Best Practices for Deploying Microsoft SQL Server on AWS

May 2020

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. 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.

Contents

Abstract

This whitepaper focuses on best practices to attain the most value for the least cost when running Microsoft SQL Server on AWS. Although for many general-purpose use cases, Amazon Relational Database Service (Amazon RDS) for Microsoft SQL Server provides an easy and quick solution, in this paper we focus on scenarios where you need to push the limits to satisfy your special requirements.

In particular, this whitepaper explains how you can minimize your costs, maximize availability of your SQL Server databases, optimize your infrastructure for maximum performance, and tighten it for security compliance, while also enabling operational excellence for ongoing maintenance. The flexibility of AWS services, combined with the power of Microsoft SQL Server, can provide expanded capabilities for those who seek innovative approaches to optimize their applications and transform their businesses.

The main focus of this paper is on the capabilities available in Microsoft SQL Server 2019, which is the most current version at the time of publication. Existing databases that are running on previous versions (that is, 2008, 2012, 2014, 2016, and 2017) can be migrated to SQL Server 2019 and run in compatibility mode.

Mainstream and extended support for SQL Server 2000, 2005 and 2008 has been discontinued by Microsoft. Any database running on those versions of SQL Server must be upgraded to a supported version first. Although it is possible to run those versions of SQL Server on AWS, that discussion is outside the scope of this whitepaper.

Introduction

AWS offers the best cloud for SQL Server, and it is the right cloud platform for running Windows-based applications today and in the future. SQL Server on Windows or Linux on Amazon EC2 enables you to increase or decrease capacity within minutes, not hours or days. You can commission one, hundreds, or even thousands of server instances simultaneously.

Customers have been running Windows workloads on AWS for over a decade. AWS runs nearly two times more Windows Server instances than the next largest cloud provider, according to an IDC report. AWS continues to be the most preferred option for deploying and running Microsoft SQL Server. This is due to the unique combination of breadth and depth of services and capabilities offered by AWS, providing the optimum platform for MS SQL Server workloads.

Requirements for running SQL Server often fall under following categories:

- High availability and disaster recovery
- Performance
- Security
- Cost
- Monitoring and maintenance

These requirements map directly to the five pillars of the AWS Well-Architected Framework, namely:

- Reliability
- Performance efficiency
- Security
- Cost optimization
- Operational excellence

This paper will discuss each of these requirements in further detail, along with best practices using AWS services to address them.

High Availability and Disaster Recovery

Every business seeks data solutions that can address their operational requirements. These requirements often translate to specific values of the Recovery Time Objective (RTO), and Recovery Point Objective (RPO). The RTO indicates how long the business can endure database and application outages, and the RPO determines how much data loss is tolerable. For example, a RTO of one hour tells us that, in the hapless event of an application outage, the recovery plans should aim to bring the application back online within one hour. Likewise, a RPO of zero indicates that, should there be any minor or major issues impacting the application, there should be no data loss after the application is brought back online.

The combination of RTO and RPO requirements dictates what solution should be adopted. Typically, applications with RPO and RTO values close to zero need to use a high availability (HA) solution, whereas disaster recovery (DR) solutions could be used for those with higher values. In many cases, HA and DR solutions can also be mixed to address more complex requirements.

Microsoft SQL Server offers several High Availability and Disaster Recovery (HA/DR) solutions, each suitable for specific requirements. The following table comparatively shows these solutions:

Table 1: HA/DR options in Microsoft SQL Server

Solution	HA	DR	Enterprise edition	Standard edition
Log shipping	No	Yes	Yes	Yes
Mirroring (deprecated)	Yes	Yes	Yes	Yes (Full safety only)
Always On availability groups	Yes	Yes	Yes	Yes (2 nodes)[1]
Always On failover	Yes	No	Yes	Yes (2 nodes)

[1] Always On basic availability groups in SQL Server 2019 Standard edition support up to two passive replicas (in addition to the primary replica) for a single database per availability group. If you need multiple databases in HA mode, a separate availability group needs to be defined for each database. The maximum number of replicas using Microsoft SQL Server 2016 and 2017 drops to one. Basic availability groups are not available with lower versions of SQL Server.

cluster instances				
Distributed availability groups	Yes	Yes	Yes	No

These solutions rely on one or more secondary servers with SQL Server running as active or passive standby. Based on the specific HA/DR requirements, these servers can be located in close proximity to each other or far apart.

In AWS, you can choose between low latency or an extremely low probability of failure. You can also combine these options to create the solution that is most suitable to your use case. In this paper, we look at these options and how they can be used with SQL Server workloads.

Availability Zones and Multi-AZ Deployment

AWS Availability Zones are designed to provide separate failure domains, while keeping workloads in relatively close proximity for low latency communications. Availability Zones are a good solution for synchronous replication of your databases using Mirroring, Always On Availability Groups, Basic Availability Groups, or Failover Cluster Instances. SQL Server provides zero data loss and, when combined with the low-latency infrastructure of Availability Zones, provides high performance.

This is one of the main differences between most on-premises deployments and AWS. For example, Always On Failover Cluster Instances (FCI) is often used inside a single data center. This is because all nodes in an FCI cluster must have access to the same shared storage. Locating these nodes in different data centers could degrade performance. However, with AWS, FCI nodes can be located in separate Availability Zones and still provide high performance because of the low-latency network link between all Availability Zones within a Region.

This feature enables a higher level of availability and could eliminate the need for a third node, which is often coupled with an FCI cluster for disaster recovery purposes.

SQL Server FCI relies on shared storage being accessible from all nodes participating in FCI. Amazon FSx for Windows File Server is a fully managed service providing shared storage that automatically replicates the data synchronously across two Availability Zones, provides high availability with automatic failure detection, failover, and failback, and fully supports the SMB Continuous Availability (CA) feature. This enables you to simplify your SQL Server Always On deployments and use Amazon FSx as storage tier for MSSQL FCI.

Scenarios where Amazon FSx is applicable for performance tuning and cost optimization are discussed in subsequent sections of this document.

