Cost Effective Cloud Storage Interoperability Between Public Cloud Platforms

Yassine Serhane, Abderrahim Sekkaki, Karim Benzidane and Mehdi Abid

Computer Science Department, Laboratory of Research in Computer Science and Innovation
Faculty of Sciences Ain Chock, University Hassan II Casablanca, Morocco

***Abstract*:** With recent advancement in technology, cloud storage became cheaper enabling organizations around the world to store more data on the cloud (texts, images, videos, databases etc.), whereas it's for a backup, archiving or just storing data streams. New digital laws and regulation (e.g., General Data Protection Regulation) require these organizations to change their way of processing or handling data, which results usually in a change of cloud providers or adoption of hybrid architecture or multi-cloud one. With the amount of data stored increasing year after year, it becomes difficult for these organizations to change cloud platforms or cloud provider and migrate their data without thinking about the technical complexity, the time, and the huge cost it may incur.

This article discusses the data migration and interoperability issues between cloud platforms; the proposed approach provides a simple cost-effective migration that would help organizations save time and money in this process based on a hybrid ontology approach for the brokerage of data transfers.

***Keywords*—**Cloud Computing, Storage, Security, Ontologies, Data, Interoperability, Migration, Cost Optimization**.**

1. Introduction

Technology has evolved in a way to facilitate the accessibility and consumption of IT resources as a service provided by another entity, thus, giving customers the opportunity to request and use a small or huge infrastructure outside of their premises. Cloud computing has made these infrastructures globally available over the internet, making it easy to share data, applications, and highly available services, to be reachable from anywhere and anytime, while being agnostic to the underlying devices being used to consume these resources (computer, tablet, smartphone, IoT device, etc.). The use of cloud computing allows information sharing at any time through cloud services based on configurable IT resources (servers, applications, networks, storage...) which ease the accessibility to any user. These cloud services may be fitted and defined in three models [1]:

Infrastructure as a Service (IaaS) [2], services that make the base infrastructure, such as servers, networks, and storage, virtualized and make infrastructure resources available as needed.

Platform as a Service (PaaS) [3], services that provides an integrated platform for users to develop, test and build their applications.

Software as a Service (SaaS) [4], Service provided as an app hosted on the cloud to be consumed by users and doesn't require maintenance from their perspective.

With cloud becoming more conventional, cloud storage became cheaper and affordable for companies to store more data and offload their archives and backups into the cloud. Additionally, some other entities have been given the ability to collect more data, like manufacturers connecting all their factories machinery and ingesting a big amount of data through IoT, or Retail industry actors collecting consumers data and interests for the purpose of serving them better using the power of machine learning and building product recommendation models for example. Customers became dependent on this type of storage, since storing Peta bytes of data and archives became cheaper, until they are faced with the need to move from one cloud provider (CP) to another, this struggle is what makes organizations get locked-in to only one CP, due to the effort, time, and cost required to make the change.

Many solutions exist to help customers migrate, but they still come at a pricy cost. Our motivation in this article is to provide a cost-effective approach delivering a storage multi-Cloud interoperability to help organizations migrate from one CP to another in a short amount of time and low investment.

2. Background

Cloud computing – which we are going to refer to in our study as the *cloud* is considered an IT platform/environment which allows a user to request IT resources per their needs or consume resources to answer an organization's specific requirements and use these resources at request, as needed and at any time. It is divided into deployment models based on the location and the operational standards of the infrastructure, but also depending on the way the users are able to access these cloud services, which are Public Private and Hybrid deployment models:

Public cloud: Services which are provided by a "cloud vendor" to be used by anyone and is provided to general users or large corporations in a form of pay as you go model depending on the services usage [2], [5] . The infrastructure of the public cloud is owned by the CP that sells the service. However, the data/applications hosted on it are owned by user.

Private cloud: Services which provide a cloud computing experience accessed by a specific organization and implemented in a closed environment. The private cloud infrastructure is hosted and managed by the organization itself or a third party. However, the organization has a complete ownership over its workload and data [5], [6].

Hybrid cloud: Concept that enables an organization to use both the private and public cloud benefits, used usually to respond to compliance, regulatory as well as latency requirements [3], [4], [7], [8]. The public cloud infrastructure is owned by the CP. As for the private cloud, the organization has control and ownership over their data.

In our study, we focused on the public cloud as it's the most common in the industry, the approach can also be applied on a hybrid cloud concept if it complies with our requirements – and according to Gartner, 90 percent of organizations will adopt the hybrid cloud concept by 2020 [6],[9].

From a Total Cost of Ownership calculation (TCO) [7], [8],[10], [11] the cloud remains the most economical way for organizations to operate, either they have their own infrastructure and need to expand through additional resources, launching a new modern project or service, modernizing their current infrastructure to adopting digital transformation pillars. Additionally, this technology allows each of these organizations to adjust their infrastructure depending on their needs and to optimize considerably the costs and investments required. From a financial point of view, companies are moving from Capital Expenditure (CAPEX) to Operation Expenditure (OPEX).

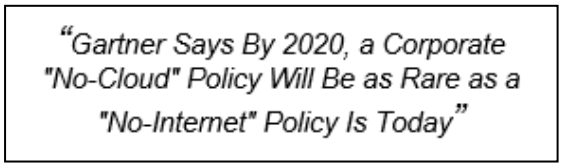

Figure 1. Quote from Gartner about the future of cloud [9], [12]

However, with the IT evolution and cloud becoming a must – growing year after year – it allowed itself to prevail against other type of infrastructures and became the basis for industrial growth and technological implementation.

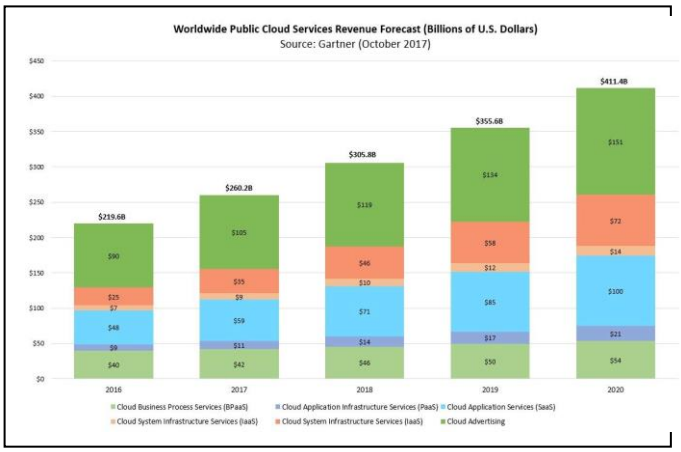

Figure 2. Gartner Forecasts Worldwide Public Cloud Services Revenue to Reach $260 Billion in 2017 published October 12, 2017

CPs are constantly improving and bringing innovation to their services in order to differentiate themselves, it results in a competition between the providers, leading to many benefits for the customers (new services and lower prices). As they assess the best platform continuously, customers consider changing their CP, however, they are quickly discouraged when they evaluate the critical and humongous mass of data, they stored on it, and the challenges associated with data transfer.

