Using Windows Storage Spaces and iSCSI on Amazon EBS

October 2015

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The software included with this paper is licensed under the Apache License, Version 2.0 (the "License"). You may not use this file except in compliance with the License. A copy of the License is located at http://aws.amazon.com/apache2.0/ or in the "license" file accompanying this file. This code is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Contents

Abstract

This whitepaper is intended for Microsoft Windows IT professionals and developers who are interested in combining Windows Server 2012 R2 Storage Spaces with the Amazon Elastic Block Store (Amazon EBS) service. This paper describes potential use cases and security technologies for running Storage Spaces in AWS over the Server Message Block (SMB) and Internet Small Computer System Interface (iSCSI) protocols. To embrace the DevOps philosophy of "infrastructure as code," Windows PowerShell scripts are provided to quickly deploy Storage Spaces in AWS.

Introduction

Windows Server in Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides a secure global infrastructure to run Microsoft Windows Server workloads in the cloud, including Internet Information Services (IIS), SQL Server, Exchange Server, SharePoint Server, Lync Server, Dynamics CRM, System Center, and custom .NET applications. Pre-configured Amazon Machine Images (AMIs) enable you to start running fully supported Windows Server virtual machine instances in minutes. You can choose from a number of server operating system versions and decide whether or not to include pre-installed SQL Server in the hourly cost.

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) provides persistent block-level storage volumes for use with Amazon EC2 instances in the AWS cloud. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes provide consistent low-latency performance. On Windows Server instances in AWS, Amazon EBS volumes are mounted to appear as regular drive letters to the operating system and applications. Amazon EBS volumes can be up to 16 TiB in size, and you can mount up to 20 volumes on a single Windows instance.

Here are some of the key benefits of Amazon EBS:

- Block storage—use it as you would use a hard disk.
- Designed for 99.999% *availability*.
- Ability to back up the data via snapshots to Amazon Simple Storage Service (Amazon S3) for 99.999999999% *durability*.
- Ability to provision storage independent of instance.
- Data persists independent of instance.
- Three volume types to choose from (magnetic and SSD).
- Ability to encrypt data in flight and at rest.
- Ability for customers to supply their own encryption keys.

Windows Server Storage Spaces

Storage Spaces in Windows Server 2012 R2 allows administrators to pool multiple storage volumes into large virtual drives. Each pool may include disk drives of different media types, such as low-cost magnetic drives and high-performance solid-state drives (SSD). Storage Spaces can automatically optimize the use of the different media types with a feature called *storage tiering*. Storage tiering keeps frequently accessed working set data on the SSD (where it can be accessed the fastest), and moves less frequently used data to magnetic drives.

Storage Spaces also provides a data deduplication feature to reduce storage consumption, a write-back cache to improve performance, and an auto-repair feature to rebuild data on failed disk sectors.

In addition to supporting NTFS, which is popular and proven technology, Storage Spaces offers a new file system called *ReFS*. ReFS stands for Resilient File System, and gets its name because it provides additional failure monitoring and correction features.

It's clear that Microsoft has invested a great deal to develop powerful and economical Storage Area Network (SAN) features in Windows Server 2012 R2, but to use Storage Spaces in an on-premises environment, administrators would need to purchase and manage dozens of JBOD disks in external enclosures. That

costs time and money, but commercial SAN alternatives are also very expensive. Also, these storage enclosures aren't really dynamically scalable. Drives can be added and removed, but it's tedious.

The AWS cloud offers a solution to this dilemma. By integrating Amazon EBS with Windows Storage Spaces, administrators can deliver powerful and low-cost, software-based SAN capabilities in the cloud, with very high agility, durability, and scalability.

Storage pools may be clustered for high availability, but in Windows Server 2012 R2, these must be based on Serial Attached SCSI (SAS) connected physical disks. This means that clustered Storage Spaces on top of Amazon EBS is currently not a scenario supported by Microsoft. Amazon EBS offers higher durability than traditional disk drives, but each EBS volume can be attached to only a single Windows Server instance at a time. This, unfortunately, is a single point of failure and does not ensure high availability. This paper will propose several use cases and security options for Storage Spaces with Amazon EBS. It is up to you to evaluate if the lack of high availability is a critical issue in your scenario.

> **Note** Microsoft has announced that Windows Server 2016 will include a new feature called Storage Spaces Direct[1] that will enable storage devices from multiple servers to be pooled.

Other Storage Services in AWS

This paper focuses on Amazon EBS; however, a brief overview of other AWS storage services is provided here for comparison.

Amazon S3

Amazon Simple Storage Service (Amazon S3) provides developers and IT teams with secure, durable, highly scalable, cost-effective object storage. Amazon S3 is easy to use and includes a simple web services interface to store and retrieve any amount of data from anywhere on the web. Object storage is not appropriate for workloads that require data insertions, such as databases. However, Amazon S3 is an excellent service for storing snapshots of Amazon EBS volumes. While

Amazon EBS duplicates your volume synchronously in the same Availability Zone, snapshots to Amazon S3 are replicated across multiple zones, substantially increasing the durability of your data.

Amazon Glacier

Amazon Glacier is a secure, durable, and extremely low-cost storage service for data archiving and online backup. To keep costs low, Amazon Glacier is optimized for infrequently accessed data where retrieval times of several hours are suitable. Amazon Glacier works together with Amazon S3 lifecycle rules to help you automate archiving of Amazon S3 data and reduce your overall storage costs. For example, you can easily set up a rule that stores all your previous Amazon S3 object versions in the lower-cost Amazon Glacier storage class and deletes them from Amazon Glacier storage after 100 days.

Amazon EFS

Amazon Elastic File System (Amazon EFS) is a file storage service for Amazon EC2 instances that allows you to create and configure file systems quickly and easily. Storage capacity in Amazon EFS is elastic—it grows and shrinks automatically as you add and remove files. Multiple Amazon EC2 instances can access an Amazon EFS file system at the same time. Amazon EFS supports the Network File System version 4 (NFSv4) protocol. NFS is most popular with Linux and Unix servers, while the Common Internet File System (CIFS) is most popular with Windows servers and clients. CIFS is also called Server Message Block (SMB). Amazon EFS does not currently support CIFS.

