Azure Backup Deep Dive

Adin Ermie

Charbel Nemnom

Contents

Part 1: Introduction

Let us start this whitepaper by very clearly indicating that we will not cover the hybrid Azure Backup related technologies, such as System Center Data Protection Manager (SC DPM), and the Microsoft Azure Backup Server (MABS).

This whitepaper was written to explore the Azure Backup service in a deeper way, diving into the finer details of how things work, and to help people understand where it differs from what we traditionally used to do in the backup world.

To provide a little bit of history, the Azure Backup service specifically the one based on ARM/Recovery Services Vaults (RSVs) became Generally Available (GA) on May 12, 2016. Since then, the Azure Backup team has been working hard to innovate and integrate new technologies, scenarios, and components into the service, as the entire Azure cloud environment landscape continues to evolve.

As an example, here are just a few of the most recent enhancements to the Azure Backup service:

- Support for backup of Azure File Shares
- Improved backup and restore performance
- Support for Large Disks
- A new Log Analytics Backup Monitoring solution
- Rich reporting with Power BI integration
- Integration of Azure Backup into the Virtual Machine creation experience
- Support for BitLocker Encryption Key (BEK) encrypted Azure virtual machines
- Windows System State Backup to Azure with Azure Backup
- SQL Server in an Azure Virtual Machine

You may think that backup is dull and boring, but there is nothing boring about the amazing rapid changes and improvements that keep coming to a service that, traditionally, is just “set it and forget it”.

What motivated us to write this whitepaper, is the fact that many of the customers we speak to, still struggle to understand how Azure Backup works, especially in comparison to what they may traditionally be used to do in their on-premises environments.

We frequently hear customers say that they “want a full backup every day/week/month”, or even “alternating full backups” (i.e. two separate backup streams, “in case the first one becomes corrupted”).

So, let’s jump into by starting with a breakdown of the components that make up the Azure Backup service.

Part 2: Component Breakdown

In this section, we will cover each of the components that make up the Azure Backup service.

Now, of course, Microsoft has their own official documentation. And it is not our intention to just re-hash what's already been written/provided. we may refer to specific sections of the documentation at times, but we want to go deeper.

Azure Backup (MARS) Agent

This, very clearly, is the Agent where all the data funnel through. But, let's look at it a little closer.

Notice the acronym for the Agent is "MARS". If you are not familiar with other Microsoft Business Continuity and Disaster Recovery (BCDR) technologies, you may not realize that **MARS** actual stands for "Microsoft Azure Recovery Services". That may seem odd, especially if you're familiar with the Azure Site Recovery (ASR) service.

But ever since Microsoft combined the Azure Backup (ABU), and the Azure Site Recovery (ASR) services together under the "Azure Recovery Services" label, these two technologies work as 2 pieces in the same puzzle of "recovery".

Agent Cache Location

When you install the Microsoft Azure Recovery Services (MARS) Agent (which is required for backing up Files/Folders or System State only), aside from the installation folder, you will need to provide a 'Cache Location'. The current Microsoft documentation on installing and registering the Agent does not provide any additional details about this cache location. The only thing it does say is that "Your cache folder should be 5% of the space required for data storage."

So, let's dive a little more on this.

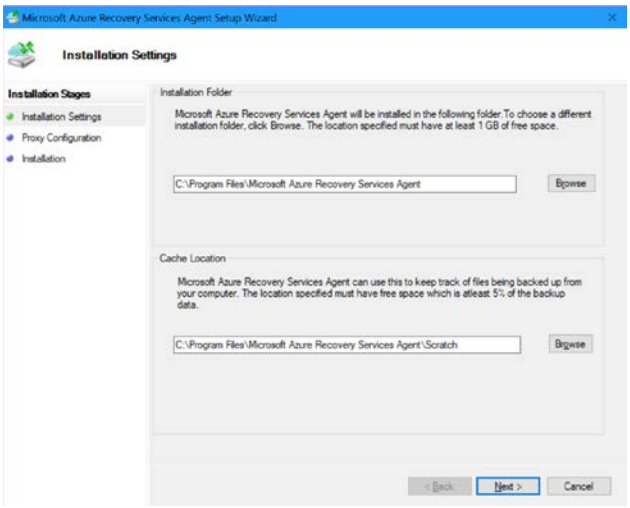

Figure 1 – Microsoft Azure Recovery Services Agent Cache Location

This “cache location” is also referred to as the “scratch space”. The Scratch Space is used during online backups (i.e. when you use Azure Backup as your long-term backup location in the case of System Center Data Protection Manager (SC DPM) and the Microsoft Azure Backup Server (MABS); or in general when you use the Azure Backup Direct with the MARS Agent).