Changing a cloud provider or platform implies deep study on what would be the best and optimal options to make the change happen, keeping the service disruption, the time to move them and the cost of transfer at a minimum. Organizations would be looking at Interoperability in four main areas [12]:

Identity and Access Management, IAM: user identification and authentication. Ability to use same identity on another CP and enterprise applications, one of the most known concepts is Identity Federation with Single Sign On, SSO, being the most used method, which enables IT Systems to trust and authenticate a User electronically identified through a single authentication token.

Workload Migration: Ability to migrate a workload (e.g., IaaS or PaaS) and execute it on another CP. An example, migrating a Virtual Machine (VM) from AWS to Google Cloud Platform (GCP) is not a straightforward process, as the VM configuration is different between both CPs, it would require that we stop the source VM, detach the storage disk attached, create a VM on GCP with similar attributes, then migrate the storage disk data to GCP, and then attach it the newly created VM [13].

Storage and Data Migration: Ability to migrate data from one CP to another in a secure way while keeping the data-integrity, data might be residing in a Database, Data Lake, a blob storage etc. when migrated it needs to remain readable and accessible as per the destination access configuration.

Workload management: Ability to use workload management tools on different clouds, this scenario is typical to multi-cloud concepts where an organization is managing its workloads on multiple cloud platforms and needs a centralized tool to manage them all.

On the Storage and Data Migration, cost of transfer is often a huge concern when it comes to public cloud as the CP charges for bandwidth usage for the egress data leaving the Data Center (DC). This means that a company can upload data as much as they want/need to the public cloud DC and be charged $0 but downloading it would be charged egress fees depending on the size downloaded.

2.1 Cloud Storage

Storage in the cloud world is managed differently than on premise. The most used storage services are based on **object storage services**, also known as object-based storage. It is a storage architecture that manages unstructured data as objects, to be stored beside each other in a pool. Unlike file systems, which manage data as a file hierarchy, and block storage which manages data as blocks within sectors and tracks.

A Generic cloud storage would translate to a cluster of storage devices and distributed file systems underneath a storage middleware, Microsoft Azure Blob Storage for example is built on top of HDFS.

Throughout the major CPs, we will find multiple supported traditional protocols, like file-based APIs, NFS, CIFS, HDFS or SOAP, and recently some customers would use cloud storage gateways like StorSimple and access cloud storage through on-premises appliances with REST-based access.

For years, the cloud has been known as a perfect place to store big data, even though enterprises were reluctant to put their data on the cloud since security and privacy concerns

outweighed the benefits they would get from it. Today cloud storage evolved and become much secure, enterprises overcame these issues by aiming to transform while they perform and strive to operationalize. Additionally, cloud storage is helping them comply with regulations, through many services e.g., Disaster recovery, data redundancy, Availability etc.

2.2 Accessing data from cloud providers

With all data flowing in cloud storage, customers need a way to access them in a simple way. Cloud services based on object storage, such as Azure with Azure Storage, Amazon with Amazon S3 or OpenStack's Swift, offer their users web-services APIs (REST or SOAP) to access their data as if it were residing on a disk, depending on the size of it the requester would get quick access to it. There are other cloud storage services specific to each vendor for e.g., for archiving purpose like Azure Cold Storage or Amazon Glacier where the access process would take time as it is designed to be an archival solution with an infrequent access.

The method of data access varies from CP to another. It's based on the type of APIs they support and type of data a user is storing (e.g., SOAP, WebDAV etc.).

2.3 Data Transfer

Transferring a file or data in general to the cloud is different than the on-premise process, where the transfer can be done with known standardized protocols [14] whether it is within the same machine, LAN, or inter-LAN networks connected via VPN, by hard drive etc.

In the on-premises world, an IT person generally when transferring data would consider the size of the data and the bandwidth speed which impacts the time required to complete this task. In the cloud world, a lot of parameters would come in the play, especially if dealing with a public CP, as an important cost parameter would be added in the equation. Bandwidth usage in a cloud environment is charged by amount of data transferred, and it only applies on the outbound data for most of the providers. Uploading Terabytes of data won't incur any fees, but the moment one will start downloading or retrieving any of it from the cloud storage, the billing begins. Moving data within the same datacenter (DC) would not incur anything, but from a DC to another one, bandwidth fees will apply as the data goes out of a DC and is considered outbound/egress.

2.4 Storage Challenges

Moving data to the cloud is not a straight forward process for its users [15], customers, especially established ones, face many challenges to consume the service. Most of these challenges are security related, "how to ensure Data security in the cloud?". Tackling this issue becomes a priority for the users, as all kinds of data are being stored, from public to confidential and classified, and it needs to be available when needed. While availability is an important factor, other elements are as important as well, the list includes but is not limited to: Trust, Data sovereignty, Data History, and the agreement between the provider and consumers:

2.4.1 *Trust*

Trust in Data stored in the cloud, is not only about data confidentiality but data integrity, accuracy and availability, where, due to internal errors or changes in the provider's systems, data might get compromised and still be delivered to the customers, leading to a severe business loss before being uncovered, not to mention from a security perspective, the CP needs to ensure an enterprise grade security against hackers [16], there are also cases where the DCs due to natural disaster become inaccessible, while data and cloud services need to remain available at all time for companies to operate. To mitigate these issues, CPs would come up with various techniques and solutions. For e.g., Microsoft Azure approached this issue, by using data replication as a solution [17]. Thus, any data hosted on their cloud storage service will be replicated within multiple datacenters and regions.

Failure to comply with these elements would come with consequences, hence the need to have an agreement between the service providers and the users to protect both parties.