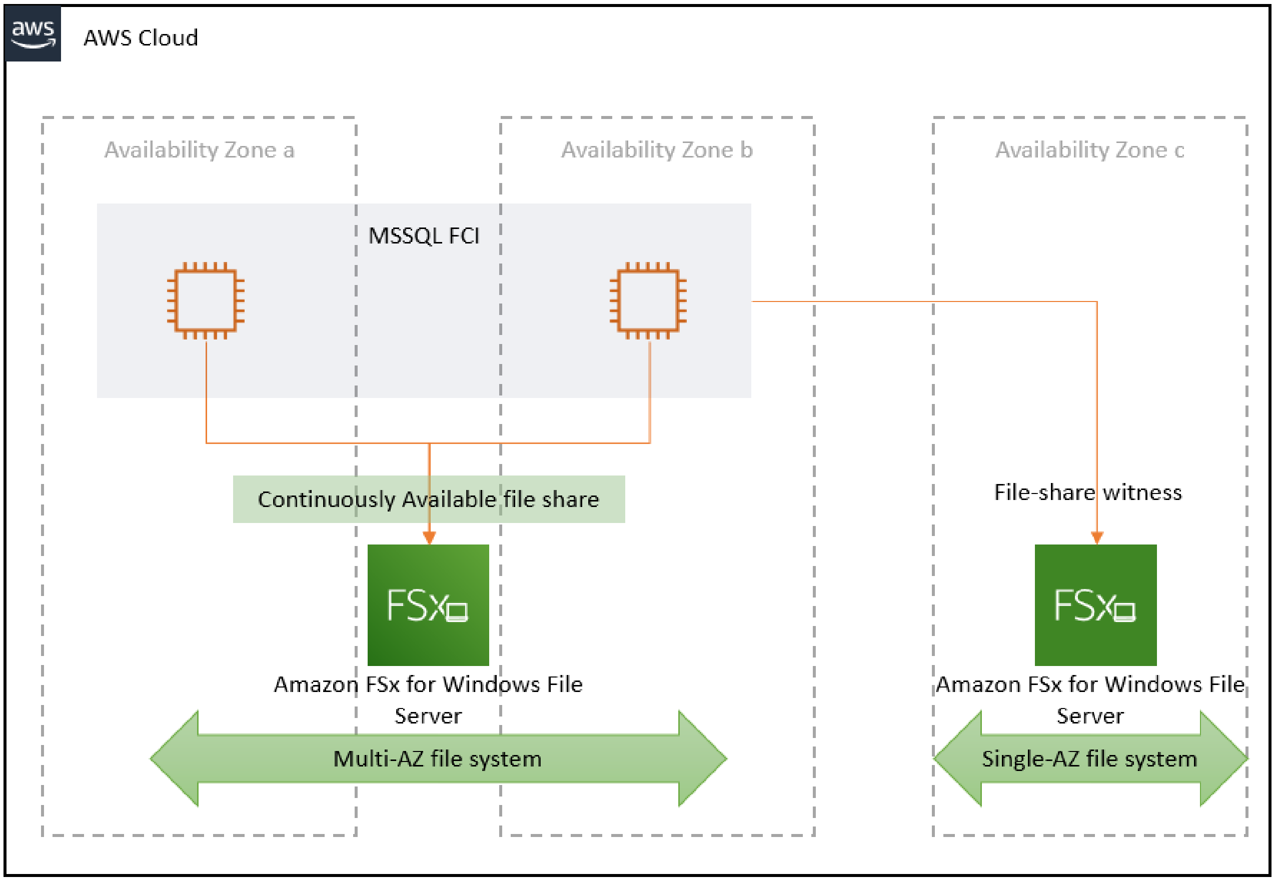

Figure 1: SQL Server Always On Failover Cluster Instances in AWS

Cluster Placement Groups and Enhanced Networking

Traditionally, higher availability is often at the expense of lower performance. In AWS, there are ways to mitigate odds of this dichotomy.

Amazon EC2 enables you to deploy a number of EC2 instances inside a cluster placement group. This means that those EC2 instances are not only inside a single AZ, but also, to ensure minimum network latency, within close physical proximity in the same data center. To gain the highest bandwidth on AWS, you can leverage enhanced networking and Elastic Network Adapter (ENA), or the new Elastic Fabric Adapter (EFA) which, when combined with M5n, M5dn, R5n, and R5dn instances, can provide up to 100-Gbps bandwidth.

At first glance, this might appear to conflict with HA requirements because the close proximity of servers increases the likelihood of simultaneous failures. However, this approach can be combined with Multi-AZ, to provide higher levels of availability. For example, you can create an Always On FCI inside a cluster placement group and add that cluster as a node to a Multi-AZ Always On Availability Group using asynchronous replication. This arrangement gives you maximum performance while providing automatic failover to another local instance if a process or instance fails.

Additionally, in the event of an AZ failure, you can manually fail over your database to the second AZ. Relatively low latency between Availability Zone provides near-zero data-loss, even with asynchronous replication in place. This is an example of HA with close to zero performance penalty for applications that are sensitive to latency, combined with a DR solution within the same AWS Region.

As noted previously, although using a cluster placement group reduces network latency to a minimum, it increases the likelihood of simultaneous failures for instances running inside it. For example, if you have two EC2 instances, both running on the same physical host, or both sharing the same power source and network switch, a failure of any of these underlying components results in the failure of both of your EC2 instances. Instead, you may prefer to sacrifice the lower latency of cluster placement groups in exchange for lowering the probability of simultaneous failures of your EC2 instances. In such cases, you can leverage spread placement groups to ensure that your instances are placed far enough apart to minimize chances of a partial or full failure at the same time.

Multi-Region Deployments

For those workloads that require even more resilience against unplanned events, you can leverage the global scale of AWS to ensure availability under almost any circumstances.

Since by default Amazon Virtual Private Cloud (Amazon VPC) is confined within a single Region, for a multi-region deployment, you would need to establish connectivity between your SQL Server instances that are deployed in different Regions. In AWS, there are a number of ways to do this, each suitable for a range of requirements:

1. VPC peering: Provides an encrypted network connectivity between two VPCs. The traffic flows through the AWS networking backbone, therefore eliminating latency and other hazards of the internet.

2. AWS Transit Gateway: If you need to connect two or more VPCs or on-premises sites, you can use AWS Transit Gateway to simplify management and configuration overhead of establishing network connection between them.
3. VPN connections: AWS VPN solutions are especially useful when you need to operate in a hybrid environment, and connect your AWS VPCs to your on-premises sites and clients.
4. VPC Sharing: If your applications or other clients are spread across multiple AWS accounts, an easy way to make your SQL Server instance available to all of them is using VPC Sharing. A shared VPC can also be connected to other VPCs using AWS Transit Gateway, AWS VPN CloudHub, VPN connections, or VPC peering. These connections are useful when workloads are spread across multiple accounts and Regions.

If you have applications or users that are deployed in remote Regions which need to connect to your SQL Server instances, you can use the AWS Direct Connect feature that provides connectivity from any Direct Connect connection to all AWS Regions.

Although it is possible to have synchronous replication in a multi-region SQL Server deployment, the farther apart your selected Regions are, the more severe the performance penalty is for a synchronous replication. Often the best practice for multi-region deployments is to establish an asynchronous replication, especially for Regions that are geographically distant. For those workloads that come with aggressive RPO requirements, asynchronous multi-region deployment can be combined with a Multi-AZ or Single-AZ synchronous replication. You can also combine all three methods into a single solution. However, these combinations would impose a significant increase in your SQL Server license costs, which must be considered as part of your planning.

In cases involving several replicas across two or more Regions, distributed availability groups might be the most suitable option. This feature allows you to combine availability groups deployed in each Region into a larger distributed availability group.

