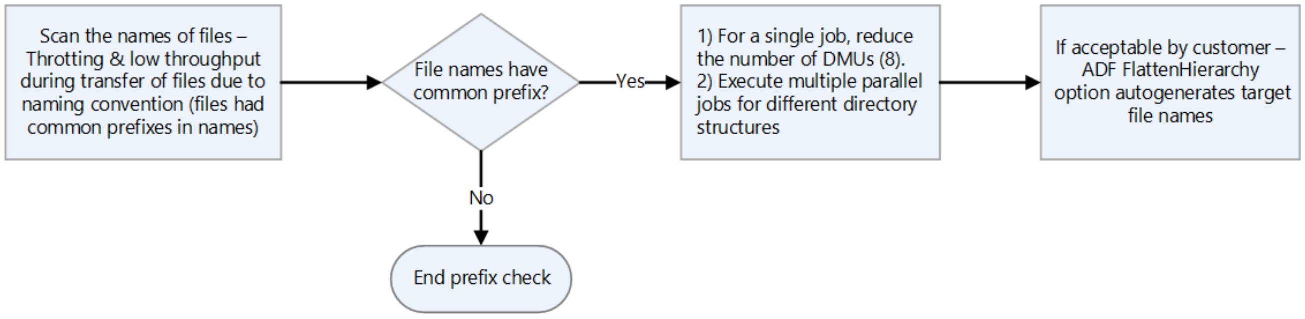
The many challenges presented by an on-premises HDFS implementation can be reasons to consider the advantages of migrating to the cloud:

- Frequent HDFS version upgrades
- Increasing amounts of data
- Having many small files that increase the pressure on the NameNode, which controls the metadata of all the files in the cluster. More files often means more read traffic on the NameNode when clients read the files, and more calls when clients are writing.
- If multiple teams in the organization require different datasets, splitting the HDFS clusters by use case or organization isn't possible. The result is that data duplication increases, which increases costs and reduces efficiency.
- The NameNode can become a performance bottleneck as the HDFS cluster is scaled up or out.
- Prior to Hadoop 2.0, all client requests to an HDFS cluster first pass through the NameNode, because all the metadata is stored in a single NameNode. This design makes the NameNode a possible bottleneck and single point of failure. If the NameNode fails, the cluster is unavailable.

Migration considerations

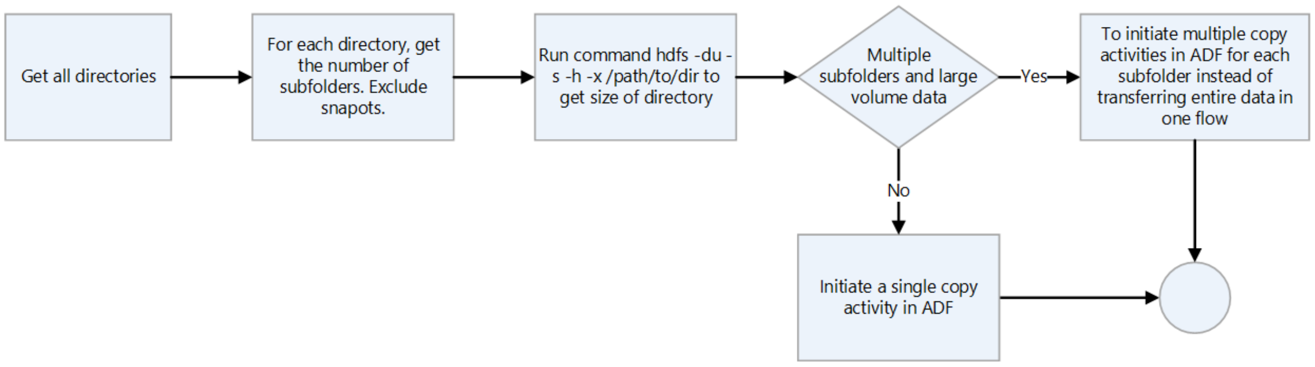
Here are some things that are important to consider when you plan a migration of HDFS to Data Lake Storage:

- Consider aggregating data that's in small files into a single file on Data Lake Storage.
- List all the directory structures in HDFS and replicate similar zoning in Data Lake Storage. You can obtain the directory structure of HDFS by using the `hdfs -ls` command.
- List all the roles that are defined in the HDFS cluster so that you can replicate them in the target environment.
- Note the data lifecycle policy of the files that are stored in HDFS.
- Keep in mind that some system features of HDFS aren't available on Data Lake Storage, including:
  - Strictly atomic renaming of directories
  - Fine-grained HDFS permissions
  - HDFS symlinks
- Azure Storage has geo-redundant replication, but it's not always wise to use it. It does provide data redundancy and geographic recovery, but a failover to a more distant location can severely degrade performance and incur additional costs. Consider whether the higher availability of the data is worth it.
- If files have names with the same prefixes, HDFS treats them as a single partition. Therefore, if you use Azure Data Factory, all data movement units (DMUs) write to a single partition.


Scan the names of files – Throtting & low throughput during transfer of files due to naming convention (files had common prefixes in names)
File names have common prefix?
Yes
1) For a single job, reduce the number of DMUs (8). 2) Execute multiple parallel jobs for different directory structures
If acceptable by customer – ADF FlattenHierarchy option autogenerates target file names
No
End prefix check

- If you use Data factory for data transfer, scan through each directory, excluding

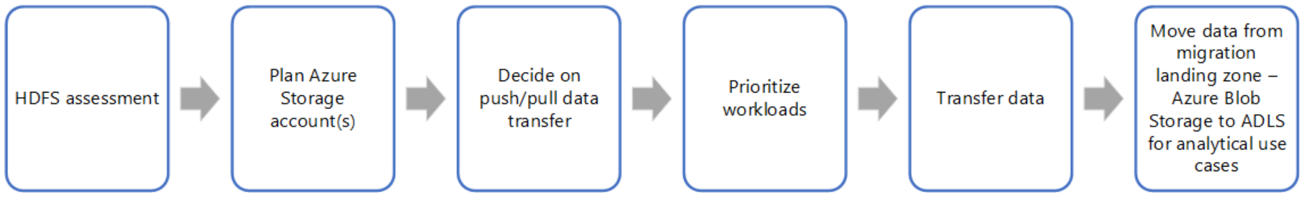
  snapshots, and check the directory size by using the `hdfs du` command. If there are multiple subdirectories and large amounts of data, initiate multiple copy activities in Data Factory. For example, use one copy per subdirectory rather than transferring the entire directory by using a single copy activity.