2.4.2 *Cloud Vendor Agreement*

As security gained priority [18] in CPs selection, it started being positioned as a differentiator between the competition, thus, pushing CPs to race for market needed certifications and compliance (e.g. Cloud Security Alliance Security, Trust & Assurance Registry, CSA STAR or PCI DSS certification used widely in the banking and fintech industry). Several organizations need to comply with these local regulations or industry certifications to operate, they rely on their agreement with the CP to ensure the services they are using, and data location are compliant.

Retrieving data from the cloud would always raise questions around its integrity, and how to trust the data stored on the service provider. Though it is possible to perform an integrity check, it won't give any insight about how the data was stored or where was it stored in the datacenter. This is a trade/internal secret that is well kept by the CPs to ensure security of their operation and agreed upon with the customers.

These agreements exist to protect both providers and consumers, since that we are never safe from a disaster. Major Public Cloud outages happened for renown providers causing huge losses for compagnies data and businesses.

2.5 Cloud Vendor Lock-in

The main challenge of vendor lock-in remains on the dependency of one's application or software to use specific services in a CP, when the move to another cloud is needed, it won't be a straightforward process and would incur, a migration cost, legal constraint and technical incompatibility that will probably lead to code refactoring and/or rearchitecting.

This is due to the lack of normalization and standardization between CPs [12], hence majority of the solutions are tackling these issues from a technology aspect, a famous example would be from an infrastructure perspective, the containerization of applications, and more importantly the use of Kubernetes as a cloud agnostic orchestrator [19].

Another type of vendor lock-in is related to PaaS types. PaaS services are easy to deploy and help organizations be more agile in releasing products, features or getting insights out of their data. However, adding a PaaS service makes the solution dependent to a certain CP (Amazon AWS, Microsoft Azure, etc.), thus, moving the workload to another provider will depend on the existence of similar PaaS services required to complete the task. If not, a refactoring is necessary in these cases due the heterogeneity of the application program interfaces (APIs), which creates a technical conflict, leading to inter cloud interoperability issues.

3. Related Work

Most of the papers around cloud interoperability are looking into IaaS services and intercloud architecture, while literature on cloud storage interoperability is scarce.

The Authors in [20] presented a model driven approach for Cloud-to-Cloud Interoperability, C2CI by defining a model of 5 levels to assess the interoperability maturity, L0 Domain-based, L1 Enterprise-based, L2 Portability interoperability, L3 Security interoperability, L4 – Mobile interoperability in a public, private, or hybrid cloud environment. It's an adaptation of the Levels of Information System Interoperability, LISI maturity model, additionally in their paper they described some of the challenges in achieving interoperability intercloud.

In [21], the authors provided some basis on the interoperability in cloud, by identifying in the existing literature how multiple research done in the field of interoperability addressed the matter, they have surveyed several literature for up to 2016 highlighting the solutions. One solution is the standardization of cloud services with CPs adhering to the initiatives, another one is building Abstraction Layers, which consists of abstracting common feature of CPs, an interesting approach is the Cloud Storage Abstraction Layer, CSAL proposed by Hill and Humphrey [22] to provide common storage abstraction layer between clouds.

Alternatively the authors in [23], tackled the PaaS interoperability with a Model Driven Engineering (MDE) approach, proposing a middle platform that decouples a PaaS hosted Application and its databased into two layers and ports it to another PaaS platform. MDE approach proved to be a better alternative to standardization a per literature, they leveraged it to make the middleware flexible, further making application layer and DB layer portable, productive and reusable.

Although these papers are relevant, we thought it would be important to go beyond the simple service interoperability and cover the aspects of the storage that matters to most cloud users, the complexity, the cost, and the time needed for data transfer.

4. Our Approach

For the purpose of this research, we designed an architecture based on ontologies to solve the storage interoperability between two cloud platforms, we choose to focus first on public cloud platforms for their transparency, API documentation, availability, ease of access and setup.

Each public cloud platform has its own way of storing data and own APIs.

Therefore, an ontology-based architecture could be used in this case to create an abstraction layer that can unite Cloud APIs [24]. The use of ontologies solved many issues in interoperability in the IT world. The Hybrid Ontologies (Fig. 1) approach seems fitting well in our scenarios and design, where, a global ontology regroups local ontologies, and each of these local ontologies has its own knowledge base which helps it communicate with a specific entity [25], [26]. The global ontology would act as a broker for all data requests and redirect to the right local ontology depending on the targeted cloud. We named our approach, the Storage Broker Service, SBS.

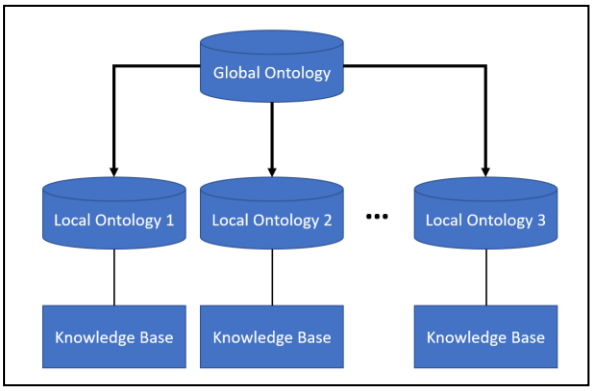

Figure 3. Generic architecture for Hybrid Ontologies.

The following conceptual design, Fig. 4, showcases the proposed approach for SBS, which is based upon the previous hybrid ontologies shown on Fig. 3. This architecture can be mapped on multiple public clouds of choice.

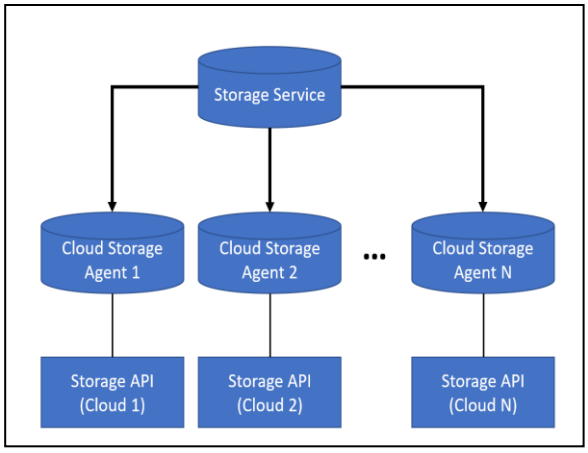

Figure 4. Hybrid Ontology architecture with SBS approach.

