



WHITEPAPER

Cloud Migration Guide: SQL Server to AWS

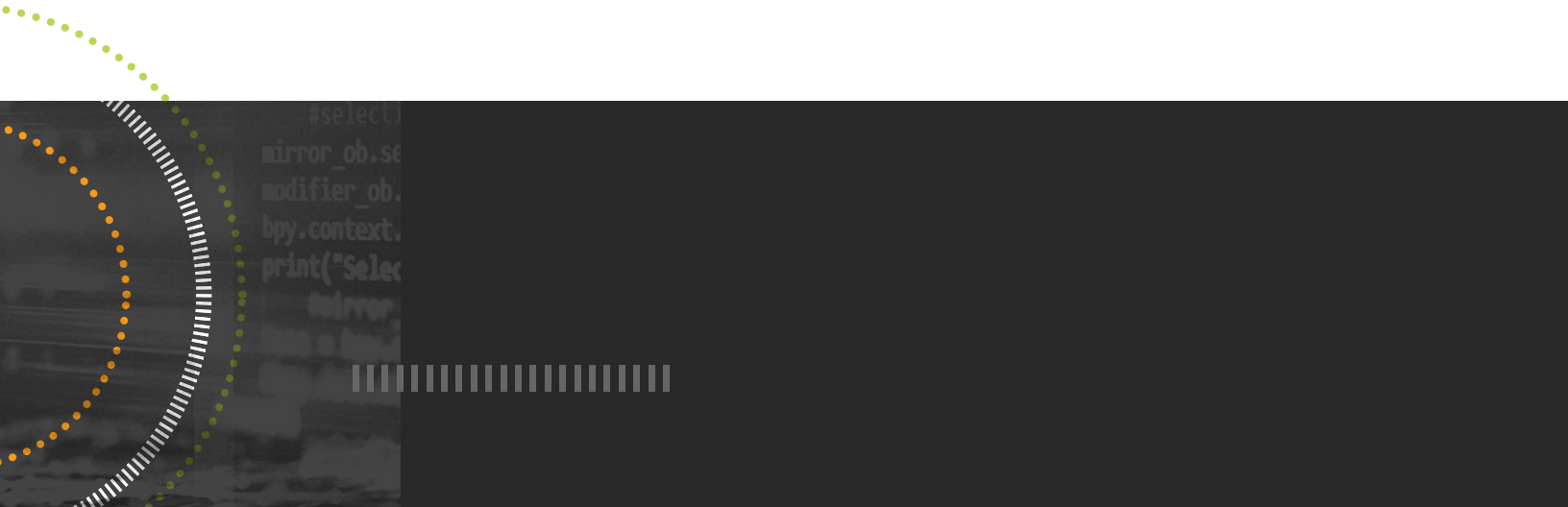


Table of Contents

Introduction	1	Migration to AWS	18
The Migration Process	1	RDS for SQL Server	18
Scope Definition	2	Amazon EC2 VM	20
Planning	2	Migration Summary	22
Phase 1: Discovery	3	Post-Migration Operations	23
Phase 2: Testing and Validation	3	Monitoring	24
Phase 3: Migration	4	Performance Monitoring	24
Phase 4: Post-Migration Operational Support	4	Data Integrity Monitoring	24
Example Migration Scenario and Objective	4	Maintenance Activities	25
Discovery	5	Index and Stats Maintenance	25
Scope Definition	5	Business Continuity Testing	25
Documenting the Current System	5	Summary	26
Understanding the Business	7	Disclaimer	26
System Performance Baselines	7		
Discovery Summary	8		
Analysis and Testing	9		
Availability and Up-Time	10		
Identifying the Migration Target	10		
Sizing the Cloud Systems	10		
Amazon RDS for SQL Server and EC2	10		
Adopting a Cloud Mindset	11		
Analyzing Workloads	11		
Workload Replay Options	12		
Workload Replay Process	13		
Monitoring for Workload Analysis	14		
Validating the Data	15		
Platform Capability Validation	15		
High Availability Configurations	16		
Disaster Recovery Scenarios	16		
Migration Dry Run	17		
Analysis and Validation Summary	17		

Introduction

The Microsoft Data Platform technology landscape is rapidly advancing, with increased release cadence and new capabilities regularly added. Businesses need to evolve their data platform strategies—migrating to the cloud is a key pillar of this strategy. Amazon Web Services (AWS) can deliver greater agility and operational efficiencies when running SQL Server in the cloud.