> **Note:** In case of SCDPM and MABS (for local backups), there is another location for online recovery called **Recovery Folder Settings** which also requires enough free space for the recovery to be succeeded. For example, if you have a Virtual Machine that uses 100 GB of storage, then the Recovery Folder should also have 100 GB of free space (the ratio is 1:1 for temporary storage).

Before the data is sent to the Recovery Services Vault (covered in more detail in the next section), its metadata and checksum are collected in the Scratch Space so that the next incremental backups can happen seamlessly.

According to the documentation (and the installation wizard), this space is normally 5-10% of the size of the backup. So, this means you should perform some pre-deployment planning, and estimate the expected size of your backup data. If you’re backing up 1 TB of data, then your chosen Scratch Space should have approximately 51.2 – 102.4 GBs of free space available. And that doesn’t account for data growth!

Recovery Services Vault

When it comes to the Recovery Services Vault (RSV), think of it as the cloud version of SC DPM, and MABS.

If you’re familiar with System Center Data Protection Manager (SC DPM), or the Microsoft Azure Backup Server (MABS), you know there is a main server that performs the processing, coordination, etc. of the agents, backup/recovery jobs. Kind of like the air traffic controller between the source (i.e. the Agent), and the destination (i.e. the storage) locations. See *figure 2*.

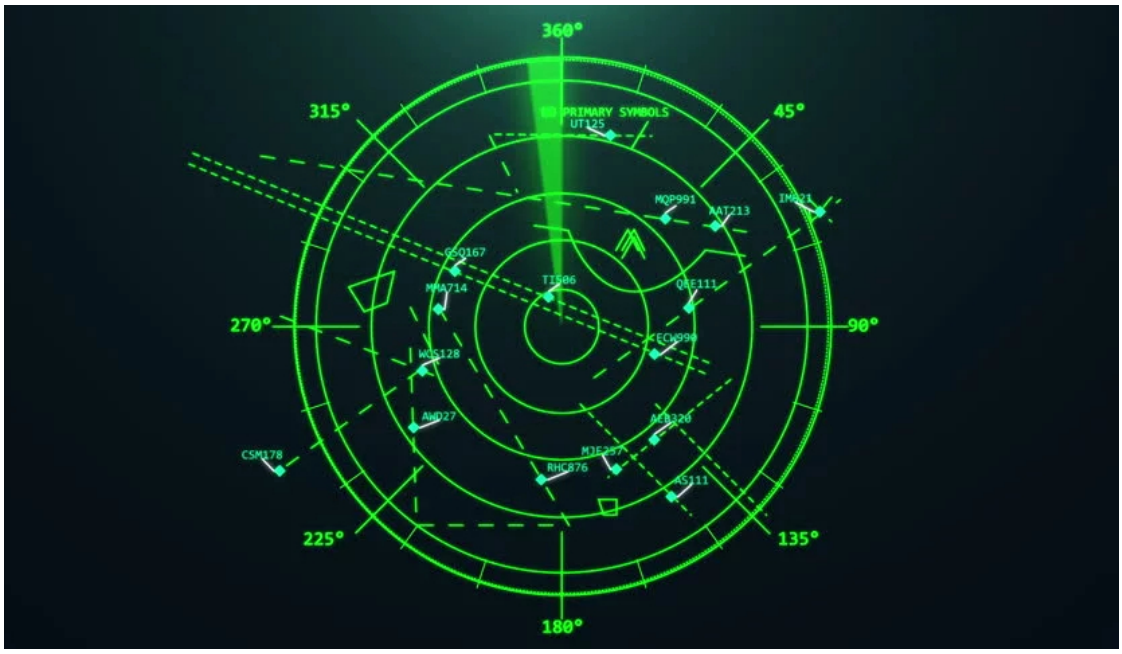

Figure 2 - Air traffic controller screen [Image credit: Carl Hendrick, https://bit.ly/2uAvCSg]

In the Recovery Services Vault (RSV), we have some collective data about all the data "airplanes" that are flying into the vault's "airport".