Amazon EBS Pricing and Performance

This table summarizes the features and limitations of the three volume types in Amazon EBS:

	General Purpose (SSD)	Provisioned IOPS (SSD)	Magnetic
Use cases	Boot volumes, small to medium DBs, and dev/test environments	I/O-intensive workloads and large DBs	Infrequent access
Maximum size	16 TiB	16 TiB	1 TiB

	General Purpose (SSD)	Provisioned IOPS (SSD)	Magnetic
Maximum IOPS/volume	10,000	20,000	40-200
Maximum burst IOPS	3,000	N/A	hundreds
Maximum throughput/volume	160 Mbps	320 Mbps	40–90 Mbps
Maximum IOPS/instance	48,000	48,000	48,000
Maximum throughput/instance	800 Mbps	800 Mbps	800 Mbps

Understanding Latency, IOPS, and Throughput

Although the storage industry often refers to IOPS as the key metric of performance for storage systems, it turns out that nowadays, latency is actually the most important factor for customers.

One problem with IOPS is that it has no standard definition. A disk drive spinning at 7,200 RPM can perform at 75 to 100 IOPS, whereas a drive spinning at 15,000 RPM will deliver 175 to 210 IOPS. The exact IOPS number will depend on a number of factors, including the access pattern (random or sequential) and the amount of data transferred per read or write operation. For example, one vendor might claim 2,000 IOPS after testing 64-KB blocks, with random access, read and write, no cache; while another vendor might claim 1,000,000 IOPS based on 16-KB blocks, with sequential access, read-only. Clearly the second case will sound “faster” than the first, but those two IOPS numbers are apples and oranges.

Amazon EBS measures each I/O operation per second (that is, 256 KB or smaller) as one IOPS, for read or write. I/O operations that are larger than 256 KB are counted in 256-KB capacity units. For example, a 1,024-KB I/O operation would count as 4 IOPS. When you provision a 4,000-IOPS volume and attach it to an EBS-optimized instance that can provide the necessary bandwidth, you can transfer up to 4,000 chunks of data per second, provided that the I/O does not exceed the per volume throughput limit of General Purpose (SSD) and Provisioned IOPS (SSD) volumes.

This same configuration could transfer 4,000 32-KB chunks, or 2,000 64-KB chunks, or 1,000 128-KB chunks of data per second as well, before hitting the volume throughput limit. If your I/O chunks are very large, you may experience a smaller number of IOPS than you provisioned because you are hitting the volume throughput limit. For more information, see Amazon EBS Volume Types.[2]

For 32 KB or smaller I/O operations, you should achieve the amount of IOPS that you have provisioned, provided that you are driving enough I/O to keep the drives busy. For smaller I/O operations, you may even see an IOPS value that is higher than what you have provisioned (when measured on the client side)—this is because the client may be coalescing multiple smaller I/O operations into a smaller number of large chunks.

If you are not experiencing the expected IOPS or throughput you have provisioned, ensure that your Amazon EC2 bandwidth is not the limiting factor. Your instance should be EBS-optimized (or include 10 Gigabit network connectivity) and your instance type EBS-dedicated bandwidth should exceed the I/O throughput you intend to drive. For more information, see Amazon EC2 Instance Configuration.[3] Another possible cause for not experiencing the expected IOPS is that you are not driving enough I/O to the EBS volumes. For more information, see Workload Demand.[4]

The reason latency is typically more important than IOPS is that it determines how long users must wait for data retrieval to begin.

There are two important steps you can take to improve latency in Amazon EBS:

1. Switch Magnetic volumes to General Purpose (SSD) volumes.
2. Switch your instances that are running on older hardware to newer hardware. Specifically, the M3 instance family uses older technology than the C4 and R3 instance families, so you will notice immediate improvements in latency if you switch an m3.large instance to a c4.large or an r3.large instance.

When attached to EBS-optimized instances, Provisioned IOPS (SSD) volumes can achieve single-digit millisecond latencies.

General Purpose (SSD)—IOPS and Throughput Limits

The actual throughput that you will see in practice can vary based on instance type, file system type, and your application's unique usage pattern.

For example, a 100-GiB General Purpose (SSD) volume has a baseline of 300 IOPS and the ability to burst to 3,000 IOPS.

Volumes larger than 1,000 GiB will always be able to achieve their baseline IOPS. For example, a 2,000-GiB volume will have a baseline of 6,000 IOPS. Volumes from 3,334 GiB up to 16,384 GiB will all get a baseline of 10,000 IOPS, which is the maximum for General Purpose (SSD) volumes.

The maximum throughput of your EBS volume can limit your IOPS if each I/O operation you are performing (read or write) is relatively large. For example, for General Purpose (SSD), you can either achieve a maximum of 10,000 IOPS when driving smaller I/O, or you can achieve up to 160 Mbps when driving larger I/O. This is consistent with the AWS definition of IOPS.

Amazon EBS Pricing

This section describes how pricing works for each EBS volume type. There is slight variation in pricing among AWS regions, but as of the date of publication, pricing for Amazon EBS in the US East (N. Virginia) region is as follows:

EBS Volume Type	Pricing
Magnetic	$.05/GiB/month + $.05/million I/O requests
General Purpose (SSD)	$.10/GiB/month
Provisioned IOPS (SSD)	$.125/GiB/month + $.065/PIOPS/month

Note AWS pricing is updated frequently. The pricing in the preceding table may not reflect current prices. For the latest information, go to the Amazon EBS Pricing webpage.[5]

Magnetic Volumes

Magnetic volumes are charged by the GiB you provision per month, until you release the storage, plus a charge per 1 million I/O requests you make to your volume.

General Purpose (SSD) Volumes

General Purpose (SSD) volumes are charged by the GiB you provision per month, until you release the storage. All your I/O is included in the price.