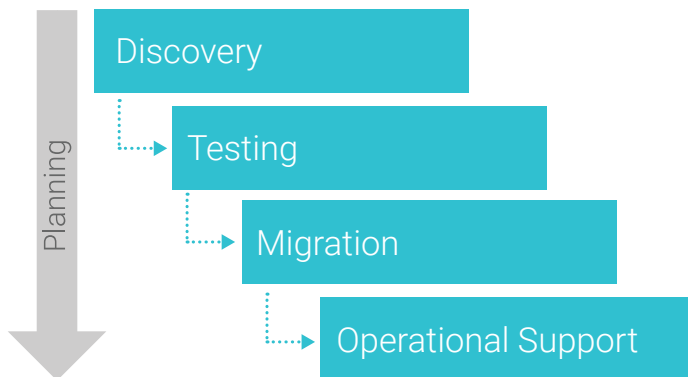
However, transitioning from on-premises to a cloud or hybrid cloud data platform can be difficult without the right solutions to help discover, document, and validate migration activities. Whether moving to Infrastructure as a Service (IaaS) or Platform as a Service (PaaS), SolarWinds provides a couple of solutions designed to reduce the risks and time taken to perform cloud migrations to AWS.

Given the increased cadence of SQL Server releases, many of the migration activities detailed in this guide also apply when performing SQL Server upgrades on-premises if you aren't yet ready for full cloud adoption. This guide will provide a detailed look at the migration process and options and show you how [SolarWinds® SQL Sentry](#) and [SolarWinds Database Mapper](#) can assist with the upgrade and migration of SQL Server platforms to the Amazon Cloud.

The Migration Process

Migrating SQL Server workloads to AWS doesn't have to be risky, difficult, or expensive. With the correct approach, you can avoid many of the common pitfalls associated with the migration process, and those that can't be mitigated can be identified to allow for a thorough assessment.

Although many project management frameworks would work well for a migration, it's important to identify the framework that works best for your organization. The framework should address discovery, testing/validation, migration, and operational support. Spanning all these stages is planning.



The four stages of the waterfall project management method

SCOPE DEFINITION

It's possible to audit an entire environment and look for all SQL Server systems when determining the scope of your migration. However, to successfully migrate workloads, it's best to restrict the initial project scope to a specific business system or department.

The initial scope will feed into the basic plan and, subsequently, the discovery phase. As the migration progresses through the discovery phase, the scope can be refined based on the data collected. Likewise, as the testing phase occurs, the scope will likely be refined depending on if it's viable to migrate workloads as-is or if more work needs to be completed before a workload can be migrated to AWS.

You also want to consider how your business might evolve during the migration process. If standards or processes change, then it's important to understand if the migration project needs to reflect the changes, too.

PLANNING

When developing a plan for conducting a migration, you and your business leaders must understand it's an iterative process. As the migration proceeds, new facts and data will become available that can impact the state of the plan. Addressing these changes is crucial—trying to force the migration to adhere to the original plan without accounting for feedback will result in failure.

A combination of waterfall and Kanban project management methods works well for managing migration activities, especially if your operations staff is managing the process. Should the migration be a larger, more formal project, then there could be a case for a combination of waterfall and scrum.

The waterfall method is used to control the phases rather than the underlying order in which the discrete activities take place. The overall process for a cloud migration would be logically broken down, so you complete the discovery phase before the analysis phase, for example. Within each of these phases, the Kanban method allows your teams to tailor the approach to each system and complete the appropriate tasks.



Refining your cloud migration scope to reflect changing conditions must happen during the migration process. However, it's important to avoid scope creep that adds extra work without a compelling reason supported by data.

PHASE 1: DISCOVERY

An in-depth understanding of your primary data systems will help you build a detailed migration plan. Key objectives for discovery include:

1. Systems from which the migration target consumes data
2. Systems for which the migration target produces data
3. Current versions of software components
4. Resource utilization (e.g., CPU, storage, memory)
5. Performance baselines
6. Sample workloads
7. Recovery Point Objectives (RPO) and Recovery Time Objectives (RTO)
8. Application owners

You can collect some of this information using automated tools such as monitoring or documentation software. However, gathering other crucial information will require you to engage your business leaders, which has the added benefit of ensuring buy-in for the migration. Buy-in ensures when you need to make requests from your business leaders, they're more inclined to collaborate with you.