Get all directories
For each directory, get the number of subfolders. Exclude snapots.
Run command hdfs -du -s -h -x /path/to/dir to get size of directory
Multiple subfolders and large volume data
Yes
To initiate multiple copy activities in ADF for each subfolder instead of transferring entire data in one flow
No
Initiate a single copy activity in ADF

- Data platforms are often used for longer term retention of information that may have been removed from systems of record. You should plan to create tape backups or snapshots of the archived data. Consider replicating the information to a recovery site. Usually data is archived either for compliance or for historical data purposes. Before you archive data you should have a clear reason for keeping it. Also, decide when archived data is to be removed and establish processes to remove it at that time.
- The low cost of the Archive access tier of Data Lake Storage makes it an attractive option for archiving data. For more information, see Archive access tier.
- When an HDFS client uses the ABFS driver to access Blob Storage, there can be instances where the method that's used by the client isn't supported and AzureNativeFileSystem throws an UnsupportedOperationException. For example, `append(Path f, int bufferSize, Progressable progress)` isn't currently supported. To check on issues related to the ABFS driver, see Hadoop features and fixes.
- There's a backported version of the ABFS driver for use on older Hadoop clusters. For more information, see Backport for ABFS Driver.
- In an Azure virtual networking environment, the DistCp tool doesn't support Azure ExpressRoute private peering with an Azure Storage virtual network endpoint. For more information, see Use Azure Data Factory to migrate data from an on-premises Hadoop cluster to Azure Storage.

Migration approach

The typical approach to migrating HDFS to Data Lake Storage uses these steps:


HDFS assessment
Plan Azure Storage account(s)
Decide on push/pull data transfer
Prioritize workloads
Transfer data
Move data from migration landing zone – Azure Blob Storage to ADLS for analytical use cases

HDFS assessment

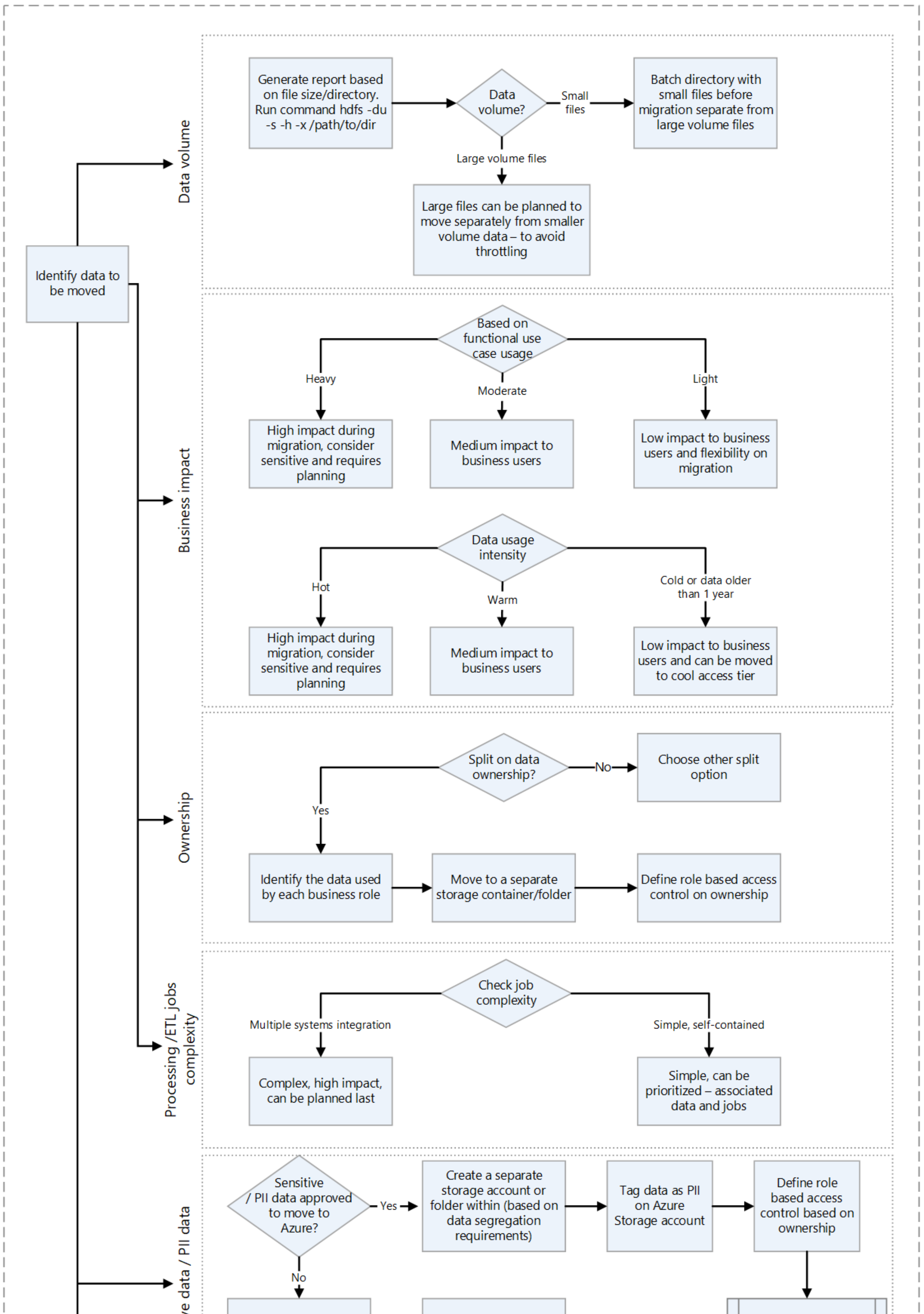
On-premises assessment scripts provide information that helps you to determine which workloads can be migrated to Azure and whether the data should be migrated all at once or a piece at a time. Third-party tools like Unravel can provide metrics and support auto-assessment of the on-premises HDFS. Some important factors to consider when planning include:

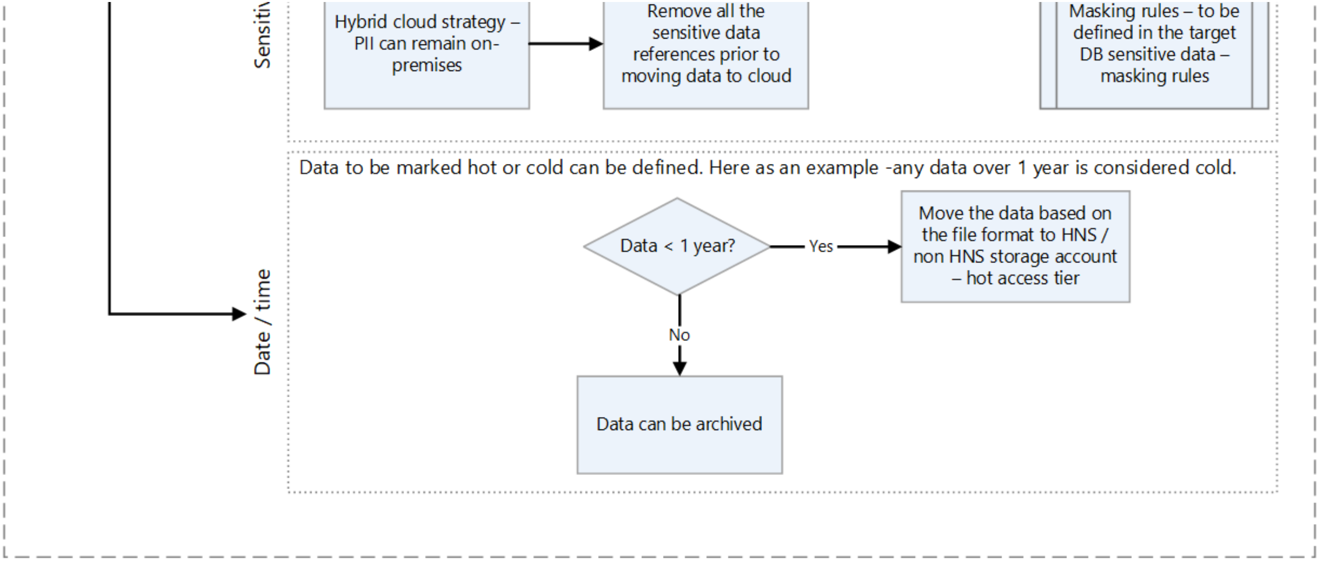
- Data volume
- Business impact
- Ownership of data
- Processing complexity
- Extract, transfer, and load (ETL) complexity
- Personally identifiable information (PII) and other sensitive data

Based on such factors, you can formulate a plan to move data to Azure that minimizes downtime and business disruption. Perhaps sensitive data can remain on-premises. Historical data can be moved and tested prior to moving an incremental load.

The following decision flow helps decide the criteria and commands to run to get the right information.

1. Data migration strategy/planning


Identify data to be moved
Data volume
Generate report based on file size/directory. Run command hdfs -du -s -h -x /path/to/dir
Data volume?
Small files
Batch directory with small files before migration separate from large volume files
Large volume files
Large files can be planned to move separately from smaller volume data – to avoid throttling
Business impact
Based on functional use case usage
Heavy
Moderate
Light
High impact during migration, consider sensitive and requires planning
Medium impact to business users
Low impact to business users and flexibility on migration
Data usage intensity
Hot
Warm
Cold or data older than 1 year
High impact during migration, consider sensitive and requires planning
Medium impact to business users
Low impact to business users and can be moved to cool access tier
Ownership
Split on data ownership?
No
Choose other split option
Yes
Identify the data used by each business role
Move to a separate storage container/folder
Define role based access control on ownership
Processing /ETL jobs complexity
Check job complexity
Multiple systems integration
Simple, self-contained
Complex, high impact, can be planned last
Simple, can be prioritized – associated data and jobs
e data / PII data
Sensitive / PII data approved to move to Azure?
Yes
Create a separate storage account or folder within (based on data segregation requirements)
Tag data as PII on Azure Storage account
Define role based access control based on ownership
No


Sensiti
Hybrid cloud strategy – PII can remain on-premises
Remove all the sensitive data references prior to moving data to cloud
Masking rules – to be defined in the target DB sensitive data – masking rules
Date / time
Data to be marked hot or cold can be defined. Here as an example -any data over 1 year is considered cold.
Data < 1 year?
Yes
Move the data based on the file format to HNS / non HNS storage account – hot access tier
No
Data can be archived

HDFS commands for getting assessment metrics from HDFS include:

- List all the directories in a location:

  ```
  hdfs dfs -ls books
  ```

- Recursively list all files in a location:

  ```
  hdfs dfs -ls -R books
  ```

- Get the size of the HDFS directory and files:

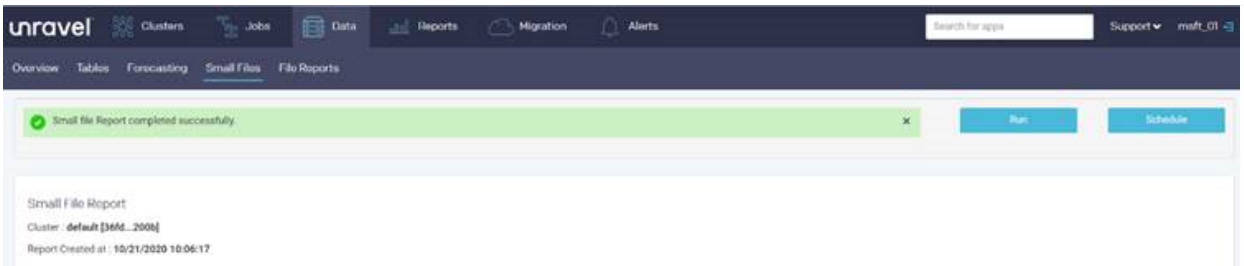
  ```
  hadoop fs -du -s -h command
  ```

  The `hadoop fs -du -s -h` command displays the size of the HDFS files and directory. Since the Hadoop file system replicates every file, the actual physical size of the file is the number of file replicas multiplied by the size of one replica.

- Determine whether ACLs are enabled. To do this, obtain the value of dfs.namenode.acls.enabled in Hdfs-site.xml. Knowing the value helps in planning access control on the Azure Storage account. For information about the contents of this file, see Default file settings .

Partner tools such as Unravel provide assessment reports for planning data migration. The tools must run in the on-premises environment or connect to the Hadoop cluster to generate reports.

- The following Unravel report provides statistics, per directory, about the small files in the directory:

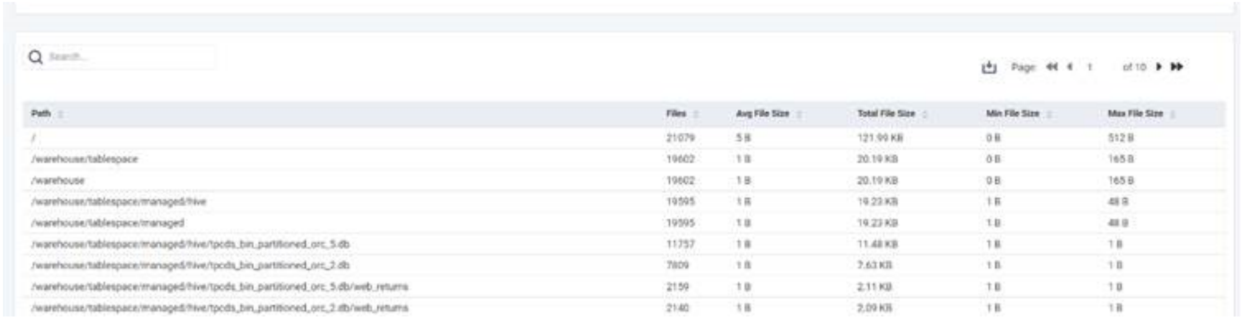


Small file Report completed successfully.