Provisioned IOPS (SSD) Volumes

Provisioned IOPS (SSD) volumes are charged by the GiB you provision per month, until you release the storage, plus a charge for the IOPS you provision multiplied by the percentage of days per month you provision the volume (even when the volume is detached from an instance). For example, if you provision a volume with 1,000 IOPS and keep this volume for 15 days in a 30-day month, then in a region that charges $0.10 per provisioned IOPS-month, you would be charged $50 for the IOPS that you provision ($0.10 per provisioned IOPS-month * 1000 IOPS provisioned * 15 days/30).

Windows Storage Spaces in Amazon EC2

Remote Access Connectivity and Security

You have a choice between two storage communication protocols in your Storage Spaces architecture: SMB 3.0 and iSCSI.

SMB 3.0 is primarily used by Windows servers and clients. When sharing a folder named **C:\myfolder** from a server named **myserver**, clients can access it over SMB by using the Universal Naming Convention (UNC) path **\\myserver\c\myfolder**.

iSCSI takes a little more work to set up than SMB, but it is interoperable with other storage products and with Windows, Mac, and Linux operating systems across the Internet. The server that hosts the iSCSI storage volume is called the *target*, while the client is called the *initiator*. When a client initiates a connection to a target, the volume appears as a regular drive letter in Windows. On the

server, the iSCSI target is a virtual disk that appears as a .vhdx file inside the Storage Spaces virtual disk. That's right, a virtual disk inside a virtual disk.

In addition to considering storage communication protocols, you must also consider the network access protocols that will connect client computers to the server that is running Storage Spaces. There are six options here:

- Amazon Virtual Private Cloud (Amazon VPC)
- VPC peering
- Virtual Private Network (VPN)
- AWS Direct Connect
- Microsoft DirectAccess
- Internet with iSCSI and CHAP

In a traditional SAN, Fibre Channel hardware switches are used to connect servers to storage cabinets. iSCSI is a software alternative to Fibre Channel that doesn't require purchasing expensive hardware switches. With support for iSCSI included in Windows Server 2012 R2, it's possible to connect to Storage Spaces over the Internet from Windows, Mac, and Linux machines.

> **Note** In this paper, references to Windows as a Storage Spaces "client" denote the following versions of Windows that support Storage Spaces technology:
>
> Windows 7, 8, 8.1, 10, Windows Server 2012 R2, Windows Server 2016

Security Technologies

The three remote access options listed in the previous section (VPN, AWS Direct Connect, and Microsoft DirectAccess) provide built-in authentication and encryption. If you plan to use a raw Internet connection without remote access technology, we recommend that you use Challenge-Handshake Authentication Protocol (CHAP) for authentication, and some form of in-flight encryption. For

in-flight encryption, you encrypt your data locally before transmitting it over the Internet, and you implement an inverse decryption step in your process when the data is accessed from the server.

In all cases, you should also use AWS security groups as a firewall, and consider using network access control lists (ACLs) in AWS as well.

AWS Security Groups

With all of the above network technologies, you can restrict access to your Storage Spaces volumes by using AWS security groups. Security groups function similarly to IPsec firewall rules and are very simple to set up. Security groups can be created automatically with AWS CloudFormation or Windows PowerShell, or manually in the Amazon EC2 console. First, to use Remote Desktop Connection, you'll need to open TCP port 3389.

To use iSCSI, you must open TCP ports 860 and 3260.

To use SMB, you must at least open TCP port 445. You can optionally open UDP ports 137 and 138, and TCP ports 137 and 139 if you want to use NetBIOS name resolution rather than IP addresses.

AWS Network ACLs

In addition to using AWS security groups to restrict traffic to the Windows Storage Spaces instance, you may also use network access control lists (ACLs). Keep in mind a couple of differences between security groups and network ACLs:

- Security groups apply at the instance level, while network ACLs apply to all instances within a subnet. Some administrators therefore like to apply network ACLs for defense-in-depth, in case an instance is ever launched without the correct security groups.
- Security groups are stateful, while network ACLs are not. This means that you only need to configure inbound rules for security groups, and they automatically permit outbound packets to be sent in response to allowed inbound requests; but with network ACLs you must explicitly configure inbound and outbound rules.

CHAP

If you are considering using iSCSI over unsecure networks, you should also plan to use Challenge-Handshake Authentication Protocol (CHAP). CHAP is an authentication protocol that is designed to prevent replay attacks. It is not an encryption protocol. Furthermore, researchers have discovered weaknesses in CHAP, so if you go this route, it's advisable to also encrypt your data before transmitting it over the Internet.

The PowerShell scripts provided later in this paper will show you how to set up CHAP with iSCSI.

Use Cases

When considering how to combine these storage network protocols and remote access protocols, certain use cases emerge. These are discussed next.

The following diagram depicts most of the networking options described previously. Each of the grey, dashed lines in the diagram represents an optional path that you could configure for access to Storage Spaces. The diagram doesn't show VPC peering, because when VPC peering is configured, connecting to Storage Spaces from a peered VPC would be just like the path from the second Availability Zone.