Distributed availability groups can also be used to increase the number of read replicas. A traditional availability group allows up to 8 read replicas. This means that you can have a total of 9 replicas, including the primary. Using a distributed availability group, a second availability group can be added to the first, increasing the total number of replicas to 18. This process can be repeated with a third availability group and a second distributed availability group. The second distributed availability group can be configured to include either the first or second availability groups as its primary. Distributed availability groups is the means through which SQL Server Always On can achieve virtually unlimited scale.

Another use case of a distributed availability group is for zero downtime database migrations, when during migration a read-only replica is available at target destination. The independence of SQL Server Distributed Availability Group from Active Directory and Windows Server Failover Cluster (WSFC) is the main benefactor for these cases. It allows you to keep both sides of the migration synchronized without having to worry about the complexities of Active Directory or WSFC. You can refer to this blog post for more details.

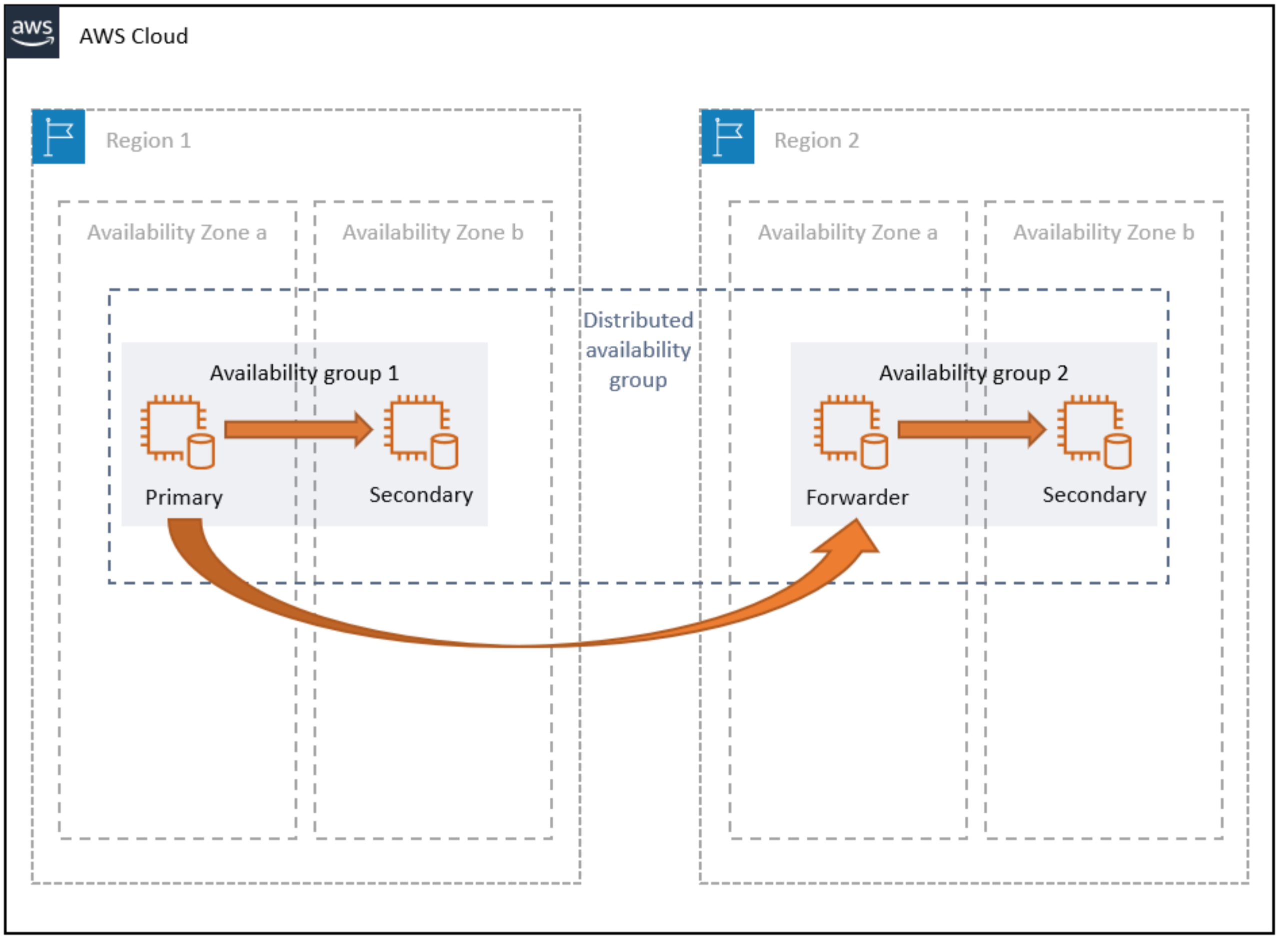

Figure 2: SQL Server Distributed Availability Group in AWS

Disaster Recovery

Similar to high availability (HA) solutions, disaster recovery (DR) solutions also require a replica of SQL Server databases in another server. However, for DR, the other server is often in a remote site far away from the primary site. This means higher latency, and therefore, lower performance if relying on HA solutions that use synchronous replication.

DR solutions often rely on asynchronous replication of data. Similar to HA, DR solutions are also based on either block-level or database-level replication. For example, SQL Server Log Shipping replicates data at the database-level, while Windows Storage Replica can be used to implement block-level replication.

DR solutions are selected based on their requirements, such as cost, RPO, RTO, complexity, and the effort to implement each solution.

In addition to common SQL Server DR solutions, such as Log Shipping and Windows Storage Replica, AWS also provides CloudEndure Disaster Recovery. You can use CloudEndure Disaster Recovery to reduce downtime to a few minutes, protect against data loss for sub-second RPO, simplify implementation, increase reliability, and decrease the total cost of ownership. CloudEndure is an agent-based solution that replicates entire virtual machines, including the operating system, all installed applications, and all databases, into a staging area. The staging area contains low-cost resources automatically provisioned and managed by CloudEndure Disaster Recovery. This greatly reduces the cost of provisioning duplicate resources. Because the staging area does not run a live version of your workloads, you don't need to pay for duplicate software licenses or high performance compute. Rather, you pay for low cost compute and storage. The fully provisioned recovery environment, with the right-sized compute and higher-performance storage required for recovered workloads, is only launched during a disaster or drill.

AWS also makes CloudEndure available at no additional cost for migration projects.