Small File Report

Cluster : default [36fd...200b]

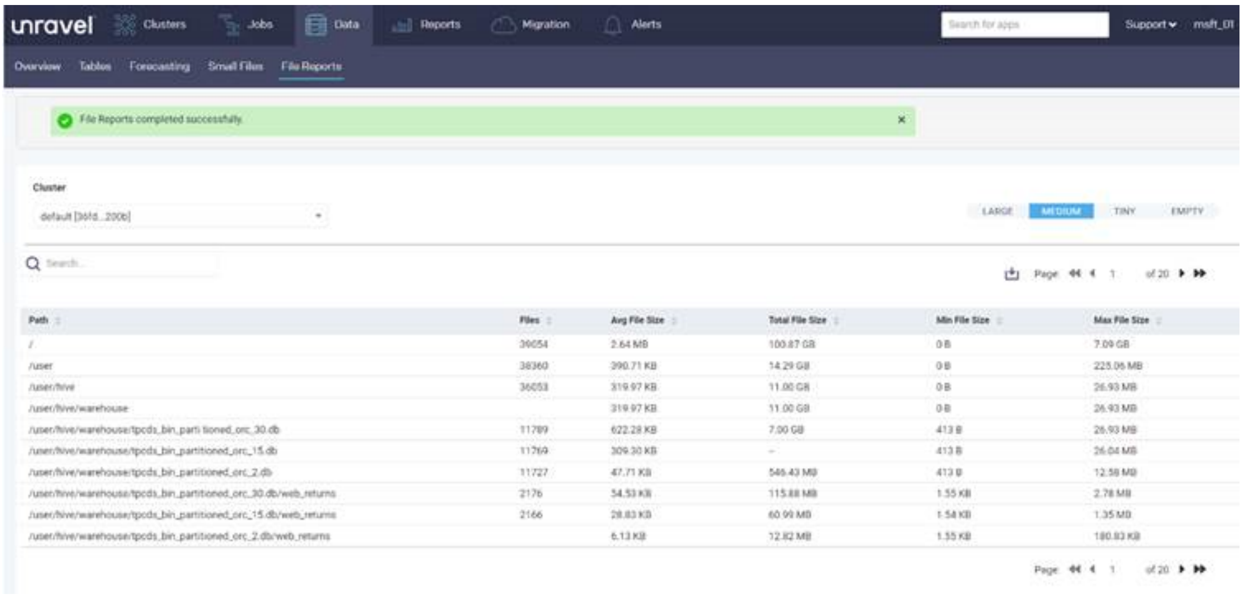
Report Created at : 10/21/2020 10:06:17



Page 1 of 10

Path	Files	Avg File Size	Total File Size	Min File Size	Max File Size
/	21079	5 B	121.99 KB	0 B	512 B
/warehouse/tablespace	19602	1 B	20.19 KB	0 B	165 B
/warehouse	19602	1 B	20.19 KB	0 B	165 B
/warehouse/tablespace/managed/hive	19595	1 B	19.23 KB	1 B	48 B
/warehouse/tablespace/managed	19595	1 B	19.23 KB	1 B	48 B
/warehouse/tablespace/managed/hive/tpcds_bin_partitioned_orc_5.db	11757	1 B	11.48 KB	1 B	1 B
/warehouse/tablespace/managed/hive/tpcds_bin_partitioned_orc_2.db	7809	1 B	7.63 KB	1 B	1 B
/warehouse/tablespace/managed/hive/tpcds_bin_partitioned_orc_5.db/web_returns	2159	1 B	2.11 KB	1 B	1 B
/warehouse/tablespace/managed/hive/tpcds_bin_partitioned_orc_2.db/web_returns	2140	1 B	2.09 KB	1 B	1 B

- The following report provides statistics, per directory, about the files in the directory:



File Reports completed successfully.

Cluster: default [36fd...200b]

Page 1 of 20

Path	Files	Avg File Size	Total File Size	Min File Size	Max File Size
/	39054	2.64 MB	100.87 GB	0 B	7.09 GB
/user	38360	390.71 KB	14.29 GB	0 B	225.06 MB
/user/hive	36053	319.97 KB	11.00 GB	0 B	26.93 MB
/user/hive/warehouse		319.97 KB	11.00 GB	0 B	26.93 MB
/user/hive/warehouse/tpcds_bin_parti tioned_orc_30.db	11789	622.28 KB	7.00 GB	413 B	26.93 MB
/user/hive/warehouse/tpcds_bin_partitioned_orc_15.db	11769	309.30 KB	-	413 B	26.04 MB
/user/hive/warehouse/tpcds_bin_partitioned_orc_2.db	11727	47.71 KB	546.43 MB	413 B	12.58 MB
/user/hive/warehouse/tpcds_bin_partitioned_orc_30.db/web_returns	2176	54.53 KB	115.88 MB	1.55 KB	2.78 MB
/user/hive/warehouse/tpcds_bin_partitioned_orc_15.db/web_returns	2166	28.83 KB	60.99 MB	1.54 KB	1.35 MB
/user/hive/warehouse/tpcds_bin_partitioned_orc_2.db/web_returns		6.13 KB	12.82 MB	1.55 KB	180.83 KB

Page 1 of 20

Transfer data

Data must be transferred to Azure as outlined in your migration plan. Transferring requires the following activities:

1. **Identify all of the ingestion points.**

   If, because of security requirements, data can't be landed to the cloud directly, then on-premises can serve as an intermediate landing zone. You can build pipelines in Data Factory to pull the data from on-premises systems, or use AZCopy scripts to push the data to the Azure Storage account.