From Fig. 4, we can infer that a major requirement for this concept to work is the existence of a Storage Service API for the selected CP, which most of the public clouds offer.

4.1 Storage Service (Global Ontology)

The Storage Service, acting as the Global Ontology, is a brokering service that will be managing the storage requests, whether we need a file to be sent from a public Cloud 1

Storage to a public Cloud N Storage and vice-versa, depending on the nature of the request (CREATE, READ, UPDATE, DELETE) as well as the platform targeted, the broker will call out for the right agent (local ontology) to execute the request.

4.2 Cloud Storage Agent (Local Ontology)

The Cloud Storage Agent is the entity responsible of executing the requests coming from the Storage Service on its respective platform using the linked Storage API (Knowledge base).

4.3 Storage API (Knoweldge Base)

Storage API would hint to the Storage Service API for the targeted public Cloud, the Storage Agent will refer to it to fetch the right API for a defined action (CREATE, READ, UPDATE, DELETE).

4.4 Excecution Process:

When it comes to applying this architecture between two or multiple public clouds, we would investigate, (1) the process and method for transferring the data from one point to another, and (2) Can we reduce the cost of transfer.

4.4.1 Transferring Data process:

When a file or a data needs to be moved from one platform to another, in many cases there is no direct way to do it, data needs to be first downloaded in a temporary disk storage attached to the agent, as a buffer, then erased from the source, and written to the destination, then finally removed from the temporary storage, while preserving the integrity and authenticity of the data throughout the process.

4.4.2 Cost Optimization : locating the nearest DC

Optimization in a migration is about reducing the time needed for migration as well as reducing the operational fees related to it (data transfer, read write operations etc.). During a data transfer, a customer is charged for two main components, the data read operations requests and the data egress with the latter being the most expensive, then comes the non-billable cost of time.

To reduce the cost of migration, we had to think about (1) reducing the Time To Destination, TTD, (2) reducing the cost of outbounds charges to destination to proceed with the migration task, downloading data on premises would clearly help reducing the outbound charges but impact the TTD as the bandwidth on premises wouldn't be limited by the internet provider only, but the organizations hardware and upload policies, etc. from a time perspective, the best outcome would be keeping the migration Cloud to Cloud, with the broker being the destination cloud and knowing that data transfers within the same DC are not charged.

The solution would be to create a migration agent right where the data needs to be migrated to, in order to avoid being charged twice on the data-out (egress) but also reduce the time needed for data to travel, as Cloud to Cloud transfer would offer a better networking bandwidth.

5. Experiment: Proposed system

Based on the previous high-level architecture design, we proposed a solution (SBS) which implements techniques that helps migrate data securely from one cloud to another in a way to preserve its integrity.

We started by selecting the CPs we were going to work with and knowing that many public cloud platforms exist we decided to focus on only two for the purpose of this experiment, Microsoft Azure and Amazon Web Services (AWS) as they are both widely spread worldwide, and they both have well documented public APIs to manage and access their services.

Implementing the SBS approach based two public cloud platforms, Azure and AWS, we infer the below architecture, Fig. 5.

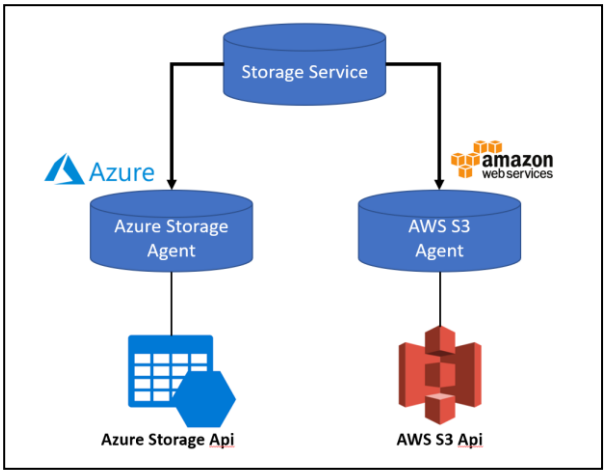

Figure 5. SBS approach applied for two public cloud interoperability (Azure, AWS)

5.1 Storage Service

Storage Service in the SBS architecture, is defined as a web-based application, with a UI to give users a control over files to move or copy, by sending requests to either one of the storage agents.

5.2 Storage Agent

The CP's Storage Agent can either be a script or a console application that will execute a specific command using one of the providers' API.

5.3 Storage API

In our case, would refer to the providers' storage API respectively (Azure Storage and Amazon S3). The storage service will be calling the CP's agent to fetch the right API for a defined/requested action.

5.3.1 Azure Storage API

Azure storage, (formerly Windows Azure Storage, WAS) is a highly available cloud storage system widely used by Microsoft to provide its service to the public, like Xbox Live, OneDrive, Skype etc. Many customers use it to host their own services too, since it has several forms of cloud storage, like Blobs, Tables, message queues and Disks, and Queues

(message delivery)[10], [18], It has also an extension based on HDFS, making it possible to use it as a Data Lake service.
The .NET SDK for Azure Storage Service is easy to use, and documented, the below code shows that few lines are required to begin writing a blob after calling the library, Fig. 6.

```
private string storageConnectionString=
    "DefaultEndpointsProtocol=https;"
        + "AccountName=[Storage Account Name]"
        + ";AccountKey=[Storage Account Key]"
        + ";EndpointSuffix=core.windows.net";
CloudStorageAccount account =
    CloudStorageAccount.Parse(storageConnectionString);
CloudBlobClient serviceClient =
    Account.CreateCloudBlobClient();

// Create container. Name must be lower case.
Console.WriteLine("Creating container...");
var container =
    serviceClient.GetContainerReference("mycontainer");
container.CreateIfNotExistsAsync().Wait();

// write a blob to the container
CloudBlockBlob blob =
    container.GetBlockBlobReferenc("myfile.txt");
blob.UploadTextAsync("text in a file").Wait();
```

Figure 6. Sample code for writing and editing a blob.

An access through REST APIs exist to get/put/delete a file/blob on Azure. The requests must be executed to the following HTTPS link (Fig. 7) which has 3 parameters, *myaccountName*: the name of the storage account, *mycontainerName*: the container inside of the storage account, and *myBlobName*: the name of the blob we are trying to access.

```
Host : https://{myaccountname}.blob.core.windows.net
    /{mycontainerName}/{myBlobName}
```

Figure 7. Request URL for Azure Storage API

To execute the request URL (Fig. 7) based on the request types, simply use the adequate http method.