We have Flight 'AVM' (known as Azure Virtual Machine) coming in, and Flight 'ASAF' (known as Azure Storage – Azure Files) landing at the same time. And it's the air traffic controller's (Recovery Services Vault) responsibility to manage all that activity. The current list of available Backup Types is shown in *figure 3.*

BACKUP MANAGEMENT TYPE	BACKUP ITEM COUNT
Azure Virtual Machine	0
Azure Backup Agent	0
Azure Backup Server	0
DPM	0
Azure Storage (Azure Files)	0
SQL in Azure VM	0

Figure 3 - Azure Backup Management Types

Underlying Storage

If the Azure Backup (MARS) Agent is the "airplane", and the Recovery Services Vault (RSV) is the "air traffic controller", then the underlying Azure Storage would be the "tarmac" or "runway". See *figure 4*.

Figure 4 - Airport runway [Image credit: aertecsolutions.com, https://bit.ly/2L0tFJE]

But here's the thing. When we create a new Recovery Services Vault (RSV), we only get to choose how many "runways" there are, and not really. It's not like we can control how many data "airplanes" can use the airport. Instead, we control the Replication option, or, how many copies of our backup data exists.

Azure Backup (via the Recovery Services Vault) will create a *hidden* Azure Storage account. This Storage Account is where all our backup data will be maintained. But, Azure Storage provides different redundancy and replication options. This is what this Backup Configuration 'Storage replication type' setting refers to.

But remember, once you set this configuration, and then start your backups, you **cannot** change it! This will be a factor into the cost of the Azure Backup service; as we have to pay a set cost per protected instance, plus the cost of the amount of storage consumed. The cost per protected instance does not change with this configuration, but the cost for the underlying storage does (depending on if you use Local-Redundant Storage (LRS), or Geo-Redundant Storage (GRS)). See *figure 5*.

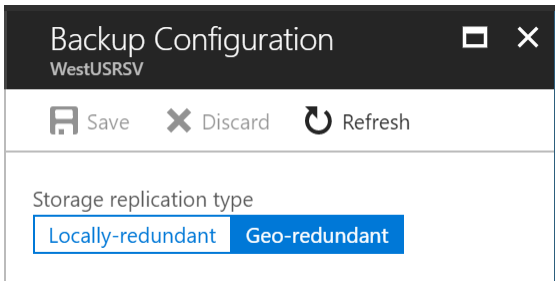

Figure 5 - Azure Backup Storage Configuration

If you're familiar with Azure Storage, and the various Storage Account Types, you may wonder which **Type** is used for the underlying storage component of Azure Backup.

The Storage Account Type that is used is General Purpose v1 Storage. This is because a General Storage account can be used for Blobs and Azure virtual machine disks. See *figure 6*.

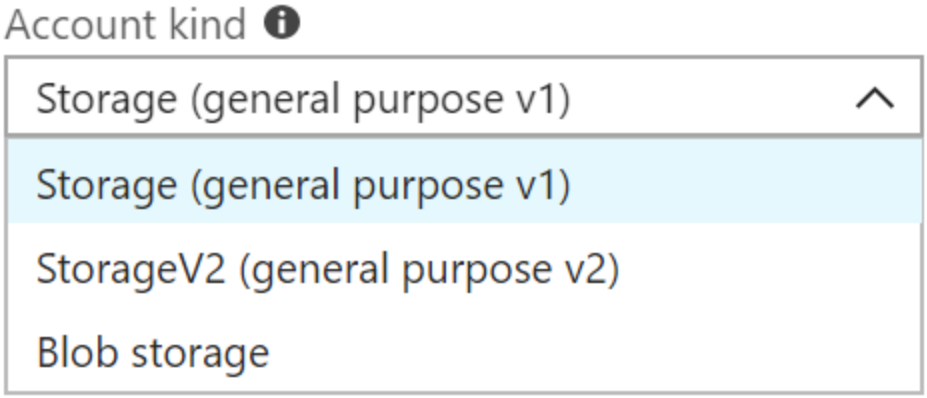

Figure 6 - Azure Storage Account Type

And, if you think about all the different types of data that Azure Backup can protect (i.e. Azure VMs, Files/Folders, Azure Files, SQL DBs, etc.) this then makes sense. Some of those are Binary Large Object (BLOB) items, and obviously the Azure Virtual Machine backups are of the disks.

Disk Failure

Knowing that the Recovery Services Vault utilizes a General-Purpose Storage account behind it, some people have questioned the stability of the Azure Backup service if something were to happen to the disk(s) that hold the backup data.