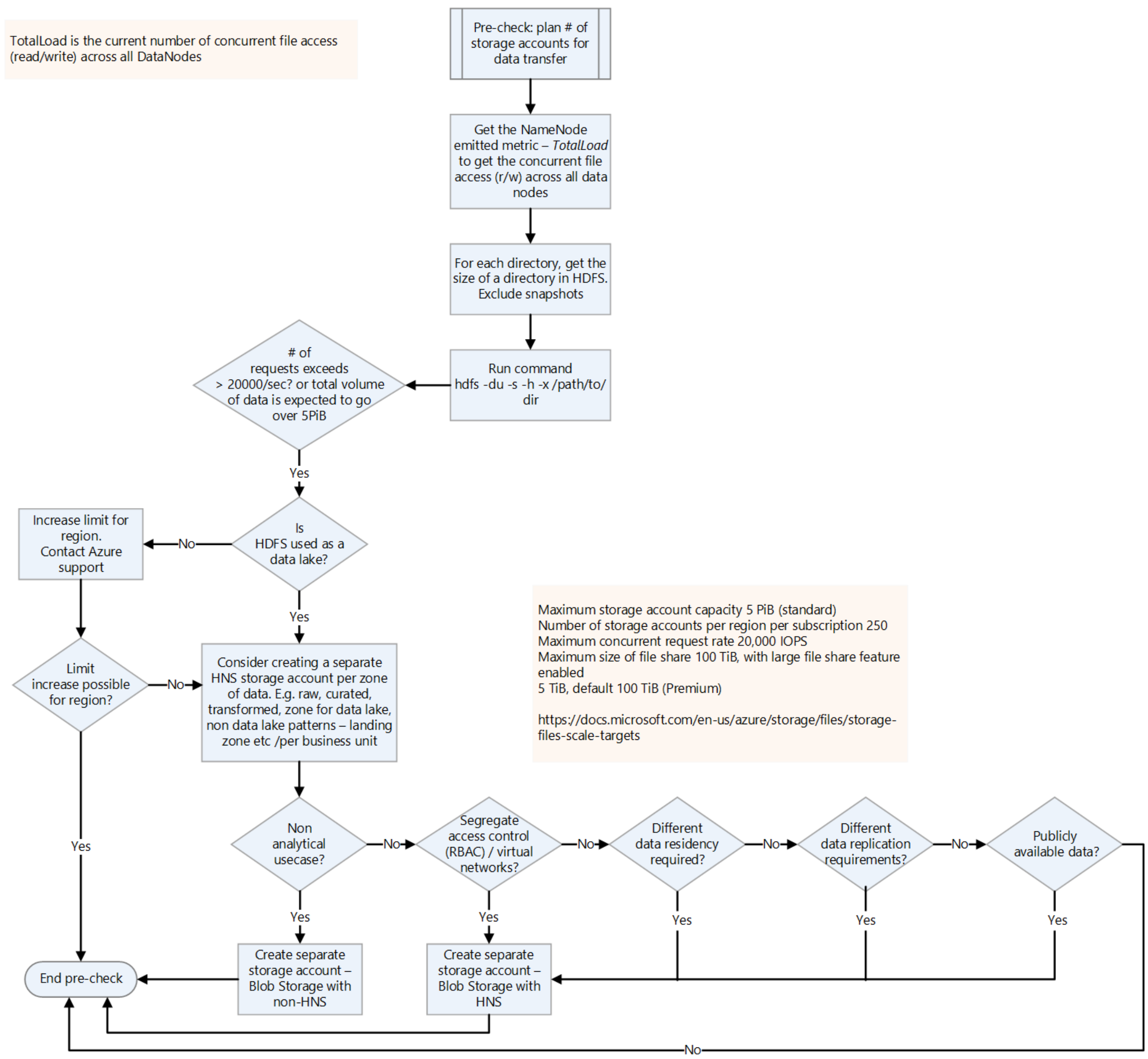
   Common ingestion sources include:
   a. SFTP server
   b. File ingestion
   c. Database ingestion
   d. Database dump
   e. Change data capture
   f. Streaming ingestion

2. **Plan the number of storage accounts required.**

   To plan the number of storage accounts required, understand the total load on the current HDFS. You can use the TotalLoad metric, which is the current number of concurrent file accesses across all DataNodes. Set the limit on the storage account in the region according to the TotalLoad value on-premises and the expected growth on Azure. If it's possible to increase the limit, a single storage account may suffice. However for a data lake, it's best to keep a separate storage account for each zone, to prepare for future data volume growth. Other reasons to keep a separate storage account include:

- Access control
- Resiliency requirements
- Data replication requirements
- Exposing the data for public usage

When you enable a hierarchical namespace on a storage account, you can't change it back to a flat namespace. Workloads such as backups and VM image files don't gain any benefit from a hierarchical namespace.



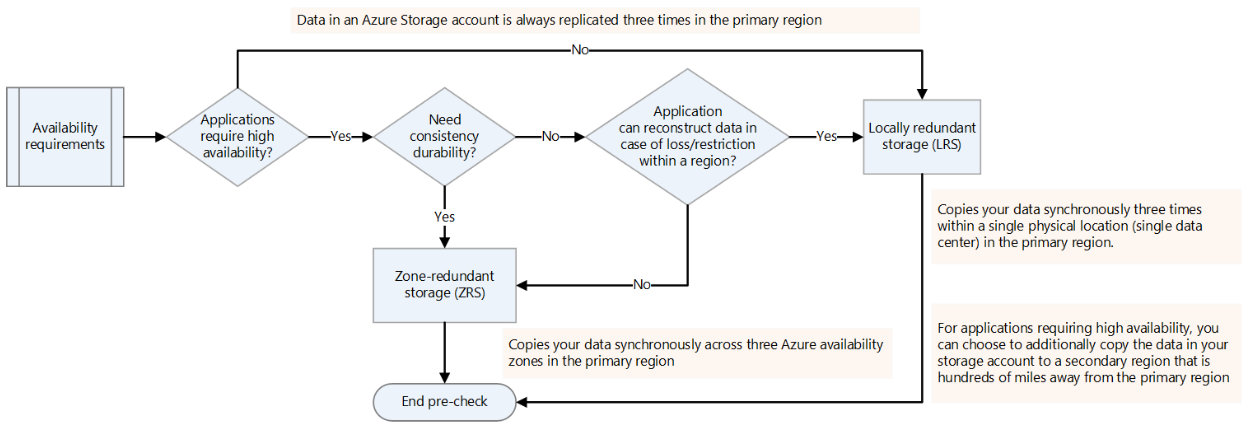
TotalLoad is the current number of concurrent file access (read/write) across all DataNodes
Pre-check: plan # of storage accounts for data transfer
Get the NameNode emitted metric – *TotalLoad* to get the concurrent file access (r/w) across all data nodes
For each directory, get the size of a directory in HDFS. Exclude snapshots
Run command hdfs -du -s -h -x /path/to/dir
# of requests exceeds > 20000/sec? or total volume of data is expected to go over 5PiB
Yes
Is HDFS used as a data lake?
No
Increase limit for region. Contact Azure support
Yes
Limit increase possible for region?
No
Consider creating a separate HNS storage account per zone of data. E.g. raw, curated, transformed, zone for data lake, non data lake patterns – landing zone etc /per business unit
Maximum storage account capacity 5 PiB (standard)
Number of storage accounts per region per subscription 250
Maximum concurrent request rate 20,000 IOPS
Maximum size of file share 100 TiB, with large file share feature enabled
5 TiB, default 100 TiB (Premium)
https://docs.microsoft.com/en-us/azure/storage/files/storage-files-scale-targets
Yes
Non analytical usecase?
No
Segregate access control (RBAC) / virtual networks?
No
Different data residency required?
No
Different data replication requirements?
No
Publicly available data?
Yes
Yes
Yes
Yes
Yes
End pre-check
Create separate storage account – Blob Storage with non-HNS
Create separate storage account – Blob Storage with HNS
No

For information about securing the traffic between your virtual network and the storage account over a private link, see Securing Storage Accounts.