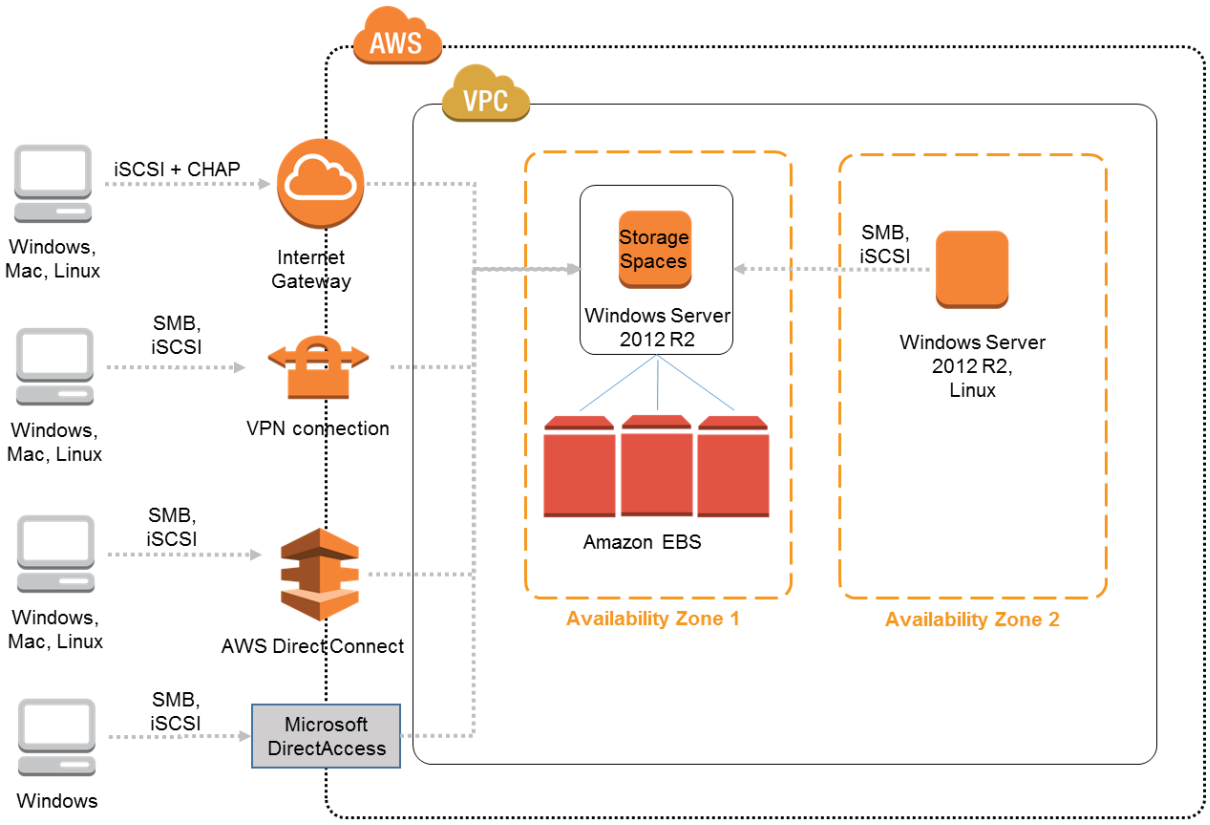

Figure 1: Networking Options for Storage Spaces

Given this diagram showing storage networking and remote access protocols, here are some potential use cases for Storage Spaces in AWS:

- You could pool multiple EBS volumes into a very large volume for application servers running on Windows Server, including SQL Server, Exchange Server, Lync Server, SharePoint Server, Dynamics CRM, or Oracle. The server application could be running on the same instance or on a separate instance. In fact, the application server could technically reside in a separate Availability Zone or in a peered VPC. In any case, since the application server is running Windows Server as a "client" to the storage pool, you would share the volume over SMB as a UNC path (you would not need to use iSCSI).
- You could set up on-premises servers to back up files to the large AWS volume. If the on-premises servers are Linux or Unix, you would connect via iSCSI and CHAP. If the servers are Windows, and if you are using one of

the remote access technologies listed earlier (VPN, AWS Direct Connect, or Microsoft DirectAccess), then you could connect via SMB.

- You could allow mobile and on-premises employees to back up files to a personal virtual disk created in a storage space in AWS. This can be enabled in Windows 7, 8, or 10 (where it would appear as a local drive letter or a UNC share name), as well as in Mac and Linux clients.

Benefits of Storage Spaces in AWS

These are some of the benefits of using Storage Spaces in AWS:

- A key advantage in each of the above use cases is that administrators only need to take backups from a single point in the cloud.
- Mirroring can be easily enabled in the storage pool to provide even higher durability than with individual Amazon EBS volumes, but with some loss of storage capacity at the same cost.
- The pool can take advantage of economic storage tiering across the different media types offered by Amazon EBS, including SSD and magnetic drives. This allows you to maximize the ratio of IOPS per dollar.

Limitations of Storage Spaces in AWS

Here are some current limitations to be aware of when using Storage Spaces in AWS:

- Certain performance features in SMB 3.0 are not available in AWS. This includes Multipath I/O (MPIO) and receive side scaling (RSS). Network cards in EC2 instances are virtualized, and currently there are no EC2 instance types that support multiple physical NICs, or NICs that support RSS.
- As mentioned previously, each volume in a Storage Spaces pool must be attached to the same server—**this is not a high-availability architecture**.

Deploying Storage Spaces with Amazon EBS

The remaining sections of this paper will show you how to deploy Storage Spaces between two Windows Server 2012 R2 EC2 instances. One instance will be used as a Storage Spaces server, and the other instance will be used as a client for demonstration purposes. First, you will set up a Storage Spaces volume and then configure a test connection over the SMB protocol. Then, using the same Storage Spaces volume, you will configure a test connection using iSCSI.

> If you choose to follow along with the remaining sections in this paper, you will need to launch Amazon EC2 instances and create EBS volumes in your AWS account. Be aware that there will be billing charges for these resources.

The .zip file provided with this whitepaper includes the following code you can use to fully automate the deployment of Storage Spaces in AWS. These scripts are listed here in the order in which you will run them.