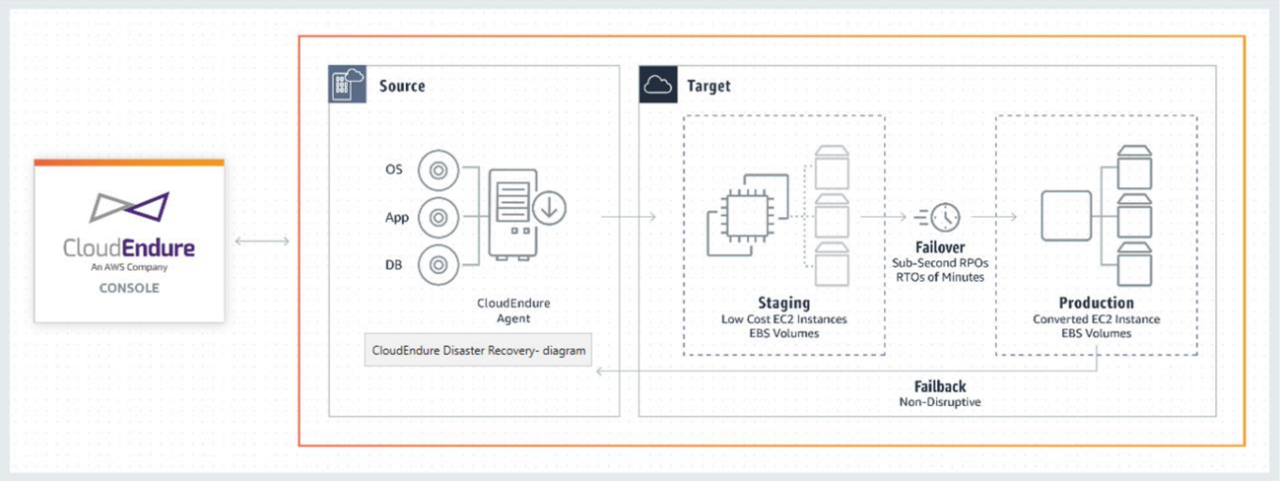

Figure 3: CloudEndure Disaster Recovery

Performance Optimization

In some cases, maximizing performance might be your utmost priority. Both SQL Server and AWS have several options to substantially increase performance of your workloads.

The first and most effective way to improve the performance of your workloads is by optimizing your applications and database schemas. You might also be able to significantly improve application performance by changing your application to use purpose-built database solutions instead of a traditional relational database. AWS database options for modernizing applications include Amazon DynamoDB for key-value and document storage, Amazon Neptune for graph data, Amazon Quantum Ledger Database, and Amazon Managed Blockchain.

To modernize applications that still need to use a relational database, many businesses choose Amazon Aurora. Aurora is a managed, open-source, enterprise-grade RDBMS service built for the cloud, which is fully compatible with MySQL and PostgreSQL. It is up to five times faster than standard MySQL databases and three times faster than standard PostgreSQL databases. It provides the security, availability, and reliability of commercial databases at one tenth the cost. Amazon Aurora is fully managed by Amazon Relational Database Service (RDS), which automates time-consuming administration tasks like hardware provisioning, database setup, patching, and backups. You can also benefit from Aurora’s serverless capability to achieve highest degrees of cost optimization coupled with steady performance, without having to manage your servers.

Another area to consider for performance tuning is your physical database schema. Careful physical design, defining non-clustered indexes, and using horizontal partitioning of tables and indexes, can drastically increase your database performance. For analytic processing of transactional data, you might leverage columnstore indexes, a feature of Microsoft SQL Server, which can provide up to ten times performance and data compression improvements.

Sometimes, you can solve your scaling problems by logically slicing your data into smaller chunks and then hosting each chunk on a separate server. This technique is called sharding and is particularly useful for scaling write-heavy databases.

However, you might have already done application and DB level optimization, or your situation might not lend itself to such practices (for example, having to run legacy line-of-business applications, or projecting impending load increase, etc.) In these cases, the next place to focus is on your infrastructure and assess how it can be reconfigured to solve or alleviate your problem. The next sections focus on infrastructure design options for maximizing performance of Microsoft SQL Server in the AWS Cloud.

Using Amazon Elastic Block Store (Amazon EBS)

Amazon EBS is a Single-AZ block storage service with a number of flexible options to cater to diverse requirements. When it comes to maximizing performance with consistent and predictable results on a single volume, using a Provisioned IOPS volume type (io1) is the easiest choice. You can provision up to 64,000 IOPS per io1 EBS volume (based on 16 KiB I/O size), along with 1000-MB/s throughput.

If you need more IOPS and throughput than provided by a single EBS volume, you can create multiple volumes and stripe them in your Windows or Linux instance (Microsoft SQL Server 2017 and later can be installed on both Windows and Linux systems). Striping allows you to further increase the available IOPS per instance up to 80,000, and throughput per instance up to 2,375 MB/s.

One point to remember is to use EBS-optimized EC2 instance types. This means that a dedicated network connection is allocated to serve requests between your EC2 instance and the EBS volumes attached to it.

While you can use a single Provisioned IOPS (io1) volume to meet your IOPS and throughput requirements, General Purpose SSD (gp2) volumes offer a better balance of price and performance for SQL Server workloads when configured appropriately.

General Purpose SSD (gp2) volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods. This ability is well suited to SQL Server. The IOPS load generated by a relational database like SQL Server tends to spike frequently. For example, table scan operations require a burst of throughput, while other transactional operations require consistent low latency.

One of the major benefits of using EBS volumes, is the ability to create point-in-time and instantaneous EBS snapshots. This feature copies the EBS snapshot to Amazon S3 infrastructure, which provides 99.999999999% durability. Despite EBS volumes being confined to a single AZ, EBS snapshots can be restored to any AZ within the same Region. Note that block-level snapshots are not the same as database backups and not all features of database backups are attainable this way. Therefore, this method is often combined and complemented with a regular database backup plan.

Although each EBS volume can be as large as 16 TB, it could take a long time to transfer all of its data to Amazon S3, but EBS snapshots are always point-in-time. This means that SQL Server and other applications can continue reading and writing to and from the EBS volume while data is being transferred in the background.

When you restore a volume from a snapshot, the volume is immediately available to applications for read and write operations. However, it takes some time until it gets to its full performance capacity. Using Amazon EBS fast snapshot restore, you can eliminate the latency of I/O operations on a block when it is accessed for the first time. Volumes created using fast snapshot restore instantly deliver all of their provisioned performance.

You can use AWS Systems Manager Run Command to take application-consistent EBS snapshots of your online SQL Server files at any time, with no need to bring your database offline or in read-only mode. The snapshot process uses Windows Volume Shadow Copy Service (VSS) to take image-level backups of VSS-aware applications. Microsoft SQL Server is VSS-aware and is perfectly compatible with this technique. It is also possible to take VSS snapshots of Linux instances, however, that process requires some manual steps, because Linux does not natively support VSS.

You can also take crash-consistent EBS snapshots across multiple EBS volumes, attached to a Windows or Linux EC2 instance, without using orchestrator applications. Using this method, you only lose uncommitted transactions and writes that are not flushed to the disk. SQL Server is capable of restoring databases to a consistent point before the crash time.

EBS volumes are simple and convenient to use, and in most cases effective, too. However, there might be circumstances where you need even higher IOPS and throughput than what is achievable using Amazon EBS.