PHASE 2: TESTING AND VALIDATION

Once you have a clear understanding of the data within the scope of your migration, you need to know how it will behave on your new platform. This is an ideal time to look at the workload on the new system to identify regressions. Although performance degradation is important to identify, you should also document performance improvements, as improvements are also a change in workload behavior.

During this stage, you should begin evaluating migration options and testing the options you think might meet your needs. Although experienced staff will have preferences for how to move databases and other objects, it's important to keep an open mind about your migration options.

Finally, you'll need to define and set up the tests that will validate the migration. It's vital for decision points to be defined, and the information needed to make decisions about proceeding or rolling back is available well ahead of the migration.

When possible, automation should be at the forefront of your testing processes. By automating tests, you can ensure the output between test runs can be compared and any differences will be valid. If you manually perform testing, you run the risk of someone not following the process correctly, which invalidates the results, leading to additional work and risk being added to the migration process.

PHASE 3: MIGRATION

The actual migration, although critical to the process, should be one of the least stressful steps. With effective discovery and testing completed, there should be no surprises from areas you control. By documenting a clearly defined timeline and decision gates, you can also provide your business leaders with confidence the migration process will have minimal risk.

PHASE 4: POST-MIGRATION OPERATIONAL SUPPORT

Often, once the migration has been completed, many teams treat the new platform as business as usual (BAU). However, your team must keep a close eye on the system to ensure any issues not detected during the pre-migration testing and discovery phases are identified and addressed immediately. During this phase, you'll also need to update your documentation based on the data gathered during the migration.

Example Migration Scenario and Objective

Let's look at an example scenario in which you have a range of SQL Server versions and technologies in play and explore your options for migrating to a fully supported data platform based in AWS.

For the purposes of this guide, this example scenario assumes you're leveraging SolarWinds database management solutions to support your migration effort. The following SolarWinds solutions will be referenced throughout the process:

- [SQL Sentry](#)
- [Database Mapper](#)

Using these solutions, you can speed up the migration process by automating activities and provide more detailed data capture and a consolidated location for team members to store and obtain information about your database environment.

Discovery

Once you've identified the systems involved in the migration process, you need to perform a comprehensive discovery on them to gather the data needed to make decisions about how best to perform the migration.

SCOPE DEFINITION

In this example scenario, you're working with a clearly defined scope and the data platform has OLTP, ETL, and reporting layers. However, you should perform effective due diligence to ensure you haven't omitted any necessary systems or data and there aren't any departmental applications in use that you aren't aware of. It's both a discovery exercise and a validation of the initial scope.

DOCUMENTING THE CURRENT SYSTEM

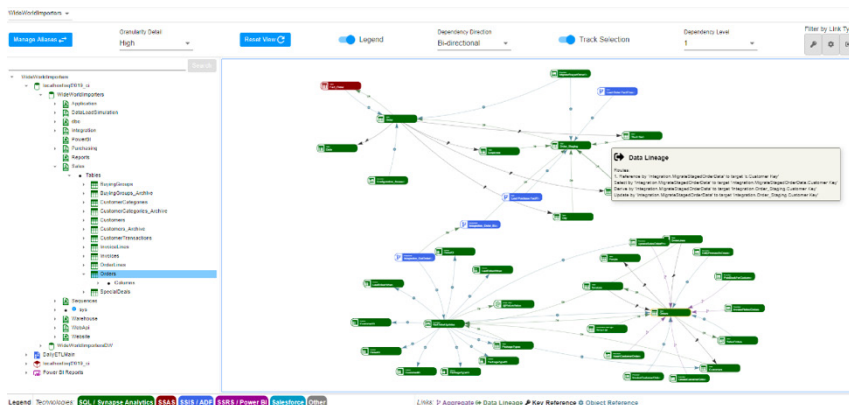
Typically, documenting a collection of SQL Server systems and producing a cohesive library of information can take some time. You can speed up this process with [SolarWinds Database Mapper](#).

In Database Mapper, you can create a solution containing the database, ETL, and reporting servers you want to analyze. This information is then stored in a SQL Server database and can be accessed via the Database Mapper interface to allow for effective collaboration among team members.

After you've completed a scan of your systems, you can then review all the configuration data about the servers and the databases they contain. Collecting this data allows you to build a data map of your environment. The data lineage functionality enables you to perform detailed impact analysis of any changes you'll make as part of the migration process. Visibility into the data flowing through each server, database, and SSIS package you'll be migrating allows you to understand the upstream and downstream impact of changes. You can also document and analyze the risks to the business for the migration activity.