File	Description
StorageSpacesStack.json	AWS CloudFormation template that deploys two EC2 instances and two EBS volumes
storagespaces_smb_server.ps1	PowerShell script that sets up Storage Spaces on the server instance
storagespaces_smb_client.ps1	PowerShell script that tests Storage Spaces over SMB
storagespaces_iscsi_client1.ps1	PowerShell script that obtains the iSCSI identifier of the client
storagespaces_iscsi_server.ps1	PowerShell script that sets up iSCSI on the server
storagespaces_iscsi_client2.ps1	PowerShell script that tests Storage Spaces over iSCSI

Launch Windows Instances with AWS CloudFormation

This diagram shows what the Storage Spaces environment will look like after you create the stack in AWS CloudFormation:

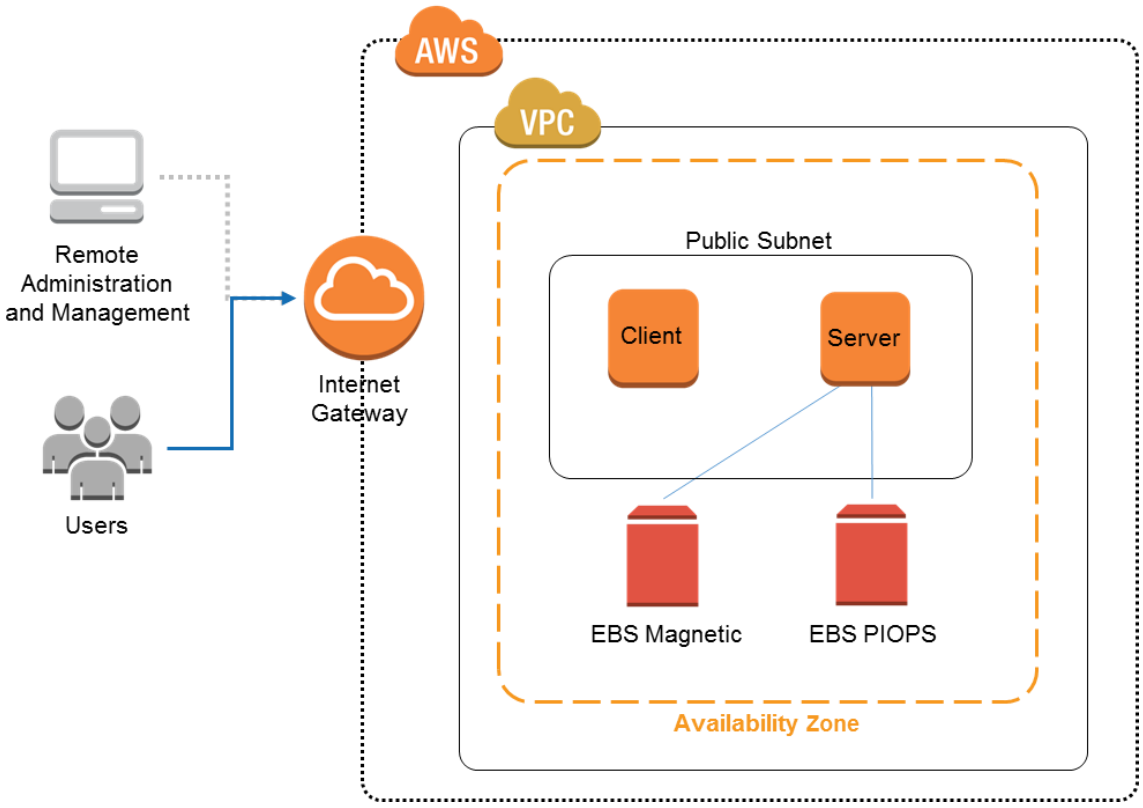

Figure 2: Storage Spaces Environment Built with AWS CloudFormation

Inside the AWS CloudFormation template

One instance is named `server` and the other is named `client`, referring to how the instances will be used with Storage Spaces. In addition to the boot volume, the server instance has two additional EBS volumes attached: a 50-GiB volume and a 40-GiB volume in the `Resources` section. For the purposes of setting the media type in Windows, it simplifies our example if we use a different size for the PIOPS volume than for the magnetic volume. You can test storage tiering by filling the magnetic volume to capacity.

For PIOPS volumes the `IOPS` value must be a maximum of 30 times the Size value, so `IOPS` is set to **1,200** (40 * 30).

Follow these steps to build the environment:

1. Download the .zip file for this paper[6] and unzip the PowerShell scripts and AWS CloudFormation template to a folder on your workstation.
2. Open the AWS CloudFormation template, **StorageSpacesStack.json**, in your preferred text editor.
3. Edit the following fields in the `Parameters` and `Mappings` sections at the top of the template:

Field Name	Section	Description
`InstanceType`	`Parameters`	You may want to use t2.micro instances if your account is eligible for the AWS Free Tier.[7]
`AdminPassword`	`Parameters`	You can enter a Windows Server password to use by default; or you can leave this blank and enter the password when you are creating the stack.
`RDPHome`	`Parameters`	Enter your subnet or IP address in CIDR notation. If you are unsure, you can leave the default as `0.0.0.0/0`.
`KeyName`	`Parameters`	Enter the name of a key pair that you've created in your AWS account in the same region where you're creating the AWS CloudFormation stack.
`AWSRegion2AMI`	`Mappings`	As of the time of publication, the AMI IDs specified in the template were current for each region. But AWS regularly updates the Windows Server AMIs as AWS and Microsoft release patches. If the AMI IDs in this map are outdated, stack creation will fail. You will need to at least update the AMI ID for the region where you are creating the stack.

Note The `RDPHome` value is used in the security group. Using `0.0.0.0/0` opens the required port for RDP to the entire Internet. You should provide a more restricted CIDR that provides access for only your workstation.

Note Make sure you update the AMI ID for the region where you are launching the stack, or stack creation will fail.

4. Log in to your AWS account, and on the **Services** menu, choose **CloudFormation**, **Create Stack**.
5. Make up a name for the stack (e.g. "test"), choose **Upload a template to Amazon S3**, browse to your customized version of **StorageSpacesStack.json**, and then choose **Next**.
6. Edit the values on the **Specify Details** page as needed. In particular, you must provide a Windows password and select a valid keypair. Otherwise, stack creation will fail. Then choose **Next**.
7. On the **Options** page, choose **Next**.
8. On the **Review** page, choose **Next**.

   After a few minutes, you should see the status **CREATE_COMPLETE** displayed for your stack.
9. Choose the **Outputs** tab, and then copy the IP address of the **server** instance.
10. Open your RDP client program (on Windows, it's called Remote Desktop Connection) and connect to the **server** instance with username "administrator" and the password that you specified when creating the stack in AWS CloudFormation.
11. When you are prompted about an invalid certificate, choose **Yes** to proceed. If you can't connect at first, give the instance another a minute to finish booting up.