To put it simply, this is not an issue. There are many levels of redundancy that make up the Azure Storage subsystem.

Data Transfer, Compression, and Encryption

Data Transfer

With Azure Backup, the Agent uses change tracking to identify which *blocks* have changed, compresses this for low-bandwidth consumption, and then only transfers the changed content.

Although the data transfer in and out of Azure Backup is not limited (or charged for), if you have a large amount of data, it can be a challenge at first to get that initial content seeded. Azure has several alternative options to import large amounts of data into the platform.

The first one, is the Azure Import/Export service. This allows you to securely transfer large amounts of data into Azure (via physical HDDs), instead of transferring it over your network.

The second option is the newly announced Azure Data Box service (currently in Preview). Azure Backup has yet to announce its utilization/integration with this service, however, in our collective opinions, this is the next version of the “Import/Export” service referenced earlier. So, it is just a matter of time.

Compression

There is always questions around compression, and sub sequentially, deduplication.

When you are using the Azure Backup direct-agent, in connection with System Center Data Protection Manager (SC DPM) or the Microsoft Azure Backup Server (MABS), then the compression is offloaded from the source system, onto the SC DPM/MABS server. This removes the strain on the target systems resources (CPU, Memory, Storage), and places it on the backup server itself.

Azure Backup does not perform deduplication of the data that is transferred into the Recovery Services Vault (RSV). You may wonder why this is. Although this may cause the amount of storage used to maintain the backups to increase slightly, it allows for a faster recovery of that data. And, since Azure Backup only backs up incremental changes (after the initial full backup), the slight increase to the storage usage is minor.

If you are using either System Center Data Protection Manager (SC DPM), or the Microsoft Azure Backup Server (MABS) to support on-premises backups, those systems can/do support deduplication. However, this is done at the host-level. This means, if your DPM/MABS server is running in a Hyper-V Virtual Machine, then the VM host would perform the deduplication on the virtual hard disks (VHDXs) that are utilized by SC DPM/MABS as the local backup storage.

For more information on this, especially around critical details on File Formats, Volume Allocation Units, and optimizing backup and deduplication schedules; check out Charbel Nemnom's article: How To Reduce DPM 2016 Storage Consumption by Enabling DeDuplication on Modern Backup.

Encryption

When data is transferred into the Recovery Services Vault (RSV), it is encrypted using the Advanced Encryption Standard 256 AES method. The backup data is sent over a secure HTTPS link, and the data is also stored in the Recovery Services Vault in encrypted form. For a detailed explanation of this, please see the next section entitled Azure Backup Encryption Key.

During the installation of the MARS Agent, you are prompted to provide a passphrase key for data encryption, see *figure 7*. Although it is mentioned everywhere, we want to remind you that Microsoft **does NOT** save or manage this passphrase. It is **not** stored in the Recovery Services Vault (RSV), or anywhere else.

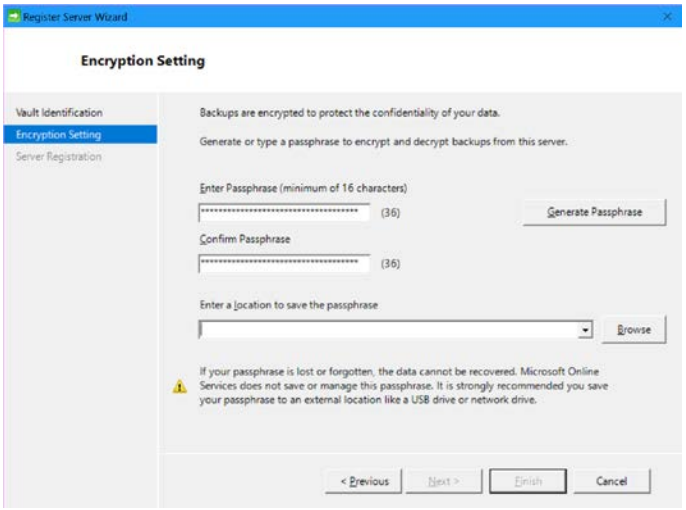

Figure 7 - Register Server Wizard

Everyone always asks, though, is this passphrase stored anywhere (i.e. the Registry, some Config file, etc.). The answer is a resounding **NO!** You must record this passphrase in a secure (and retrievable) location. **Do not** store the passphrase on the protected server. That will not help you if you are unable to get into the affected server. **You are the only person responsible for the passphrase key**. If the server is compromised, and you store your passphrase on the protected server, this will allow an attacker to manipulate your backups!