Table 1. HTTP Methods used by our agent for Azure calls

Request type	Http method
Read	GET
Update	PUT
Remove	DELETE

5.3.2 AWS S3 API:

Amazon Web Service (AWS) also has REST API for its storage service S3. In contrast with Azure storage, it is about the target API, where *ObjectName* refers to the file we are accessing and *BucketName* the storage account.

Table 2. HTTP Method and Targeted host address used for AWS S3 service calls

Request type	Http method	
Read	GET	[GET/PUT/DELETE] /*ObjectName* HTTP/1.1
Update	PUT	Host: *BucketName*.s3.amazonaws.com
Remove	DELETE	

5.4 Solution's Application Architecture

For this system to be functional, we needed a user interface for the controls which would implement the Storage Service capabilities. For this matter, we used a MVC model.
As seen previously, the solution should have the knowledge base of both public clouds to be able to communicate between them, Additionally, we should keep in mind that the migration service/application should be hosted on the web. For the purpose of our testbed, the best location for hosting the service would be on either one of the clouds we chose and, in our case, for familiarity with the service we hosted the solution on Azure, with the agent being deployed dynamically as needed on a VM, it can also be deployed on a container, or as a serverless function (e.g., AWS Lambda, Azure Function App). Given that the cloud gives us the ubiquity to host data anywhere in the world, – i.e., create a storage container in any DC available – we implemented a feature that selects the best and cheapest cloud DC to which we want to migrate data to. Below is the architecture used for the UI of our testbed based on an MVC design pattern, Fig. 8.

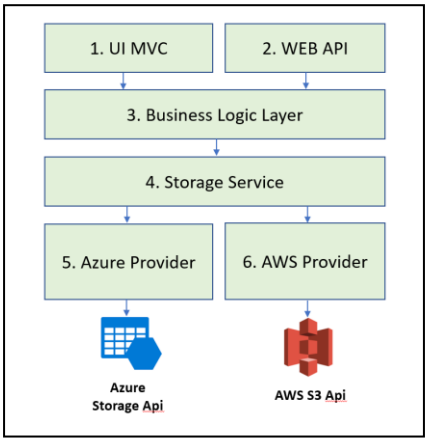

Figure 8. Proposed MVC architecture for UI/Control application architecture

1. *UI MVC*: Front end web application for solution control.
2. *WEB API*: Web services to query data from other tools.
3. *Business Logic Layer*: Layer that contains all business logic and rules to as per the MVC pattern to control the app.
4. *Storage Service*: Abstraction composed mainly of interfaces and Object for Data transfer and files management.
5. *Azure Provider:* Implementation of the Azure Storage Service
6. *AWS Provider*: Implementation of the AWS Storage Service

5.5 A secure cloud service

Data flowing over the internet can be intercepted, read, and altered, therefore, we are securing our transfer by using a SSL secured channel and making sure the data is encrypted at rest, as well as verifying its integrity. An efficient way to verify file

integrity is to check the data's Hash Code –if compromised even the slightest– the hash would change; it will set the file integrity to null and the process to be redone.
By using either Azure or AWS for hosting the migration agent, we benefit from their secure platform, data replication, and integrity check, we can then add another manual check on the application side to verify the Hash Code.

5.6 Cost Optimization : locating the nearest datacenter

To enable and implement this feature, we need to identify the destination DC's location, each CP has a public list of region locations for each of their DC, with AWS having more than 24 regions covered [27], and Azure, more than 60 regions covered [28] at the time of writing this article.
Destination's location can be inferred from a CP's service, in our case storage service, through the property Location with a simple GET request using one of the below APIs depending on the destination CP.

Table 3. Azure and AWS HTTP Requests to identify services locations

Http method	HTTP Request
Azure	https://management.azure.com/subscriptions/*SubscriptionId*/resourceGroups/*ResourceGroupName*/providers/Microsoft.Storage/storageAccounts/*AccountName*?api-version=2019-06-01
AWS	Http://*BucketName*.s3.amazonaws.com/?location

Where *SubscriptionId* refers to the Azure Subscription ID, *ResourceGroupName* refers to the Resource Group containing the Account Storage and *AccountName* is the Azure Account Storage name, and *BucketName* the AWS Storage Bucket Name. A successful response from both services would result in the following (Fig. 9, Fig. 10)

```
{
    "value": [
        {
            "id": "/subscriptions/{subscription-id}/resourceGroups/testagent13/providers
            "identity": {
                "principalId": "b2e5457d-416b-91ab-cba5-b2e547bc44dd",
                "tenantId": "2d7c88bf-db47-8a30-41af-2d7cd01186f1",
                "type": "SystemAssigned"
            },
            "kind": "Storage",
            "location": "northeurope",
            "name": "agentstor014a3",
            "properties": {
                "creationTime": "2020-08-03T10:06:30.6093014Z",
                "primaryEndpoints": {
                    "blob": "https://agentstor014a3.blob.core.windows.net/",
                    "file": "https://agentstor014a3.file.core.windows.net/",
                },
                "primaryLocation": "northeurope",
            },
            "type": "Microsoft.Storage/storageAccounts"
        } //.... other storage services
    ]
}
```


Figure 9. Azure API response for service location

```
<LocationConstraint
xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
us-east-2
</LocationConstraint>
```


Figure 10. AWS API response for service location

5.7 Migration Process

Once DC destination region is identified, we then proceed to create the agent VM within that same region with an automated process, as all processes are digitalized, we needed to create this agent VM dynamically and using code only.

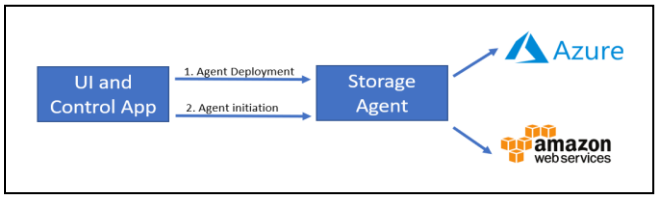

Figure 11. Application interaction with the agent and the cloud platforms

The SBS Control Application initiates the Agent deployment and migration job run in sequence and automatically as seen in Fig. 11, we opted for Infrastructure as Code, IaC tools, as a way to manage the infrastructure (VMs, networking, Storage, etc.) in a descriptive model, enable the versioning of configuration files the same way developers use for source code, with one difference, instead of releasing the same binary, we generate the same infrastructure environment when the job is executed.
We assessed the open-source world for solutions to automate the infrastructure and the agent deployment, and short listed two solutions that would aid us in the end-to-end process:

- **Ansible**: an open-source software enabling infrastructure as code, it's used as a provisioning, configuration management and application deployment tool. It was acquired by RedHat in 2015 [29], [30].
- **Terraform**: an open-source software tool enabling infrastructure as code operations by HashiCorp. It helps users to define and provision infrastructure using a declarative configuration language [31], [32].