Provision Storage Spaces with PowerShell

It's possible to set up Storage Spaces using the Windows UI, but this paper sets it up with PowerShell snippets that you can subsequently reuse. In the cloud, instances can be terminated and restarted frequently, and given that, it doesn't make sense to configure Windows Server manually using the mouse. The goal is to be able to redeploy Storage Spaces automatically on any instance. This is a key tenet of the DevOps philosophy: *infrastructure as code*. The benefits of using the scripts is that this entire lab exercise can be completed in 10-20 minutes, and as soon you're done, you can simply delete the stack in AWS CloudFormation and you won't be billed further for the resources you created.

Although the scripts are quick and easy to use, you may want to spend more time browsing in the Storage Spaces UI and the iSCSI UI (both in Server Manager) as you run the scripts to see how PowerShell is building the environment. Also, if you wish to explore the many other PowerShell cmdlets related to storage and iSCSI, using the Server Manager UI will be essential for you to confirm that PowerShell is automating the steps you would do manually in the UI.

Now that you are logged into the **server** instance, follow these steps to set up Storage Spaces.

1. When the Windows desktop for the server instance appears, right-click the **PowerShell** icon in the Windows taskbar, and choose **Windows PowerShell ISE**. It's easier to edit and run multiline commands in PowerShell ISE than it is in the PowerShell console.

2. Press Ctrl+R to show the **Script** pane in ISE, and then copy and paste **storagespaces_smb_server.ps1** from the .zip file you downloaded. When you run this script, you will be asked to confirm if you want to clear all data on the two extra EBS volumes. You can choose **Yes To All**.

```
# Filename: storagespaces_smb_server.ps1
# This script creates a Storage Space drive on the server, and shares it.

$drive = "E"

# Unformat the EBS volumes so they can be pooled.
# Hopefully the Where clauses avoid erasing unintended disks!
Get-Disk | Where-Object Size -GE 40GB | Clear-Disk -RemoveData
Get-PhysicalDisk | Where-Object Size -GE 40GB | Reset-PhysicalDisk

# Create the storage pool
New-StoragePool -FriendlyName ConcretePool -StorageSubsystemFriendlyName
"Storage Spaces*" -PhysicalDisks(Get-PhysicalDisk -CanPool $True)

# Set the media types for storage tiering.
# The Where clauses differentiate SSD/HDD volumes given some rounding.
# The HDD was created as 50 GB in CloudFormation.
# The SSD was created as 40 GB in CloudFormation.
Get-PhysicalDisk | Where-Object Size -GT 39GB | Where-Object Size -LT 41GB |
Set-PhysicalDisk -MediaType SSD
Get-PhysicalDisk | Where-Object Size -GT 49GB | Where-Object Size -LT 51GB |
Set-PhysicalDisk -MediaType HDD

# Create a virtual disk in the pool. The size allows for some rounding.
New-VirtualDisk -FriendlyName vdisk1 -StoragePoolFriendlyName ConcretePool -
Size 87GB -ResiliencySettingName Simple -ProvisioningType Fixed

# Create a partition on the virtual disk and format it
Get-VirtualDisk -FriendlyName vdisk1 |
Get-Disk |
Initialize-Disk -PassThru |
New-Partition -UseMaximumSize -DriveLetter $drive
Format-Volume -DriveLetter $drive -FileSystem NTFS -Confirm:$False

# Let format complete before creating share
Start-Sleep -Seconds 5

# Share a folder on the disk
$path = ("{0}:\MyShare" -f $drive)
New-Item $path -Type Directory
New-SmbShare -Name Myshare -Path $path -FullAccess Administrator
```

Notes on the PowerShell Script

- To be in the *primordial pool* for Storage Spaces, the `CanPool` property must be showing as `False` if you run the `Get-PhysicalDisk` cmdlet or look in the Storage Spaces UI in Server Manager. When a drive is formatted, it will have a partition type of MBR (master boot record) or GPT (general partition table), and it won't be available for pooling. Since we provisioned the EBS volumes with the EC2 instance in AWS CloudFormation, the volumes are formatted. So in order for them to be eligible to be pooled, the script uses `Reset-PhysicalDisk` to change the partition type to `Unknown`. And before the script can run `Reset-PhysicalDisk`, it must run `Clear-Disk`.
- The next step is to create the *concrete pool* with `New-StoragePool`. The asterisk in the `StorageSubsystemFriendlyName` is a wildcard to match the computer name.
- Because the disks are virtualized by AWS, Windows cannot automatically detect the media type, but Windows must know the media type for storage tiering. The `Set-PhysicalDisk` cmdlet can set the media type, but it only works after the volumes have been placed in a storage pool. Per the AWS CloudFormation template, the magnetic volume is 50 GiB and the PIOPS SSD volume is 40 GiB. Certainly you can change these values in the script, but you must be careful to assign the correct media type to each volume. Note that SSD stands for solid-state drive, and HDD stands for hard disk drive (HDD is synonymous with magnetic).
- The `New-VirtualDisk` cmdlet creates a storage space (or virtual disk) from the concrete pool. `ResiliencySettingName` allows you to specify `Simple` (RAID 0), `Mirror` (RAID 1), or `Parity` (RAID 5). `Simple` is the default; `Mirror` and `Parity` increase data durability, but at the cost of reduced storage capacity. Using `Mirror` or `Parity` requires at least three disks in the concrete pool. You can also choose `Fixed` or `Thin` provisioning. `Thin` provisioning allows you to allocate and reclaim capacity dynamically; `Fixed` is the default. The `Size` parameter is set to 87 GiB, so it uses all the capacity of the 40 GiB and 50 GiB EBS volumes you attached to the instance. (There is rounding in the numbers and some loss of space for overhead.)
- The script then creates a partition on the virtual disk and formats it. You can optionally use the ReFS file system instead of NTFS in `Format-Volume`; however, ReFS is not supported with iSCSI. The script does a quick format, but it doesn't zero the disk. Prior to October 2015, AWS recommended doing a

full format of EBS volumes to get the best performance. However, that is no longer the case.