For information about default limits for Azure Storage accounts, see Scalability and performance targets for standard storage accounts. The *ingress limit* applies to the data that's sent to a storage account. The *egress limit* applies to the data that's received from a storage account

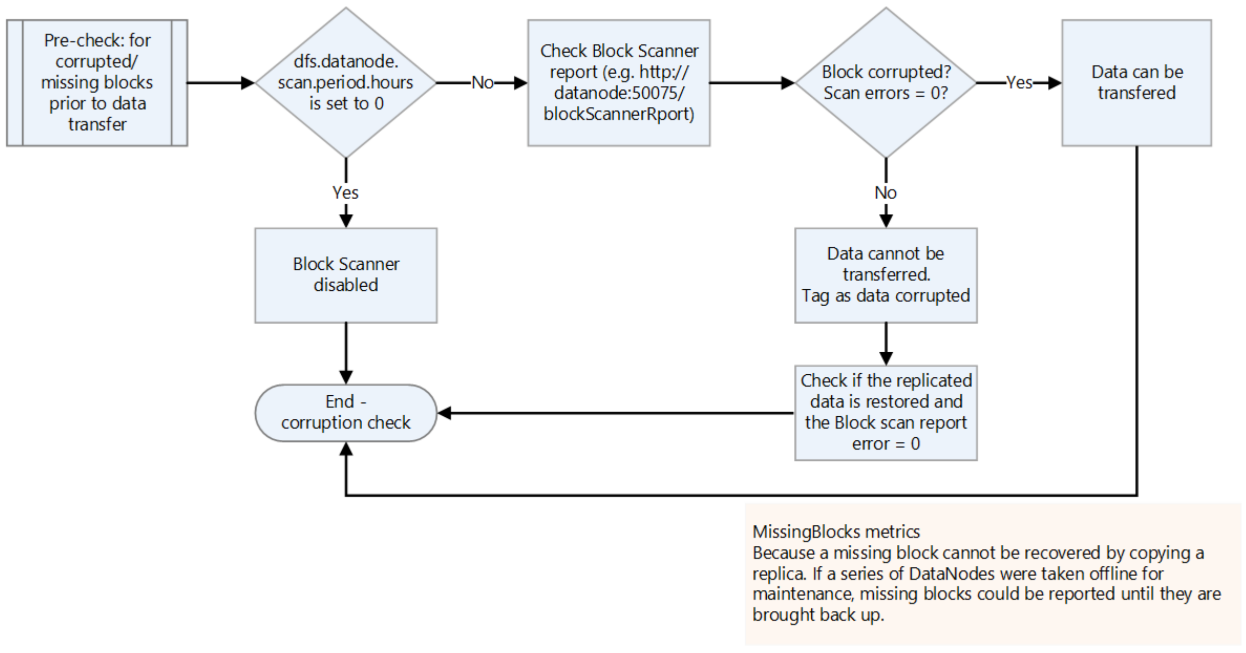
3. **Decide on availability requirements.**

   You can specify the replication factor for Hadoop platforms in hdfs-site.xml or specify it per file. You can configure replication on Data Lake Storage according to the nature of the data. If an application requires that the data be reconstructed in case of a loss, then zone-redundant storage (ZRS) is an option. In Data Lake Storage ZRS, data is copied synchronously across three availability zones in the primary region. For applications that require high availability and that can run in more than one region, copy the data to a secondary region. This is geo-redundant replication.


Data in an Azure Storage account is always replicated three times in the primary region
Availability requirements
Applications require high availability?
Yes
No
Need consistency durability?
No
Application can reconstruct data in case of loss/restriction within a region?
Yes
Locally redundant storage (LRS)
Yes
Zone-redundant storage (ZRS)
No
Copies your data synchronously three times within a single physical location (single data center) in the primary region.
For applications requiring high availability, you can choose to additionally copy the data in your storage account to a secondary region that is hundreds of miles away from the primary region
Copies your data synchronously across three Azure availability zones in the primary region
End pre-check

4. **Check for corrupted or missing blocks.**

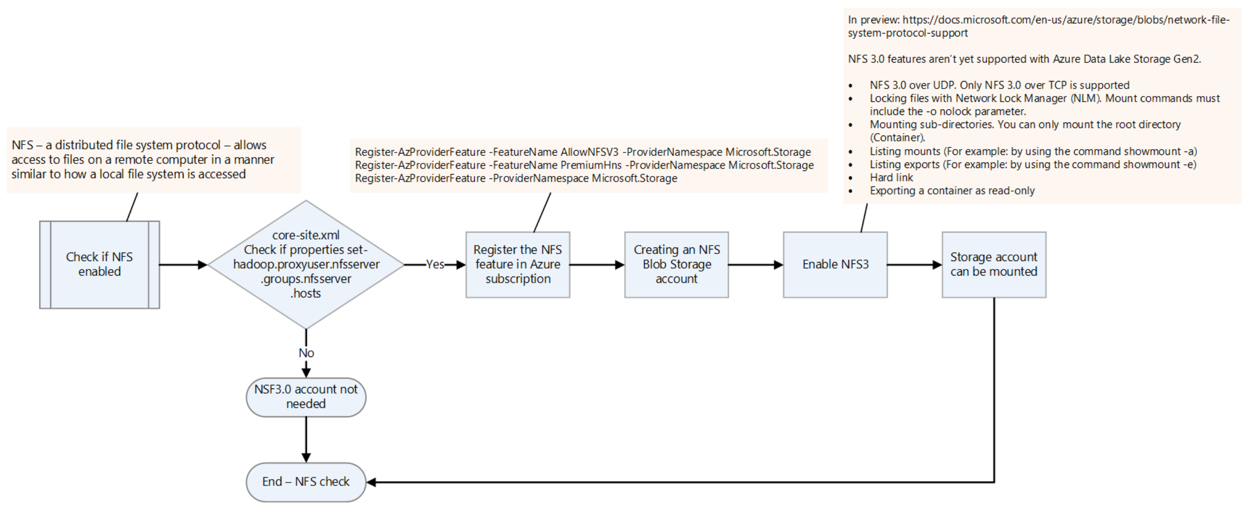
   Check the block scanner report for corrupted or missing blocks. If there are any, wait for the file to be restored before transferring it.