The SBS architecture would be represented as follow (Fig. 12), with the VM for the agent being hosted either on Azure or AWS depending on the destination cloud.

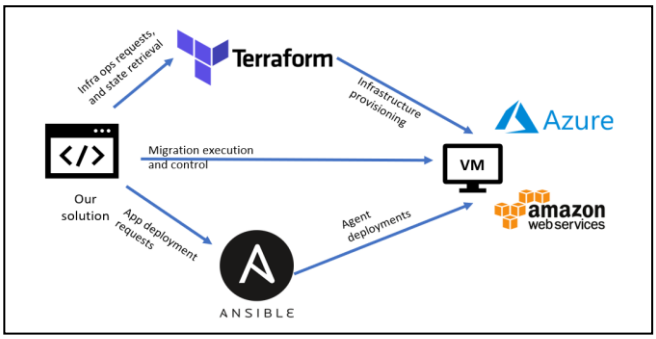

Figure 12. Diagram representing the solution architecture after adding Terraform and Ansible

Once a user selected the storage source and destination, a request is sent to (1) Terraform to deploy a Linux VM on either Azure or AWS on the destination DC, then (2) at the end of the deployment Terraform will receive the VM's metadata (e.g. Name, type, IP address etc.), (3) the IP address is then retrieved by the solution and passed through to Ansible instance which will launch the agent deployment on the VM, (4) once the solution is notified of the deployment's end, it sends a request to start the migration job by the agent.

5.7.1 Migration from Azure Storage to AWS S3

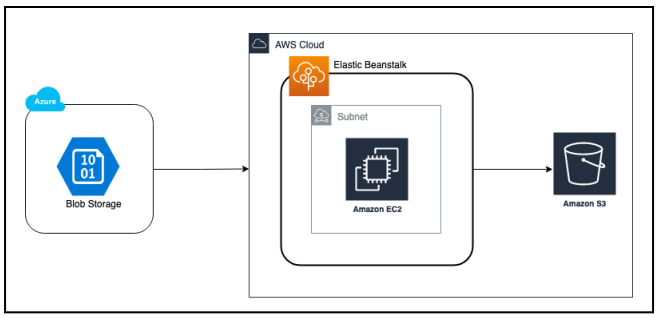

Figure 13. A pictorial representation of a Azure Blob Storage to AWS S3 migration architecture [32]

In Fig. 13, AWS described a way of moving the data from Azure Blob Storage to S3, using Elastic Beanstalk worker environment [33] as broker or migration agent. This version works but requires a lot of manual input from the end user. Following the same logic, we mounted the S3 storage onto the VM (EC2) as a remote storage to directly copy the data onto it without going through a temporary storage, thus considerably reducing the TTD. We used also the implementation in [34], which matched partially our needs and by installing the related Node Package Manager (NPM) one can make use the function *toS3* as seen on Fig. 14.

```
var toS3 = require('azure-blob-to-s3')

toS3({
  azure: {
    connection: '',
    container: 'my-container'
  },
  aws: {
    region: 'us-west-2',
    bucket: 'my-bucket'
  }
})
```

Figure 14. Code for function to move data from container to bucket on AWS

5.7.2 Migration from AWS S3 to Azure Storage

Microsoft has done a lot of improvement on its Azure Storage API too, bringing a capability to directly copy data onto Azure Storage from a remote URL, called the "Put Block From URL" operation, that can copy data from AWS S3 and which can reach a data transfer of 50 Gbps and more [35], using the following PowerShell command, Fig. 15.

```
azcopy cp "https://s3.amazonaws.com/mybucket/"
    "https://mystorageaccount.blob.core.windows.net/mycontainer<SAS>"
    --recursive
```

Figure 15. AzCopy script code to move data from AWS S3 to Azure Storage.

This gives us 2 important advantages, since there is no need for a VM to host the agent, as it's a script to run, it can easily be done through Azure Functions. (1) huge cost reduction as we won't need the additional compute for this action, and (2) TTD will considerably be reduced as it's a direct copy.

For AWS to Azure transfers we made use of AzCopy a command-line tool developed by Microsoft that is used to copy data to or from a storage account to another using PowerShell scripts.

Instead of using an Agent VM for this process, we optimized it and instead used an Azure Function which is a serverless Azure compute service that will help us run the migration script without provisioning or managing the VMs, it's also cost effective as it will run just enough to execute the script.

The architecture at this point runs smoothly, but it lacks a security component more related to the infrastructure description maintenance, thus we decided to operationalize the end-to-end deployment and solution maintenance process by adding DevOps components to maintain the solution and deployment configuration up to date.

For any change happening to either the VM configuration or Agent, the new version is committed to Git which then triggers the pipeline on Azure DevOps and releases the configurations files to either Terraform or Ansible, Fig. 16.