Instance Storage

Storage-optimized EC2 instance types use fixed-size local disks and a variety of different storage technologies are available. Among these, Non-Volatile Memory express (NVMe) is the fastest technology with the highest IOPS and throughput. The i3 class of instance types provides NVMe SSD drives, for example, i3.16xlarge, comes with eight disks, each with 1.9 TB of storage. When selecting storage-optimized EC2 instance types for maximum performance, it is essential to understand that some of the smaller instance types provide instance storage that is shared with other instances. These are virtual disks that reside on a physical disk attached to the physical host. By selecting a bigger instance type, such as i3.2xlarge, you ensure that there is a 1:1 correspondence between your instance store disk and the underlying physical disk. This ensures consistent disk performance and eliminates the noisy-neighbor problem.

Instance disks are ephemeral and only live as long as their associated EC2 instance. If the EC2 instance fails, or is stopped or terminated, all of its instance storage disks are wiped out and the data stored on them is irrecoverable. Unlike EBS volumes, instance storage disks cannot be backed up using a snapshot. Therefore, if you choose to use EC2 instance storage for your permanent data, you need to provide a way to increase its durability.

One suitable use for instance storage may be the tempdb system database files because those files are recreated each time the SQL Server service is restarted. SQL Server drops all tempdb temporary tables and stored procedures during shut down. As a best practice, the tempdb files should be stored on a fast volume, separate from user databases. For the best performance, ensure that the tempdb data files within the same filegroup are the same size and stored on striped volumes.

Another use for EC2 instance storage is the buffer pool extension. Buffer pool extension is available on both the Enterprise and Standard editions of Microsoft SQL Server. This feature uses fast random-access disks (SSD) as a secondary cache between RAM and persistent disk storage, thus, striking a balance between cost and performance when running workloads on SQL Server.

Although instance storage disks are the fastest available to EC2 instances, their performance is capped at the speed of the physical disk. You can go beyond the single disk maximum by striping across several disks.

You could also use instance storage disks as the cache layer in a Storage Spaces (for single Windows instances) and Storage Spaces Direct (for Windows Server failover clusters) storage pools.

Amazon FSx for Windows File Server

Amazon FSx for Windows File Server is another storage option for SQL Server on Amazon EC2. This option is suitable for three major use-cases:

- As shared storage used by SQL Server nodes participating in a Failover Cluster Instance
- As file-share witness to be used with any SQL Server cluster on top of Windows Server Failover Cluster
- As an option to attain higher throughput levels than available in dedicated EBS optimization

The first two cases were discussed in an earlier section of this document. To better understand the second case, notice that EBS throughput depends on EC2 instance size. Smaller EC2 instance sizes provide lower EBS throughput, therefore to attain EBS higher throughput you need bigger instance sizes. However, higher instance sizes cost more. If a workload leaves a big portion of its network bandwidth unused, but requires higher throughput to access underlying storage, using a shared file system over SMB may unlock its required performance, while reducing cost by using smaller EC2 instance sizes.

Amazon FSx provides fast performance with baseline throughput up to 2 GB/second per file system, hundreds of thousands of IOPS, and consistent submillisecond latencies. To provide the right performance for your SQL instances, you can choose a throughput level that is independent of your file system size. Higher levels of throughput capacity also come with higher levels of IOPS that the file server can serve to the SQL Server instances accessing it. The storage capacity determines not only how much data you can store, but also how many I/O operations per second (IOPS) you can perform on the storage – each GB of storage provides three IOPS. You can provision each file system to be up to 64 TB in size.

Bandwidth and Latency

When tuning for performance, it is important to remember the difference between latency and bandwidth. You should find a balance between network latency and availability. Using ENA drivers maximizes bandwidth, but it has no effect on latency. Network latency changes in direct correlation with the distance between interconnecting nodes. Clustering nodes is a way to increase availability, but placing cluster nodes too close to each other increases the probability of simultaneous failure, therefore, reducing availability. Putting them too far apart yields the highest availability, but at the expense of higher latency.

AWS Availability Zones within each AWS Region are engineered to provide a balance that fits most practical cases. Each AZ is engineered to be physically separated from other Availability Zone, while keeping in close geographic proximity to provide low network latency. Therefore, in the overwhelming number of cases, the best practice is to spread cluster nodes across multiple Availability Zone.

Read Replicas

You might determine that many of your DB transactions are read-only queries, and that the sheer number of incoming connections is flooding your database. Read replicas are a known solution for this situation. You can offload your read-only transactions from your primary SQL Server instance to one or more read replica instances. Read replicas can also be used to perform backup operations, relieving primary instance from performance hits during backup windows. When using availability group listeners, if you mark your connection strings as read-only, SQL Server routes incoming connections to any available read replicas and only sends read/write transactions to the primary instance.

Always On Availability Groups, introduced in SQL Server 2012, supports up to four secondary replicas. In more recent versions of SQL Server (that is, 2014, 2016, 2017, and 2019), Always On Availability Groups support one set of primary databases and one to eight sets of corresponding secondary databases.

There might be cases where you have users or applications connecting to your databases from geographically dispersed locations. If latency is a concern, you can locate read replicas close to your users and applications.

When you use a secondary database for read-only transactions, you must ensure that the server software is properly licensed.

Security Optimization

Cloud security at AWS is the highest priority, and there are many AWS security features available to you. These features can be combined with the built-in security features of Microsoft SQL Server to satisfy even the most stringent requirements and expectations.

Amazon VPC

There are many features in Amazon Virtual Private Cloud (Amazon VPC) that can help you secure your data in transit. You can use security groups to restrict access to your EC2 instances and only allow certain endpoints and protocols. You can also use network access control lists to deny known sources of threats.

A best practice is to deploy your SQL Server instances in private subnets inside a VPC, and only allow access to the internet through a VPC NAT gateway, or a custom NAT instance.

Encryption at Rest

If you are using EBS volumes to store your SQL Server database files, you have the option to enable block-level encryption. Amazon EBS transparently handles encryption and decryption for you. This is available through a simple check box, with no further action necessary. Amazon FSx for Windows File Server also includes built-in encryption at rest. Both EBS and Amazon FSx are integrated with AWS Key Management Service (AWS KMS) for managing encryption keys. This means, through AWS KMS, you can either use keys provided by AWS, or bring your own keys. For more information, visit the AWS KMS documentation.

At the database level, you can use SQL Server Transparent Data Encryption (TDE), a feature available in Microsoft SQL Server that provides transparent encryption of your data at rest. TDE is available on Amazon RDS for SQL Server, and you can also enable it on your SQL Server workloads on EC2 instances.

Previously, TDE was only available on SQL Server Enterprise Edition. However, SQL Server 2019 has also made it available on Standard Edition. If you want to have encryption-at-rest for your database files on Standard Edition on an earlier version of SQL Server, you can use EBS encryption instead.

It is important to understand the tradeoffs and differences between EBS encryption and TDE. EBS encryption is done at the block level, that is, data is encrypted when it is stored and decrypted when it is retrieved. However, with TDE, the encryption is done at the file level, that is, database files are encrypted and can only be decrypted using the corresponding certificate.

For example, this means that if you use EBS encryption without TDE and copy your database data or log files from your EC2 instance to an S3 bucket that does not have encryption enabled, the files will not be encrypted. Furthermore, if someone gains access to your EC2 instance, database files will be exposed instantly.