Now consider the following scenario:

You initially configured backups, along with a passphrase, but now you have lost or forgotten that passphrase.

In the Properties of Azure Backup, you can change the passphrase, see *figure 8*. However, you can only change the passphrase by confirming the action with a secure PIN, which is generated from the Azure portal, and only valid for 5 minutes. But, if you change the passphrase, what happens to previous backups that were encrypted with the original passphrase?

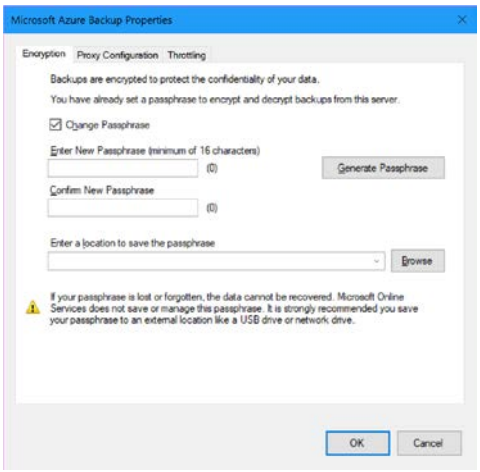

Figure 8 - Microsoft Azure Backup Properties

You can still restore files that were encrypted with a previous passphrase! Check out the next section on how the encryption works behind the scenes.

Azure Backup Encryption Key

In the encryption setting page, you can manually specify the passphrase (which requires a minimum of 16 characters) or you can auto generate a new one. There is an important point that you should be aware of. Microsoft does not encrypt your data with that passphrase, they do not use that key for encryption. Instead, they generate what is called a **Data Encryption Key (DEK)** based on the protected machine and its unique GUID key specific for that machine only. Then they use that DEK to encrypt the data. The data is compressed locally before sending it over to the Azure Recovery Services Vault (RSV).

The first question that you may ask is: "OK, so what is the purpose of the passphrase and how is the data secured then?" Right, after Microsoft encrypts the data using the auto generated DEK and copies the data to the Recovery Services Vault (RSV), they take that DEK and then they encrypt it again with the passphrase that you generate first. This process is known as **Key Envelop Key (KEK)**. The KEK is used to encrypt the DEK, and then the encrypted DEK is stored along with each recovery point.

The next question is: "So why is Microsoft performing this lengthy process?" The answer is, because some organizations have a policy in place to change the passphrase every six months, or the backup IT administrator is changing his role. In those kind of scenarios, you want to change the passphrase. But, if you want to change the passphrase, and Microsoft used that passphrase as an encryption mechanism, then Microsoft is forced to send all your backup data from the Recovery Services Vault (RSV) to the protected machine (whether it's on-premises or in the cloud), then decrypt the data using the *old* key and then *re-encrypt* it using the new key! That process is obviously not so efficient. So, instead what they do when you change the passphrase is the following:

1. They send only the encrypted **Data Encryption Keys (DEK)** back to on-premises.
2. Then they decrypt them using the old key and then re-encrypt them using the new key.
3. Finally, they send the Data Encryption Keys (DEK) (that are now encrypted with the new passphrase key) back to the Recovery Services Vault (RSV).

In this way, you gain two things:

First, you can change the passphrase anytime without the need to send large amounts of data back for decryption and re-encryption every time. Second, since Microsoft is storing the encrypted **Data Encryption Keys (DEK)** with each recovery point, if the protected machine crashes, you can then actually use the passphrase (that you kept it in a secure place, right?) to recover your data to a different machine.

Part 3: How Backups Work

In this section, we will explore how Azure Backup compares to the traditional backup technologies and methods that are used in our environments.

Traditional Methodologies

In a traditional approach, a third-party backup solution requires some sort of 'media' server that acts like the brains of the operation. This primary server also requires attached storage (in large quantities) to store the backup data.

Even when these types of solutions adapt to a cloud model, they treat the cloud as a storage endpoint or destination. Effectively, they view the cloud as just another disk or tape location.

The problem with this approach, is that there are gaps, limitations, and overhead involved. For example, one customer did their "research" on backup solutions and chose to use CommVault as their one-and-only backup solution. The problem with this, is that they were using it not only against on-premises systems, but also for 100's of Virtual Machines running in Azure.