- The last block of code creates a folder on the server and shares it for the client to access over the network.

Use Storage Spaces from the Client Instance

This section explains a quick test you can run from the client instance to access the Storage Spaces drive you created on the server instance.

1. Open another Remote Desktop Connection and log in to the **client** instance using the IP address from the **Outputs** tab in the AWS CloudFormation console.
2. Open PowerShell ISE, press Ctrl+R, and paste the **storagespaces_smb_client.ps1** script from the download .zip file into the **Script** pane.
3. Copy the public IP address of the **server** instance from the AWS Management Console into the first executable line of the script, and then run it.

   If you get an error that the shared path does not exist, you may need to allow it more time to be created on the **server** instance.

```
# Filename: storagespaces_smb_client.ps1
# © 2015, Amazon Web Services, Inc. or its affiliates.  All rights reserved.
#
# This script connects to the Storage Space that was created on the server.
#
# Insert the public or private IP address of your server instance.
$serverIP = "YOUR SERVER IP ADDRESS"

# Browse the shared folder to acquire access.
Get-ChildItem "\\$serverIP\MyShare\"

# Copy a text file to the shared folder on the server.
@'
Hello AWS!
'@ > \\$serverIP\MyShare\client.txt
```

The script copies a `here-string` file from the client instance to the shared folder on the server instance. To keep it simple, this example just connects to the server by its IP address. Since the redirect command uses the UNC path without first mapping a drive letter, you need to browse the server folder with `Get-ChildItem` first to establish authentication.

The AWS CloudFormation template opens port 445 between the instances, which is sufficient to use the SMB protocol with IP addresses. If you would like to use the NetBIOS computer names in your share names, you will also need to open ports 137-139.

Congratulations! Now you've deployed Storage Spaces in AWS, and the virtual disk is ready to use.

Deploy an iSCSI Target with PowerShell

In this section, you'll see how you can use PowerShell on the **server** instance to create an iSCSI virtual disk inside Storage Spaces, which you can then share to remote client machines.

Before you can set up an iSCSI target on the server, you need to run a script to start the iSCSI Initiator service on the **client** and obtain the client IQN.

1. Open Windows PowerShell ISE on the client instance, and run the script named **storagespaces_iscsi_client1.ps1** from the .zip file you downloaded.

```
# Filename: storagespaces_iscsi_client1.ps1
# © 2015, Amazon Web Services, Inc. or its affiliates.  All rights reserved.
#
# This script starts iscsi initiator on the client and obtains the IQN.
#
# start the iscsi initiator service
Set-Service msiscsi –StartupType "Automatic"
Start-Service msiscsi

# Copy/paste the output of this command, without the brackets,
# into storagespaces_iscsi_server.ps1
iscsicli
```

2. Carefully select and copy the IQN value (which is the last line of output from the script), and then paste it into the script **storagespaces_iscsi_server.ps1**, which you'll run in the next step on the **server** instance.

```
# Filename: storagespaces_iscsi_server.ps1
# © 2015, Amazon Web Services, Inc. or its affiliates.  All rights reserved.
#
# This script creates an iscsi drive on the server with CHAP.

# Run iscsicli on the client and insert here.
$clientIQN = "OUTPUT FROM iscsicli ON THE CLIENT, WITHOUT BRACKETS"

$username = "user"
$password = "12characters"    # must be 12-16 characters long
$drive = "E"
$size = 85GB
$target = "mytarget"

# Install Iscsi Target Feature
Install-WindowsFeature -Name FS-iSCSITarget-Server -IncludeManagementTools

# Enable iSCSI ports in Windows Firewall
netsh advfirewall firewall add rule name="iscsi-in" protocol=TCP dir=in
localport="860,3260" action=allow
netsh advfirewall firewall add rule name="iscsi-out" protocol=TCP dir=out
localport="860,3260" action=allow

# Create a virtual disk, without formatting
$vdisk = $drive+":\iscsi\vdisk.vhdx"
New-IscsiVirtualDisk -Path $vdisk -Size $size -UseFixed –DoNotClearData

# create a target called "mytarget" for the client instance to access
New-IscsiServerTarget -TargetName $target -InitiatorIds "IQN:$clientIQN"

# Require initiators to connect via CHAP as user "user" with password
$SecurePassword = ConvertTo-SecureString -String $password -AsPlainText -
Force
$chap = New-Object System.Management.Automation.PSCredential($username,
$SecurePassword)
Set-IscsiServerTarget –TargetName $target -EnableChap $true -Chap $chap

# Map the iscsi target name to the virtual disk
Add-IscsiVirtualDiskTargetMapping -TargetName $target -Path $vdisk
```

Notes on the PowerShell Script

- The iSCSI PowerShell API documentation claims that **New-IscsiServerTarget** can receive an IP address or an initiator IQN as an identifier for authentication. However, preliminary tests for this whitepaper were successful only when we used the client IQN, which is why you need to run the `iscsicli` command on the client before running `New-IscsiServerTarget` on the server.
- The CHAP password must be 12-16 characters in length.

You have now created an iSCSI target and enabled CHAP for the virtual disk in Storage Spaces. In the next section you'll see how to initiate an iSCSI connection to the target from a client machine.

Connect to the iSCSI Target with PowerShell

Now that you have an iSCSI target on the **server** instance, you can initiate a connection to it from the **client** instance to test it.