However, there is no performance penalty when using EBS encryption, whereas enabling TDE adds additional drag on your server resources.

Encryption in Transit

As a best practice, you can enable encryption in transit for your SQL Server workloads using the SSL/TLS protocol. Microsoft SQL Server supports encrypted connections, and SQL Server workloads in AWS are no exception. Furthermore, when using SMB protocol for SQL Server storage layer, Amazon FSx automatically encrypts all data in transit using SMB encryption as you access your file system without the need for you to modify SQL Server or other applications' configurations.

Encryption in Use

Microsoft SQL Server offers Always Encrypted to protect sensitive data using client certificates, therefore providing separation provides a separation between those who own the data and can view it, and those who manage the data but should have no access. This feature is also available on both Amazon RDS for SQL Server, as well as SQL Server workloads on Amazon EC2.

AWS Key Management Service (AWS KMS)

AWS KMS is a fully managed service to create and store encryption keys. You can use KMS-generated keys or bring your own keys. In either case, keys never leave AWS KMS and are protected from any unauthorized access. You can use KMS keys to encrypt your SQL Server backup files when you store them on Amazon S3, Amazon S3 Glacier, or any other storage service. Amazon EBS encryption also integrates with AWS KMS.

Security Patches

One of the common security requirements is the regular deployment of security patches and updates. In AWS, you can use AWS Systems Manager Patch Manager to automate this process. Note that use cases for Patch Manager are not restricted to security patches. For more details, refer to the patch management discussion under operational excellence topic in this whitepaper.

Cost Optimization

SQL Server can be hosted on AWS through License Included (LI), as well as Bring Your Own License (BYOL) licensing models. With LI, you run SQL Server on AWS and pay for the licenses as a component of your AWS hourly usage bill. The advantage of this model is that you do not need to have any long-term commitments and can stop using the product at any time and stop paying for its usage.

However, many businesses already have considerable investments in SQL Server licenses and might want to reuse their existing licenses on AWS. This is possible using BYOL:

- If you have Software Assurance (SA), one of its benefits is the Microsoft License Mobility through Software Assurance program. This program allows you to use your licenses on server instances running anywhere, including on Amazon EC2 instances.
- If you do not have SA, you may still be able to use your own licenses on AWS using Amazon EC2 Dedicated Hosts. For more details, consult the licensing section of the FAQ for Microsoft workloads on AWS to ensure license compliance.

The BYOL option on EC2 Dedicated Hosts can significantly reduce costs, as the number of physical cores on an EC2 host is about half of the total number of vCPU available on that host. However, one of the common challenges with this option is the difficulty of tracking license usage and compliance. AWS License Manager helps you solve this problem by tracking license usage, and optionally enforcing license compliance, based on your defined license terms and conditions. AWS License Manager is available to AWS customers at no additional cost.

Amazon EC2 CPU Optimization

The z1d instance types provide the maximum CPU power, allowing you to reduce the number of CPU cores for compute-intensive SQL Server deployments. However, your SQL Server deployments might not be compute-intensive and require an EC2 instance type that provides intensity on other resources, such as memory or storage. Because EC2 instance types that provide these resources are also providing a fixed number of cores that might be more than your requirement, the end result can be a higher licensing cost for cores that are not used at all. Fortunately AWS offers a solution for these situations. You can use EC2 CPU optimization to reduce the number of cores available to an EC2 instance and avoid unnecessary licensing costs.

Switch to SQL Server Standard Edition

Enterprise-grade features of SQL Server are exclusively available in the Enterprise edition. However, over time many of these features have also been made available in the Standard edition, therefore enabling you to switch to Standard edition, if you’ve been using Enterprise edition only for those features. One example is encryption at rest using Transparent Data Encryption (TDE), which is now available in Standard edition as of SQL Server 2019.

One of the most common reasons for using Enterprise edition has always been its mission-critical HA capabilities. However, there are alternative options that enable switching to Standard edition without degrading availability. You can use these options to cost-optimize your SQL Server deployments.

One option is using Always On Basic Availability Groups. This option is similar to Always On Availability Groups, but comes with a number of limitations. The most important limitation is that you can have only one database in a basic availability group. This limitation rules this option out for applications that rely on multiple databases.

The other option is using Always On Failover Cluster Instance (FCI). Since FCI provides HA at the instance level, it does not matter how many databases are hosted on your SQL Server instance. Traditionally this option was restricted to HA within a single data center. However, as discussed earlier, you can use Amazon FSx for Windows File Server to overcome that limitation. See the high availability and disaster recovery section of this document.

You can simplify the complexity and cost of running Microsoft SQL FCI deployments using Amazon FSx in the following scenarios:

- Due to the complexity and cost of implementing a shared storage solution for FCI, you might have opted to use availability groups and SQL Server Enterprise Edition. However, now you can switch to Standard edition and significantly reduce your licensing costs, and also simplify the overall complexity of your SQL deployment and ongoing management.
- You might already be using SQL Server FCI with shared storage using a third-party storage replication software solution. That implies that you needed to purchase a license for the storage replication solution, and then deploy, administer, and maintain the shared storage solution yourself. You can now switch to using a fully managed shared storage solution with Amazon FSx, and thus simplify and reduce costs of your SQL Server FCI deployment.
- If you were running your SQL Server Always On deployment on-premises, using a combination of FCI and AG – FCI to provide high availability within your primary data center site, and AG providing a disaster recovery solution across sites. The combination of Availability Zones and the support in Amazon FSx for highly available shared storage deployed across multiple Availability Zones, now makes it possible for you to eliminate the need for separate HA and DR solutions, and thus reduce costs as well as simplify deployment complexities.

For a more detailed discussion of Microsoft SQL Server FCI deployments using Amazon FSx, visit the blog post Simplify your Microsoft SQL Server high availability deployments using Amazon FSx for Windows File Server.

z1d EC2 Instance Type

The high-performance z1d instance is optimized for workloads that carry high licensing costs, such as Microsoft SQL Server and Oracle databases. The z1d instance is built with a custom Intel Xeon Scalable Processor that delivers a sustained all core Turbo frequency of up to 4.0 GHz, which is significantly faster than other instances. For workloads that need faster sequential processing, you can run fewer cores with a z1d instance and get the same or better performance as running other instances with more cores.

For example, moving an SQL Server Enterprise workload running on an r4.4xlarge to a z1d.3xlarge can deliver up to 24% in savings because of licensing fewer cores. The same savings are realized when moving a workload from an r4.16xlarge to a z1d.12xlarge as it is the same 4:3 ratio.