While this solution can perform data backups from inside the Virtual Machine (via the Agent), it meant that the "only" way to perform a restore of a Virtual Machine (running in the cloud), was to create an entirely new Virtual Machine, and then perform a restore operation on the data! Very inefficient.

If we were only working within an on-premises environment, then we could take a full VM-level backup from the hypervisor layer (either Hyper-V or VMware). But because this is a cloud environment, the underlying hypervisor is not exposed; at least not to the third-party solutions.

The overhead on a solution like this, is having to manage and maintain the Storage Account that this 'media' server operates on; not to mention the patching, management, security, monitoring, etc. Now, of course, storage is virtually endless in a cloud environment, but there are still limitations imposed on the number of IOPS throughput you can achieve, the bandwidth into the storage, etc.

This is the challenge with "simply" deploying a third-party solution that we would traditionally use on-premises, into a cloud environment. We have a better approach by using a solution that is native to our environment.

Azure Backup Methodologies

As was mentioned at the end of the Traditional Methodologies section, we obtain a better approach and experience by using a solution that is native to our environment. But why is that?

Because when something is native, it hooks into the fabric/framework of the supporting environment. It is through this mechanism that Azure Backup provides full VM-level backups.

But you might be thinking "I don't need full VM backups, I only care about the files within the VM." And that is a valid point. But, think about this. Let's pretend we're not even in a cloud environment, let's suppose we are on-premises. If your File Server were to go down, which would you rather do:

1. Build a new File Server system (inclusive of disk sizing/assignment, domain join, permissions, monitoring, antivirus/security, etc.), and then perform the data restore (after registering the new system with your backup software and allowing the data restoration to occur against the non-original server).
2. Or, Restore the entire server through a single operation, as if nothing had happened.

Both options work and are valid. But one takes considerably more time to complete. It is very important to take into consideration your Recovery Time Objective (RTO) to ensure that you can meet your Service Level Agreement (SLA).

Another powerful point to using Azure Backup is that the Recovery Services Vault (RSV) is the "brains" of the operation, without the overhead of needing to patch, maintain, manage yet another Virtual Machine in the environment. Remember, third-party solutions require some sort of 'media' server that coordinates everything. And you need to maintain this system, including security, OS hardening, networking and communications not only to/from the Agents, but to the storage subsystem, etc.

Finally, one of the most powerful and useful features in Azure Backup, is the ability to mount a snapshot of a Virtual Machine, browse the contents (i.e. via File Explorer), locate the Files/Folders you want to restore, and then perform the restore operation on *just* those files. This is known as Item Level Restore (IRL). In other third-party backup solutions, you will have to restore the entire VHD first, before you are able to browse its contents, and retrieve the files you're looking for. With Azure Backup, you take a single backup of the full Virtual Machine, and you get both VM-level restores, and File/Folder restores from the same snapshot. You don't need to have a backup Agent inside the system, just to have file-level restore operations.

So, let's recap. With Azure Backup we have the following features:

- Full Virtual Machine level backups
- No management infrastructure to patch, maintain, etc.
- Endless storage, without needing to maintain and manage the Storage Account
- File/Folder level restore out of the VM-level backup, *without* having to restore the entire VHD file first to browse and find your target file

Full, Differential and Incremental (Expanded)

Now we're going to deep-dive into how Azure Backup works with the different backup types.

Again, in a traditional solution/environment, we may be performing full backups every day, week, or month.

Let's use a File Server as our reference example. We're going to assume that our File Server has 10 TB of content. With a **Full Backup**, the full 10 TBs of content is backed up, *every time*.

Look at this in another way. We can compare the File Server to your car. And every week you purchase another identical vehicle, because you want a "full" backup of your car in case you need to replace a single part. That's going to greatly increase the amount of space you need (and will consume) in your garage. Not to mention the extra cost.

With differential backups, we store the changes that have occurred since the last *full* backup. Assume that you're trying to sell your car on a website, and every week you have to upload pictures of the vehicle. Now, you decided to change your tires on Monday, so you take a new picture and upload it to the website. Then on Tuesday, you change your headlights; but, instead of only taking a picture of your new headlights, you also take another picture of your tires, because they changed since your first picture upload. Then on Wednesday, you change your radio, and again, take pictures of your tires, headlights, and radio. This continues until the end of the week, when you take a new "full backup" set of pictures. Obviously, taking pictures of something that you're already taken pictures of, is redundant (even though we do live in a digital age).