For more information about what data lineage is and why it's important to identify dependencies, check out our on-demand webinar **"Data Lineage and Documentation: How Changes Affect Your Environment."**

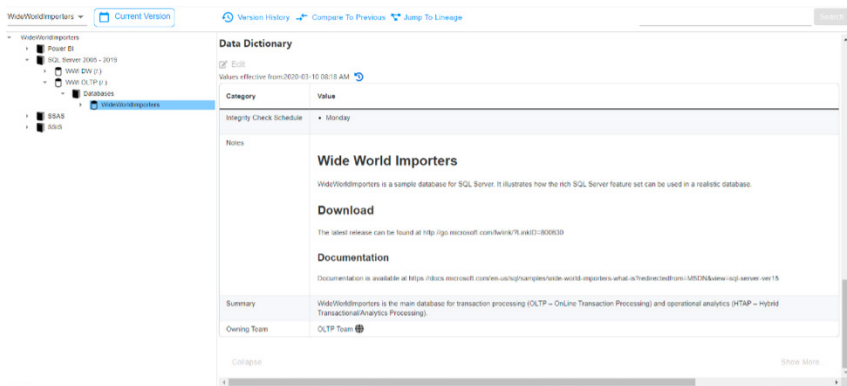


Database Mapper data lineage—from source table to SQL Server Reporting Services (SSRS) report items

In addition to collecting hard data related to servers, databases, packages, jobs, etc., you need to collect and catalog the metadata, which will be crucial to prioritizing activities, as well as ensuring the data and components are handled appropriately. By leveraging the data dictionary capabilities in Database Mapper, in combination with the data captured, you can easily tag sensitive data fields for PCI DSS, personally identifiable information (PII), General Data Protection Regulation (GDPR), HIPAA, or other compliance requirements.

Additionally, you can include extra metadata about application and database owners, points of contact, and information about what these entities are used for. The ability to add information about what an ETL job does, and more importantly why it does what it does, is key to understanding if the job is working as planned. A great use of the data dictionary is to add information around test criteria and what is a pass/fail. This information can then be used not only to build tests to validate the migration but also for post-migration BAU activities to support the continued development of your platform.

The work you do at this stage to capture the data and turn it into actionable information will live on long past the migration process. Your future self will thank you for putting in the effort during this stage.



Add business-specific and compliance metadata to data entities in the Database Mapper data catalog

UNDERSTANDING THE BUSINESS

Another area of the discovery phase is understanding the business aspects of the migration process. You must consider what data is contained within the systems and databases, how important the data being migrated is to the business operation, and who's impacted by the migration. As part of the discovery process, you need to collect information about SLAs for performance, availability, and disaster recovery (DR) scenarios, including Recovery Point Objectives (RPO) and Recovery Time Objectives (RTO), as this information will drive the selection of the appropriate cloud services.

WHAT DATA ARE YOU MIGRATING?	WHO WILL BE IMPACTED BY THE MIGRATION?
<p>Understanding the classification of the data being migrated is important. In the case of PII or healthcare data, compliance requirements need to be met. It's also advisable to classify and document data entities as early as possible to ensure appropriate data handling.</p> <p>This information will be used to understand which geo-political region the data needs to be constrained within, as well as security and compliance requirements.</p>	<p>To ensure your migration is successful, you need to identify the application and data owners, so you can get buy-in and effective communication can take place.</p> <p>It's also important to identify where users are based and their active hours, so you can assess the impact of the migration and plan for any outages. You should also understand who the key decision makers are when it comes to validating the new platform and making migration go/no-go decisions.</p>

The information you capture while speaking with your business leaders needs to be documented. Although this information can be added to your SQL Server systems via Extended Properties, annotations, or in a wiki, it's more effective to add it to Database Mapper. By using the data dictionary functionality, you can add a wealth of metadata to the data captured by the solution. Capturing this information in Database Mapper has the added benefit of allowing business users to manage this metadata to ensure it's kept up to date.

SYSTEM PERFORMANCE BASELINES

When performing an upgrade or migration, performance is a key metric the business will measure. Often, there's reluctance to adopt modern technology because of a perceived risk associated with being at the bleeding edge or even as a fast follower. However, you can establish the actual performance impact by providing greater visibility into performance and reliability by establishing a baseline. In many cases, you can leverage modern technologies to improve performance or get the same level of performance with less resources. However, it's crucial to have a solid monitoring solution implemented to capture baseline data, which can be referenced during the testing and post-migration phases.