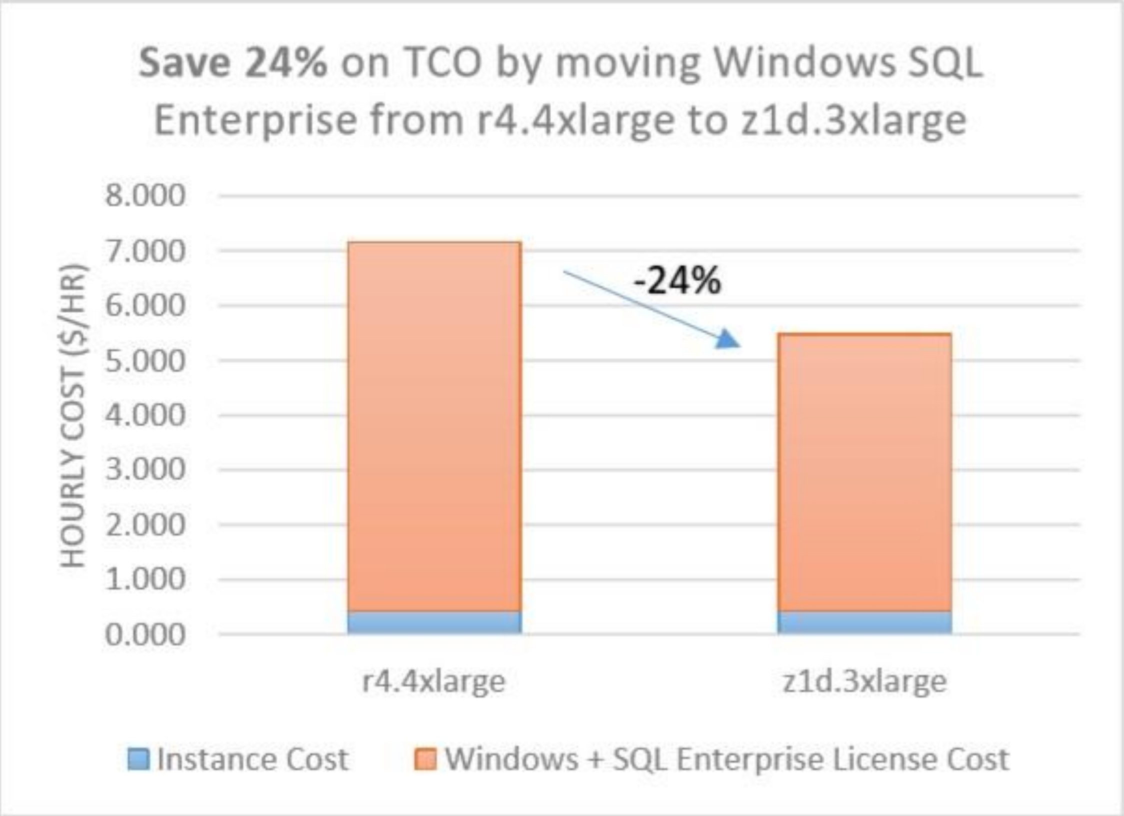

Figure 4: TCO Comparison Between SQL Server on r4.4xlarge and z1d.3xlarge

Eliminating Active Replica Licenses

Another opportunity for cost optimization in the cloud is through applying a combination of BYOL and LI models. A common use case is SQL Server Always On Availability Groups with active replicas. Active replicas are used primarily for:

- Reporting
- Backup
- OLAP Batch jobs
- HA

Out of the above four operations, the first three are often performed intermittently. This means that you would not need an instance continuously up and dedicated to running those operations. In a traditional on-premises environment, you would have to create an active replica that is continuously synchronized with the primary instance. This means you need to obtain an additional license for the active replica.

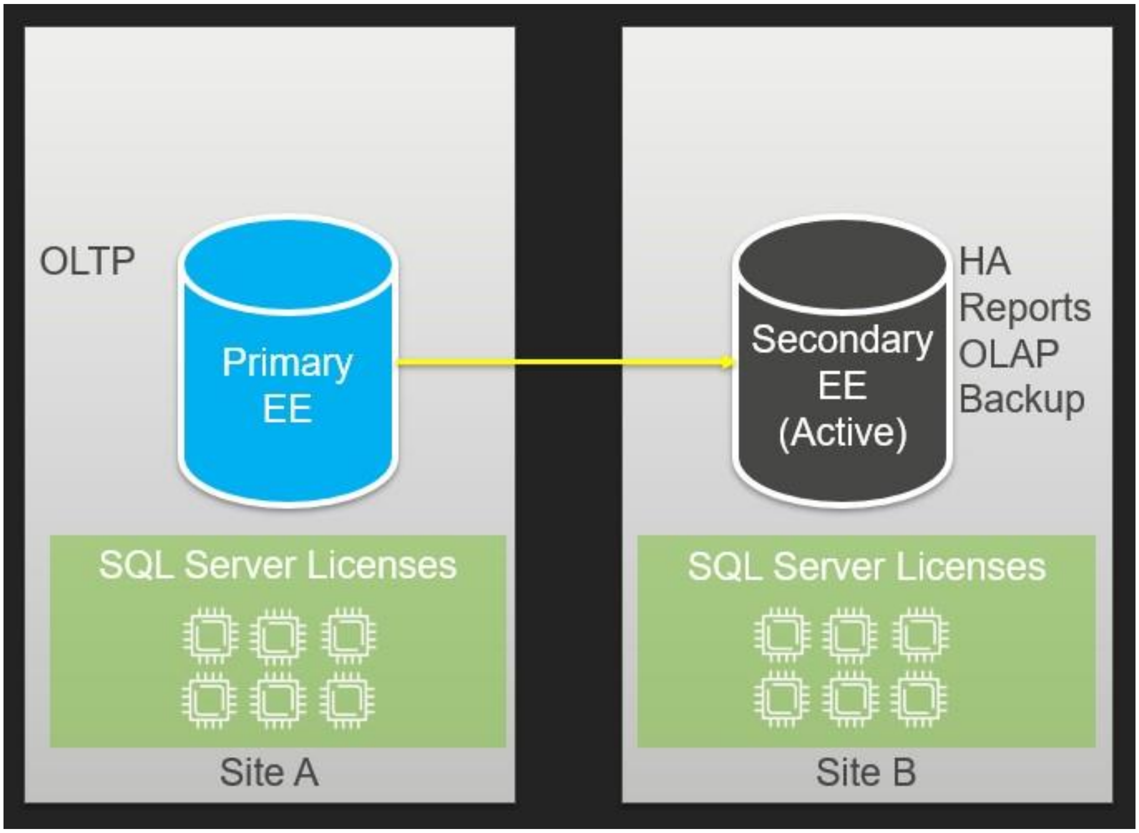

Figure 5: SQL Server Active Replication On Premises

In AWS, there is an opportunity to optimize this architecture by replacing the active replica with a passive replica, therefore relegating its role solely for the purpose of HA. Other operations can be performed on a separate instance using License Included which could run for a few hours and then be shut down or terminated. The data can be restored through an EBS snapshot of the primary instance. This snapshot can be taken using VSS-enabled EBS snapshots, thus ensuring no performance impact or downtime on the primary.