1. On the client instance, in Windows PowerShell ISE, paste the **storagespaces_iscsi_client2.ps1** script from the .zip file you downloaded.
2. At the top of the file, edit the values for the server IP address (copy the IP address from the **Outputs** tab in AWS CloudFormation) and the server IQN (run `Get-IscsiServerTarget` on the server and copy the IQN value).
3. When you run the script, look behind the Windows PowerShell ISE for the format disk popup window, and choose **Yes** to let the drive format.

```
# Filename: storagespaces_iscsi_client2.ps1
# © 2015, Amazon Web Services, Inc. or its affiliates.  All rights reserved.
#
# This script formats the iscsi volume on the client and writes to it.

# Edit these values...
$serverIP = "YOUR SERVER IP ADDRESS"
$serverIQN = "RUN Get-IscsiServerTarget ON THE SERVER AND COPY TARGET IQN"
$username = "user"
$password = "12characters"   # must be 12-16 characters
$drive = "E"

# Enable iSCSI ports in Windows Firewall
netsh advfirewall firewall add rule name="iscsi-in" protocol=TCP dir=in
localport="860,3260" action=allow
netsh advfirewall firewall add rule name="iscsi-out" protocol=TCP dir=out
localport="860,3260" action=allow

# Connect to ISCSI
New-IscsiTargetPortal -TargetPortalAddress $serverIP
Connect-IscsiTarget -NodeAddress $serverIQN -AuthenticationType ONEWAYCHAP -
ChapUsername $username -ChapSecret $password
Start-Sleep -Seconds 5  # give the connection time to initialize

# Format the drive and bring it online
$number = (Get-Disk | Where Size -GT 84GB | Where Size -LT 90GB | Where
PartitionStyle -EQ 'RAW').Number
Set-Disk -Number $number -IsReadOnly 0
Set-Disk -Number $number -IsOffline 0
Initialize-Disk -Number $number
New-Partition -DiskNumber $number -DriveLetter $drive -UseMaximumSize
Start-Sleep -Seconds 5
Format-Volume -DriveLetter $drive -FileSystem NTFS -Force
Start-Sleep -Seconds 5

# Copy a text file to the iscsi virtual disk.
$filename = $drive+":\iscsi_test.txt"
@'
Hello AWS from ISCSI!
'@ > $filename
```

Notes on the PowerShell Script

- In testing, the `Format-Volume` cmdlet would sometimes pop up the UI for confirmation even though we specified `-Force` or `-Confirm:$False`. Perhaps you can experiment with it and determine how to avoid the UI.
- The PowerShell iSCSI documentation indicates that `OneWayCHAP` can be mixed case, but it worked only when we used uppercase in testing.
- Although the target script must encrypt the CHAP password before calling the API, the initiator script calls the API with the password in plaintext.
- According to the documentation, iSCSI connections are persistent across reboots unless overridden by specifying `-IsPersistent $False`. You can also use the `Register-IscsiSession` cmdlet to register an iSCSI target as a "favorite," so it will reconnect automatically after rebooting Windows.

Conclusion

This paper described current limitations and some potential use cases for running Windows Server Storage Spaces in AWS. The DevOps "infrastructure as code" tenet was followed to automate the deployment process, making this a quick and repeatable lab exercise that is also easy to shut down.

By integrating Amazon EBS with Windows Storage Spaces, administrators can deliver powerful and low-cost, software-based SAN capabilities in the cloud, with very high agility, durability, and scalability.

Contributors

The following individuals and organizations contributed to this document:

- Scott Zimmerman, Partner Solutions Architect, AWS
- Faris Malaeb, Senior Infrastructure Engineer, PowerShellCenter.com

Further Reading

For additional information, please consult the following sources:

AWS Services

- Getting Started with Amazon EC2 Windows Instances
  http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/EC2Win_GetStarted.html
- AWS re:Invent 2015 | (STG403) Amazon EBS: Designing for Performance
  https://www.youtube.com/watch?v=2wKgha8CZ_w
- Amazon Elastic Block Store
  http://aws.amazon.com/ebs/
- Amazon EBS Volume Performance on Windows Instances
  http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/EBSPerformance.html
- Storage Options in the AWS Cloud
  http://media.amazonwebservices.com/AWS_Storage_Options.pdf

Windows Storage Spaces

- Storage Spaces Direct in Windows Server 2016
  https://msdn.microsoft.com/en-us/library/mt126109.aspx
- Hands-on Lab to setup Storage Spaces in AWS using the Windows UI
  https://run.qwiklabs.com/
- Storage Spaces Frequently Asked Questions (FAQ)
  http://social.technet.microsoft.com/wiki/contents/articles/11382.storage-spaces-frequently-asked-questions-faq.aspx

Network Connectivity

- Amazon VPC Network Connectivity Options
  http://media.amazonwebservices.com/AWS_Amazon_VPC_Connectivity_Options.pdf
- Implementing Microsoft DirectAccess and NAT in the AWS Cloud
  http://aws.amazon.com/windows/resources/whitepapers/

- Amazon Direct Connect
  http://aws.amazon.com/directconnect/
- Windows netsh command line reference (for opening firewall ports)
  https://technet.microsoft.com/en-us/library/dd734783(v=ws.10).aspx

iSCSI and Windows PowerShell

- iSCSI Security: https://technet.microsoft.com/en-us/library/cc754658.aspx
- Introduction of iSCSI Target in Windows Server 2012
  http://blogs.technet.com/b/filecab/archive/2012/05/21/introduction-of-iscsi-target-in-windows-server-2012.aspx
- Storage Cmdlets in Windows PowerShell
  https://technet.microsoft.com/en-us/library/hh848705(v=wps.630).aspx
- iSCSI Cmdlets in Windows PowerShell
  https://technet.microsoft.com/en-us/library/hh826099.aspx
- iSCSI Target Cmdlets in Windows PowerShell
  https://technet.microsoft.com/en-us/library/jj612803.aspx
- iSCSI Security:
  https://technet.microsoft.com/en-us/library/cc754658.aspx
- iSCSI Target Cmdlet Reference
  http://blogs.technet.com/b/filecab/archive/2012/06/08/iscsi-target-cmdlet-reference.aspx
- Reference pages and examples for all PowerShell storage cmdlets
  http://www.powershellcenter.com/

Notes

[1] https://msdn.microsoft.com/en-us/library/mt126109.aspx

[2] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html

[3] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-ec2-config.html

[4] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-workload-demand.html

[5] http://aws.amazon.com/ebs/pricing/

[6] http://d0.awsstatic.com/whitepapers/StorageSpacesCode.zip

[7] https://aws.amazon.com/free/