Pre-check: for corrupted/ missing blocks prior to data transfer
dfs.datanode. scan.period.hours is set to 0
No
Check Block Scanner report (e.g. http:// datanode:50075/ blockScannerRport)
Block corrupted? Scan errors = 0?
Yes
Data can be transfered
Yes
Block Scanner disabled
No
Data cannot be transferred. Tag as data corrupted
Check if the replicated data is restored and the Block scan report error = 0
End - corruption check
MissingBlocks metrics
Because a missing block cannot be recovered by copying a replica. If a series of DataNodes were taken offline for maintenance, missing blocks could be reported until they are brought back up.

5. **Check if NFS is enabled.**

   Check if NFS is enabled on the on-premises Hadoop platform by checking the core-site.xml file. It has the nfsserver.groups and nfsserver.hosts properties.

   The NFS 3.0 feature is in preview in Data Lake Storage. A few features may not be supported yet. For more information, see Network File System (NFS) 3.0 protocol support for Azure Blob Storage.


Check if NFS enabled
core-site.xml Check if properties set- hadoop.proxyuser.nfsserver .groups.nfsserver .hosts
Yes
No
Register the NFS feature in Azure subscription
Creating an NFS Blob Storage account
Enable NFS3
Storage account can be mounted
NSF3.0 account not needed
End – NFS check

NFS – a distributed file system protocol – allows access to files on a remote computer in a manner similar to how a local file system is accessed

```
Register-AzProviderFeature -FeatureName AllowNFSV3 -ProviderNamespace Microsoft.Storage
Register-AzProviderFeature -FeatureName PremiumHns -ProviderNamespace Microsoft.Storage
Register-AzProviderFeature -ProviderNamespace Microsoft.Storage
```

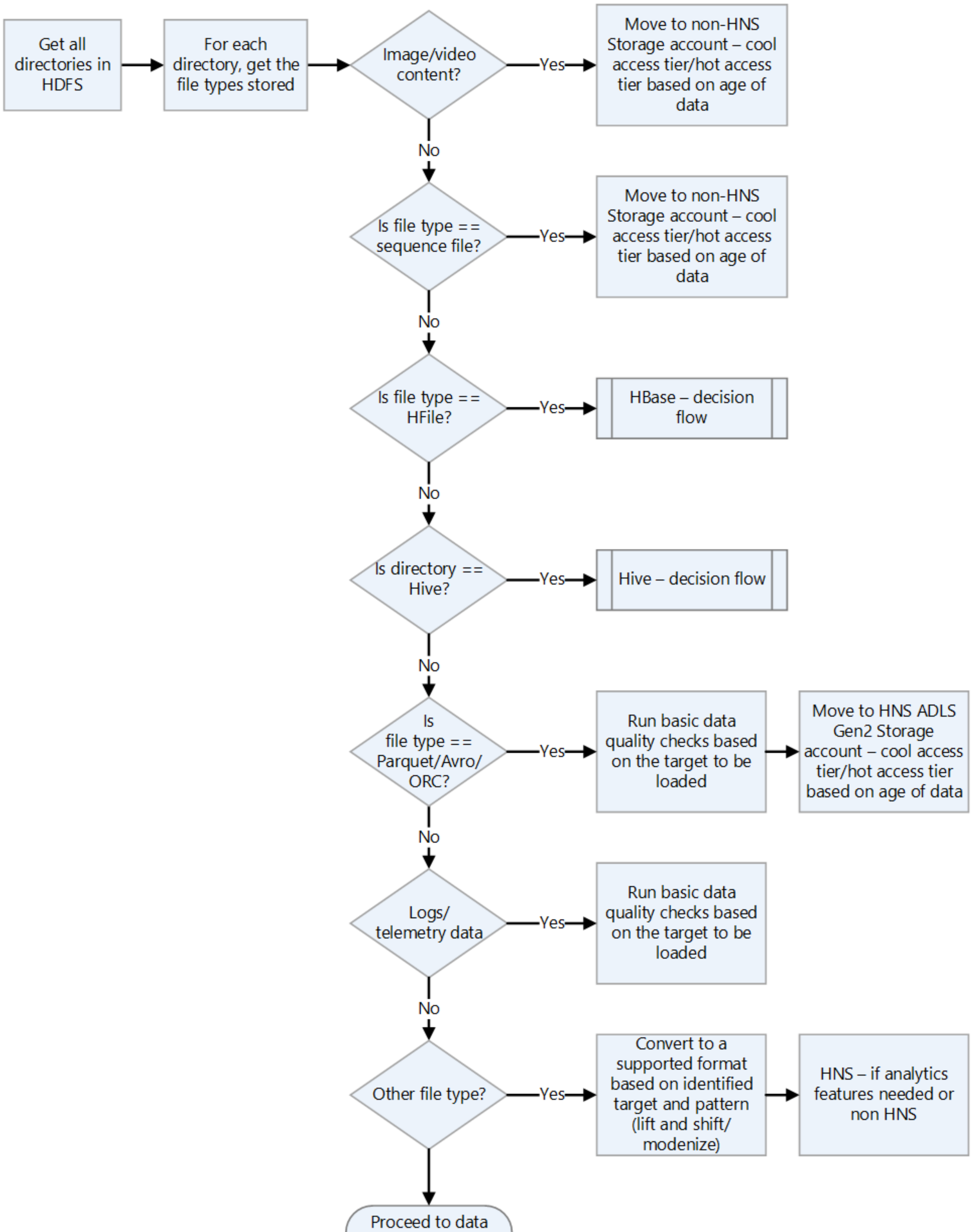
In preview: https://docs.microsoft.com/en-us/azure/storage/blobs/network-file-system-protocol-support

NFS 3.0 features aren't yet supported with Azure Data Lake Storage Gen2.

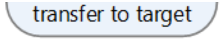
- NFS 3.0 over UDP. Only NFS 3.0 over TCP is supported
- Locking files with Network Lock Manager (NLM). Mount commands must include the -o nolock parameter.
- Mounting sub-directories. You can only mount the root directory (Container).
- Listing mounts (For example: by using the command showmount -a)
- Listing exports (For example: by using the command showmount -e)
- Hard link
- Exporting a container as read-only

6. **Check Hadoop file formats.**

   Use the following decision flow chart for guidance on how to handle file formats.