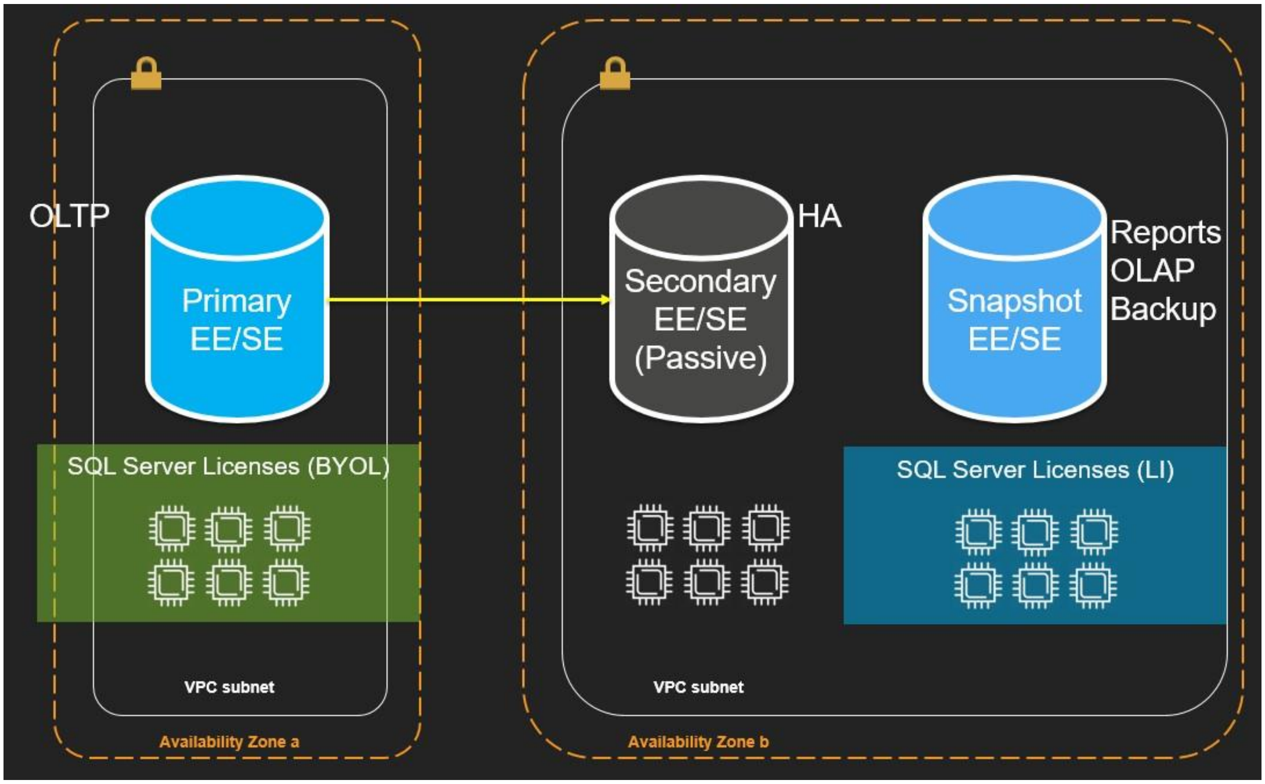

Figure 6: Eliminating Active Replica Licenses in AWS

This solution is applicable when jobs on the active replica run with a low frequency. However, if you need a replica for jobs that run continuously or at a high frequency, consider using [AWS Database Migration Service](#) (AWS DMS) to continuously replicate data from your primary instance into a secondary. The primary benefit of this method is because you can do it using SQL Server Standard edition, it avoids the high cost of SQL Server Enterprise edition licensing.

Refer to the [AWS Microsoft licensing page](#) for more details on ways you can optimize licensing costs on AWS.

Operational Excellence

Most of the discussions explored so far in this whitepaper pertain to the best practices available at deployment time of Microsoft SQL Server in AWS. However, another crucial dimension is operating and maintaining these workloads post-deployment.

As a general principle, the best practice is to assume that failures and incidents happen all the time. It is important to be prepared and equipped to respond to these incidents. This objective is composed of three subobjectives, namely:

- Observe and detect anomaly
- Detect the root cause
- Act to resolve the problem

AWS provides tools and services for each of these purposes.

Observability and Root Cause Analysis

Amazon CloudWatch is a service that enables real-time monitoring of AWS resources and other applications. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications.

Amazon CloudWatch Application Insights for .NET and SQL Server is a feature of Amazon CloudWatch that is designed to enable operational excellence for Microsoft SQL Server and .NET applications. Once enabled, it identifies and sets up key metrics and logs across your application resources and technology stack. It continuously monitors the metrics and logs to detect anomalies and errors, while using artificial intelligence and machine learning (AI/ML) to correlate detected errors and anomalies. When errors and anomalies are detected, Application Insights generates CloudWatch Events. To aid with troubleshooting, it creates automated dashboards for the detected problems, which include correlated metric anomalies and log errors, along with additional insights to point you to the potential root cause.

Reducing Mean Time to Resolution (MTTR)

The automated dashboards generated by Amazon CloudWatch Application Insights help you to take swift remedial actions to keep your applications healthy and to prevent impact to the end users of your application. It also creates OpsItems so you can resolve problems using AWS Systems Manager OpsCenter.

AWS Systems Manager is a service that enables you to view and control your infrastructure in AWS, on premises, and in other clouds. OpsCenter is a capability of AWS Systems Manager, designed to reduce the mean time to resolution. OpsCenter also provides Systems Manager Automation documents (runbooks) that you can use to fully or partially automate resolution of issues.

Patch Management

AWS Systems Manager Patch Manager is a comprehensive patch management solution, fully integrated with native Windows APIs, and supporting Windows Server and Linux operating systems, as well as Microsoft applications, including Microsoft SQL Server.

Systems Manager Patch Manager integrates with AWS Systems Manager Maintenance Windows, allowing you to define a predictable schedule to prevent potential disruption of business operations.

You can also use AWS Systems Manager Configuration Compliance dashboards to quickly see patch compliance state or other configuration inconsistencies across your fleet.

Conclusion

In this whitepaper, we described a number of best practices for deploying Microsoft SQL Server workloads on AWS. We saw how AWS services can be used to compliment Microsoft SQL Server features to address different requirements.

AWS offers the greatest breadth and depth of services in the cloud, and Amazon EC2 is the most flexible option for deploying Microsoft SQL Server workloads. Each solution and associated trade-offs may be embraced according to particular business requirements.

The five pillars of AWS Well-Architected Framework (that is, reliability, security, performance, cost optimization, and operational excellence) are explored as applicable to SQL Server workloads and AWS services supporting each requirement are introduced.

Contributors

The following individuals and organizations contributed to this document:

- Sepehr Samiei, Solutions Architect, Amazon Web Services

Document Revisions

Date	Description
May 2020	Revisited the whole document to cover new AWS services and capabilities supporting Microsoft SQL Server workloads.
March 2019	Information regarding Total Cost of Ownership (TCO), using z1d instance types and EC2 CPU Optimization added.
September 2018	First publication