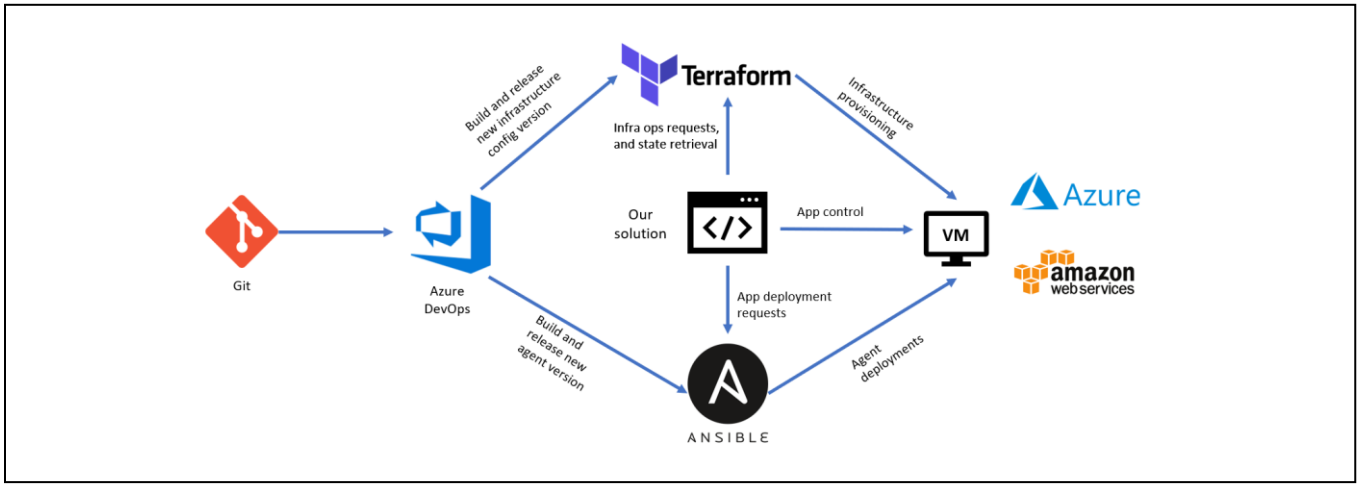

Figure 16. Diagram representing the solution architecture

5.8 Results :

Using our SBS approach, our testbed based on Azure and AWS as CPs, consisted of experimenting on transferring multiple files (medias/mp4) for a total of 29Gb with 2 main scenarios, the first one was from Azure Storage to AWS S3, and second one was from AWS S3 to Azure Storage. We selected two regions, Europe, and US for the DCs to experiment with a notable distance and compared the results with a manual migration which consists of (1) downloading the file to a remote temporary storage through a machine (2) Erase data from the source (3) write the data to destination (4) clean temporary storage.

Experiment 1: Transfer from Azure Storage in North Europe to AWS S3 (US East 2), the Agent VM was created on the Us-East-2 region on a VM of type *a1.xlarge* with an average download speed of ~274Mb/s, Fig. 17.

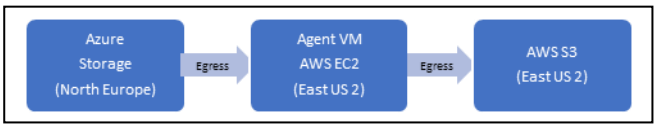

Figure 17. Diagram representing the Experiment 1 flow.

We assume the manual migration way, would have a same machine but created manually on a different location (North Europe for e.g.), which will impact the latency for data writing requests on to AWS S3, but also incur two egress charge.

Figure 18. Diagram representing the Experiment 2 flow.

Experiment 2: Transfer from AWS S3 in East US to Azure Storage in North Europe, the Agent Script was created on the North Europe region, executed through an Azure Function (serverless) with an avg speed ~25365Mb/s, Fig. 18. As for the manual migration approach, a manual creation of the machine was done on a different location (West Europe).

Table 4. Results from Experiment 1: Transfer from Azure Storage in North Europe to AWS S3 (US East 2)

	Manual migration	SBS
TTD	5mn23s	1mn11s
Cost incurred	~$4.37	~$2.66

Table 5. Results from Experiment 2: Transfer from AWS S3 in US East 2 to Azure Storage in North Europe

	Manual migration	SBS
TTD	20s	3s
Cost incurred	~$2.58	~$2.53

On the first experiment, there is a huge time and cost difference, we notice at least ~272% optimization and ~39% cost reduction which is due to the egress being charged twice in the manual way, incurred on transferring from Azure North Europe to AWS North Europe then AWS in North Europe to AWS East US 2.

On the second experiment, the cost is almost the same for both approaches. This can be explained by the fact that Azure Storage offers a capability to write directly on the Storage Account, which means that the data doesn't go through the VM but travels from S3 directly to Azure Storage. The slight difference in the transfer time is impacted by the latency for the write/read requests done by the Agent VM. It should be noted that the closer the VM is to the Storage account the better is the writing requests latency. Additionally, and since it's a direct transfer we are benefiting from the AWS egress bandwidth and Azure Storage ingress bandwidth (which can reach and exceed 50 Gb/s).

6. Conlusion and future work

By using our approach, we are able to cost effectively move data from one cloud to another while considerably minimizing the time required for migration but also optimizing the cost related to the move for customers where seconds of processing would cost millions of dollars.

Our approach is modular, the number of local ontologies is not fix, as long as we have a minimum of two, we can extend it to more local ontologies & knowledge bases, which would open new possibilities, for the purpose of this article the experiment used both AWS and Azure and it can also implement others CPs like Google Cloud platform by setting up an agent targeted on its API enabling a public cloud platforms interoperability, moreover, the private and hybrid cloud platforms would also be a candidate for this system, as OpenStack, and Microsoft Azure Stack both have APIs enabled allowing communication with their services, this would enable not only enable our approach to the public cloud interoperability but extend it to a multi-cloud interoperability.

While we proposed a technical based solution, we believe, an ideal one would be protocol based by setting standards for basic known services.

References

[1] A. F. B. Alam, A. Soltanian, S. Yangui, M. A. Salahuddin, R. Glitho, et H. Elbiaze, « A Cloud Platform-as-a-Service for multimedia conferencing service provisioning », in IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, p. 289-294, 2016.

[2] S. Alqahtany, N. Clarke, S. Furnell, et C. Reich, « A forensic acquisition and analysis system for IaaS », Clust. Comput., vol. 19, no 1, p. 439-453, 2016.

[3] W. Stefan, T. Eddy, et J. Wouter, « Comparing PaaS offerings in light of SaaS development | SpringerLink », Computing, vol. 96, no 8, p. 669–724, 2014.

[4] F. Liu, W. Guo, Z. Q. Zhao and W. Chou, "SaaS Integration for Software Cloud," IEEE 3rd International Conference on Cloud Computing, Miami, FL, pp. 402-409, 2010.

[5] A. Jivanyan, R. Yeghiazaryan, A. Darbinyan, et A. Manukyan, « Secure Collaboration in Public Cloud Storages », Yerevan, Armenia, vol. 9334, p. 190-197, 2015.

[6] C. Chilipirea, G. Laurentiu, M. Popescu, S. Radoveneanu, V. Cernov, et C. Dobre, « A Comparison of Private Cloud Systems », in 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Crans Montana, Switzerland, p. 139-143, 2016.