Now, with incremental backups we only keep the changes that have occurred since the last *backup* (not the last full backup). So, when we change our tires, we only take 1 picture. And when we change our headlights, we again, only take 1 picture, etc.

Of course, this example is a little excessive, but we're trying to compare/put into perspective the differences between full, differential, and incremental backups.

Here is a visualization to help clarify this (see *figure 9*):

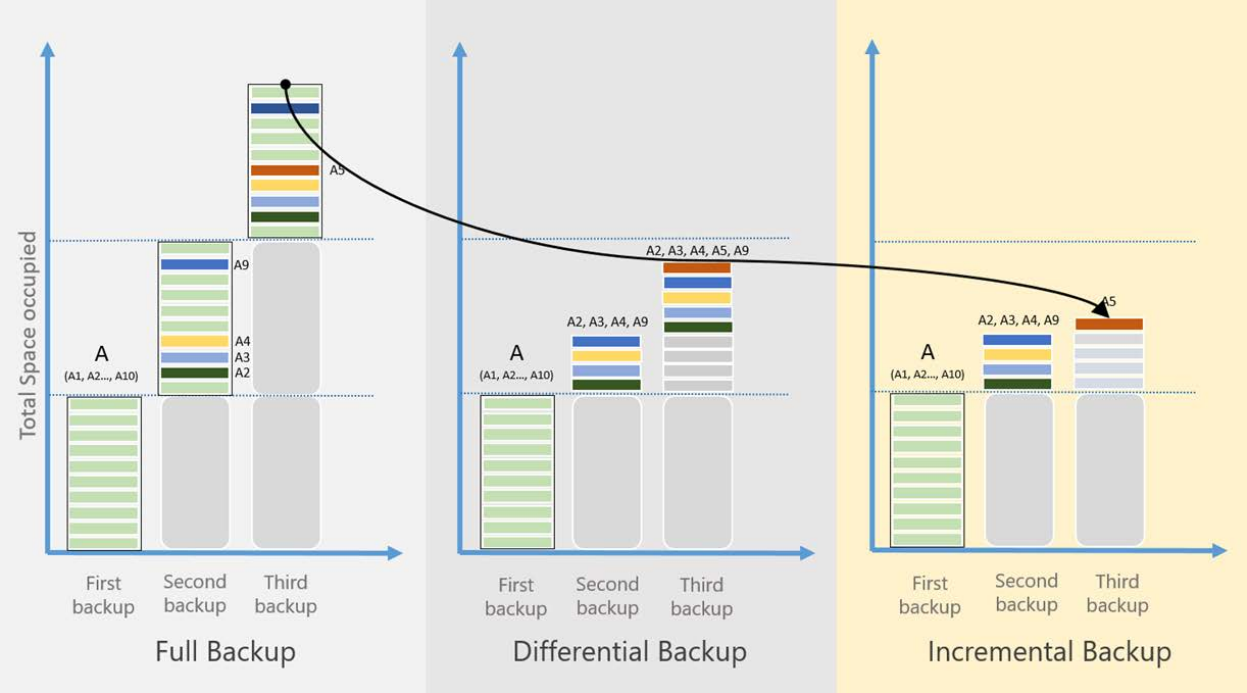

Figure 9 – Incremental backups on Microsoft Azure Backup

In this diagram, starting with the Full Backups, every backup contains the full set of data, every time, regardless if any of the data has changed.

With Differential Backups, the changed blocks A2, A3, A4 and A9 will be backed up in the second backup (because they changed from the original first), but they will also be backed up again in the third backup

(along with the A5 block), because the change *reference* point is the first original 'full' backup. Only after *another* full backup is created, will the changed blocks no longer be included in the backup jobs.

With Incremental Backups, only the blocks that have been changed since the previous backup will be included. This removes the need to take regular/multiple full backups; because the reference point is not the original full backup.

Restore Operations

For the restore operation, this is dependent on what you're trying to restore.

For example, if you are just trying to restore Files, then Azure Backup can mount your recovery point (via ISCSI) and then you can browse for the file(s) and restore them.

But what about something more complicated, like a Virtual Machine restore?

Note: In this context, we are referring to an Azure IaaS Virtual Machine, and not any on-premises (Hyper-V/VMware/Physical) type of scenario.