Get all directories in HDFS
For each directory, get the file types stored
Image/video content?
Yes
Move to non-HNS Storage account – cool access tier/hot access tier based on age of data
No
Is file type == sequence file?
Yes
Move to non-HNS Storage account – cool access tier/hot access tier based on age of data
No
Is file type == HFile?
Yes
HBase – decision flow
No
Is directory == Hive?
Yes
Hive – decision flow
No
Is file type == Parquet/Avro/ ORC?
Yes
Run basic data quality checks based on the target to be loaded
Move to HNS ADLS Gen2 Storage account – cool access tier/hot access tier based on age of data
No
Logs/ telemetry data
Yes
Run basic data quality checks based on the target to be loaded
No
Other file type?
Yes
Convert to a supported format based on identified target and pattern (lift and shift/ modenize)
HNS – if analytics features needed or non HNS
Proceed to data



7. **Choose an Azure solution for data transfer.**

   Data transfer can be online over the network or offline by using physical devices. Which method to use depends on data volume, network bandwidth, and frequency of the data transfer. Historical data has to be transferred only once. Incremental loads require repeated ongoing transfers.

   Data transfer methods are discussed in the list that follows. For more information about choosing types of data transfer, see Choose an Azure solution for data transfer.

   a. **Azcopy**

      Azcopy is a command-line utility that can copy files from HDFS to a storage account. This is an option for high-bandwidth transfers (over 1 GBPS).

      Here's a sample command to move an HDFS directory:

      Bash

      ```bash
      *azcopy copy "C:\local\path" "https://account.blob.core.win-
      dows.net/mycontainer1/?sv=2018-03-28&ss=bjqt&srt=sco&sp=rwd-
      dgcup&se=2019-05-01T05:01:17Z&st=2019-04-
      30T21:01:17Z&spr=https&sig=MGCXiyEzbtttkr3ewJIh2AR8KrghSy1DGM9ov-
      N734bQF4%3D" --recursive=true*
      ```

   b. **DistCp**

      DistCp is a command-line utility in Hadoop that can do distributed copy operations in a Hadoop cluster. DistCp creates several map tasks in the Hadoop cluster to copy data from the source to the sink. This push approach is good when there's adequate network bandwidth, and it doesn't require extra compute resources to be provisioned for data migration. However, if the source HDFS cluster is already running out of capacity and additional compute can't be added, then consider using Data Factory with the DistCp copy activity to pull rather than push the files.

      Bash

      ```bash
      *hadoop distcp -D fs.azure.account.key.<account name>.blob.-
      core.windows.net=<Key> wasb://<container>@<account>.blob.-
      core.windows.net<path to wasb file> hdfs://<hdfs path>*
      ```

   c. **Azure Data Box for large data transfers**

      Azure Data Box is a physical device that's ordered from Microsoft. It provides large-scale data transfers, and it's an offline data transfer option when network bandwidth is limited and data volume is high (for example, when volume is between a few terabytes and a petabyte).

You connect a Data Box to the LAN to transfer data to it. You then ship it back to the Microsoft data center, where the data is transferred by Microsoft engineers to the configured storage account.

There are multiple Data Box options that differ by the data volumes that they can handle. For more information about the Data Box approach, see Azure Data Box documentation - Offline transfer.

d. **Data Factory**

Data Factory is a data-integration service that helps create data-driven workflows that orchestrate and automate data movement and data transformation. You can use it when there's sufficient network bandwidth available and there's a requirement to orchestrate and monitor data migration. You can use Data Factory for regular incremental loadings of data when the incremental data arrives on the on-premises system as a first hop and can't be directly transferred to the Azure storage account because of security restrictions.

For more |information about the various transfer approaches, see Data transfer for large datasets with moderate to high network bandwidth.

For information about using Data Factory to copy data from HDFS, see Copy data from the HDFS server using Azure Data Factory or Synapse Analytics

e. **Partner solutions such as WANdisco LiveData migration**

The WANdisco LiveData Platform for Azure is one of Microsoft's preferred solutions for migrations from Hadoop to Azure. You access its capabilities by using the Azure portal and the Azure CLI. For more information, see Migrate your Hadoop data lakes with WANdisco LiveData Platform for Azure .

Contributors

This article is maintained by Microsoft. It was originally written by the following contributors.

Principal authors:

- Namrata Maheshwary | Senior Cloud Solution Architect
- Raja N | Director, Customer Success
- Hideo Takagi | Cloud Solution Architect
- Ram Yerrabotu | Senior Cloud Solution Architect

Other contributors:

- Ram Baskaran | Senior Cloud Solution Architect
- Jason Bouska | Senior Software Engineer

- Eugene Chung | Senior Cloud Solution Architect
- Pawan Hosatti | Senior Cloud Solution Architect - Engineering
- Daman Kaur | Cloud Solution Architect
- Danny Liu | Senior Cloud Solution Architect - Engineering
- Jose Mendez Senior Cloud Solution Architect
- Ben Sadeghi | Senior Specialist

- Sunil Sattiraju | Senior Cloud Solution Architect
- Amanjeet Singh | Principal Program Manager
- Nagaraj Seeplapudur Venkatesan | Senior Cloud Solution Architect - Engineering

To see non-public LinkedIn profiles, sign in to LinkedIn.

Next steps

Azure product introductions

- Introduction to Azure Data Lake Storage Gen2
- What is Apache Spark in Azure HDInsight
- What is Apache Hadoop in Azure HDInsight?
- What is Apache HBase in Azure HDInsight
- What is Apache Kafka in Azure HDInsight

Azure product reference

- Microsoft Entra documentation
- Azure Cosmos DB documentation
- Azure Data Factory documentation
- Azure Databricks documentation
- Azure Event Hubs documentation
- Azure Functions documentation
- Azure HDInsight documentation
- Microsoft Purview data governance documentation
- Azure Stream Analytics documentation
- Azure Synapse Analytics

Other

- Enterprise Security Package for Azure HDInsight
- Develop Java MapReduce programs for Apache Hadoop on HDInsight
- Use Apache Sqoop with Hadoop in HDInsight
- Overview of Apache Spark Streaming
- Structured Streaming tutorial

- Structured Streaming tutorial
- Use Azure Event Hubs from Apache Kafka applications

Related resources

- Hadoop migration to Azure
- Apache HBase migration to Azure
- Apache Kafka migration to Azure
- Apache Sqoop migration to Azure

Feedback