[7] X. Xu, « From Cloud Computing to Cloud Manufacturing », Robot Comput-Integr Manuf, vol. 28, no 1, p. 75–86, 2012.

[8] Y.-H. Chang et J.-Y. Chen, « A Hybrid Cloud for Effective Retrieval from Public Cloud Services », in Intelligent Information and Database Systems, Kuala Lumpur, Malaysia, p. 61-69, 2013.

[9] Gartner, « Gartner Says a Massive Shift to Hybrid Infrastructure Services Is Underway ». [online]. available on: https://www.gartner.com/newsroom/id/3666917, Retrieved on Nov-2020.

[10] R. Hill, L. Hirsch, P. Lake, et S. Moshiri, « Cloud Economics », in Guide to Cloud Computing, Springer, London, p. 187-207, 2013.

[11] KPMG International Cooperative, « Cloud Economics : Making the Business Case for Cloud », NDPPS 286069, 2015. [online]. available on:

https://assets.kpmg.com/content/dam/kpmg/pdf/2015/11/cloud-economics.pdf, Retrieved on Nov 2020

[12] G. A. Lewis, « Role of Standards in Cloud-Computing Interoperability », in 46th Hawaii International Conference on System Sciences, Wailea, HI, USA, p. 1652-1661, 2013.

[13] « Migrating AWS instances to Google Cloud ». https://cloud.google.com/migrate/compute-engine/docs/4.9/how-to/migrate-aws-to-gcp/migrating-aws-vms, Retrieved on Nov 2020

[14] C.-T. Yang, W.-C. Shih, C.-L. Huang, F.-C. Jiang, et W. C.-C. Chu, « On construction of a distributed data storage system in cloud », Computing, vol. 98, no 1, p. 93-118, 2016.

[15] J. P. Martin-Flatin, « Challenges in Cloud Management », IEEE Cloud Comput., vol. 1, no 1, p. 66-70, 2014.

[16] K. Benzidane, S. Khoudali, L. Fetjah, S. Jai Andaloussi, A. Sekkaki, « Application-based authentication on an inter-VM traffic in a cloud environment», International Journal of Communication Networks and Information Security. vol. 11, no 1, p. 148-166, 2019.

[17] B. Calder et al., « Windows Azure Storage: a highly available cloud storage service with strong consistency », in Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles - SOSP '11, Cascais, Portugal, p. 143-157, 2011.

[18] A. Singh et K. Chatterjee, « A secure multi-tier authentication scheme in cloud computing environment », in International Conference on Circuits, Power and Computing Technologies [ICCPCT-2015*]*, Cebu, Philippines, p. 1-7, 2015.

[19] D. Elliott, C. Otero, M. Ridley, et X. Merino, « A Cloud-Agnostic Container Orchestrator for Improving Interoperability », in IEEE 11th International Conference on Cloud Computing (CLOUD), San Francisco, CA, USA, p. 958-961, 2018.

[20] S. Dowell, A. Barreto, J. B. Michael, et M.-T. Shing, « Cloud to cloud interoperability », in 6th International Conference on System of Systems Engineering, Albuquerque, NM, USA, p. 258-263, 2011.

[21] E. Nogueira, A. Moreira, D. Lucrédio, V. Garcia, et R. Fortes, « Issues on developing interoperable cloud applications: definitions, concepts, approaches, requirements, characteristics and evaluation models », J. Softw. Eng. Res. Dev., vol. 4, no 1, p. 7, 2016.

[22] Z. Hill et M. Humphrey, « CSAL: A Cloud Storage Abstraction Layer to Enable Portable Cloud Applications », in IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, IN, USA, p. 504-511, 2010.

[23] K. Kaur, S. Sharma, et K. S. Kahlon, « Towards a Model-Driven Framework for Data and Application Portability in PaaS Clouds », in First International Conference on Sustainable Technologies for Computational Intelligence, vol. 1045, A. K. Luhach, J. A. Kosa, R. C. Poonia, X.-Z. Gao, et D. Singh, Éd. Singapore: Springer Singapore, p. 91-105, 2020.

[24] M. Abid, B. Nsiri, et Y. Serhane, « An approach based on ontologies and multi- agent systems to hide heterogeneity: Application to port information system », vol. 9, p. 82-87, 2015.

[25] S. Singh et Y.-N. Cheah, « A Hybrid Approach for Ontology Mapping via Genetic Algorithm », in 2011 Third International Conference on Computational Intelligence, Modelling & Simulation, Langkawi, Malaysia, p. 80-83, 2011.

[26] R. Meersman et C. Debruyne, « Hybrid ontologies and social semantics », in 4th IEEE International Conference on Digital Ecosystems and Technologies, Dubai, United Arab Emirates, p. 92-97, 2010.

[27] « AWS Global Infrastructure ». https://aws.amazon.com/about-aws/global-infrastructure/, Retrieved on Nov 2020.

[28] « Data residency in Microsoft Azure ». https://azure.microsoft.com/en-us/global-infrastructure/data-residency/, Retrieved on Nov 2020.

[29] « Ansible for Amazon Web Services ». https://www.ansible.com/integrations/cloud/amazon-web-services, Retrieved on Nov 2020.

[30] « Ansible and Microsoft Azure ». https://www.ansible.com/integrations/cloud/microsoft-azure, Retrieved on Nov 2020.

[31] « Terraform: Beyond the Basics with AWS ». https://aws.amazon.com/blogs/apn/terraform-beyond-the-basics-with-aws/, Retrieved on Nov 2020.

[32] « Using Terraform with Azure ». https://docs.microsoft.com/en-us/azure/developer/terraform/overview, Retrieved on Nov 2020.

[33] « One way to migrate data from Azure Blob Storage to Amazon S3 ». https://aws.amazon.com/blogs/storage/one-way-to-migrate-data-from-azure-blob-storage-to-amazon-s3/, Retrieved on Nov 2020.

[34] « azure-blob-to-s3 (2020) [Source Code] ». https://github.com/bendrucker/azure-blob-to-s3, Retrieved on Nov 2020.

[35] « Move your data from AWS S3 to Azure Storage using AzCopy ». https://azure.microsoft.com/en-us/blog/move-your-data-from-aws-s3-to-azure-storage-using-azopy, Retrieved on Nov 2020.