When baselining performance for a system prior to an upgrade or migration, it's important to identify the most important metrics to your business. Taking a blanket approach can result in too much data collected and important data points being lost in the noise. This is where [SolarWinds SQL Sentry](#) comes in, as it delivers built-in baselining capabilities and can collect all the key metrics for migration analysis.

When performing a cloud migration, the following elements can affect performance:

- **CPU utilization**—When analyzing CPU resources, it’s important to take parallelism into account. Often, you’ll hear about people trying to drive higher CPU usage by using fewer cores for a workload. Although this can be achieved, it’s important to assess the impact on large queries that use multiple threads.
- **Disk I/O**—Not all cloud storage is created equal. Amazon has two major types of storage underpinning the database offerings: Standard SSD and Provisioned IOPS, which will provide a guaranteed level of performance, something lacking in many clouds.
- **Memory utilization**—Memory in the cloud is tightly controlled. With IaaS solutions, you only get predefined configurations. With PaaS solutions, memory is abstracted away and not something you can affect directly. Knowing your memory utilization and associated PLE/storage throughput will help inform sizing decisions.
- **Query performance history**—In recent versions of SQL Server and in the PaaS solutions, Microsoft has updated parts of the optimizer. This can result in behavioral changes both, good and bad. Understanding query-level activity helps identify regressions that need to be fixed as well as performance improvements experienced from moving to a new system.

When capturing baselines, you must ensure you have all the major time periods for your business covered, including key active hours for the business and any maintenance windows. (You must take maintenance periods into account because they’re still required in the cloud. There’s a misconception that these PaaS cloud solutions are entirely managed, but they aren’t.)

DISCOVERY SUMMARY

Once you’ve collected all the data, you have most of the information you need to move onto the testing and validation phase of the migration process. It’s important to understand this data needs to be kept up to date. A common reason migration activities fail is because the project happens in isolation and the business continues moving. It’s important to regularly renew the data captured to look for new databases and data feeds. Additionally, if new users are added to a system, the performance profile of the database systems can be altered.

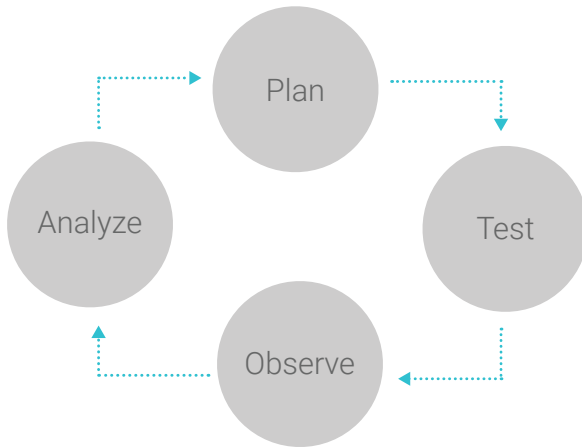
All documentation created throughout the discovery phase should be considered living documents and maintained as such. Finding out about changes early is key to ensuring they feed into the plan and result in a successful migration.



For more information about how SQL Sentry can be used to collect performance baselines, check out the on-demand webinar “[Setting SQL Server Performance Baselines and Alerts](#),” from Kevin Kline and Richard Douglas.

Analysis and Testing

With documentation and performance baseline data in hand, you can refine the plan and define the migration activities you need to undertake. This phase of the migration is broken down into three work streams, which are fed back into one another as you select the services and mechanisms you'll use to migrate your databases.



Cloud migration phases

Speaking to your business leaders will provide greater insight into their needs from the platform regarding service uptime, high availability, DR, and allowable downtime for the migration. Then, you can evaluate which cloud services best align with those needs.

AVAILABILITY AND UPTIME

Amazon has several SLAs for its cloud services, as shown in the table below. However, in the case of IaaS virtual machines (VMs), there are certain configuration requirements you must adhere to for the SLAs apply. These requirements will have an impact on the way you design your cloud-based infrastructure. If your business needs higher uptime for your systems, then more complex configurations will be required to deliver it.

AWS CLOUD SERVICE	REQUIREMENT	UPTIME SLA
EC2 Compute	IaaS EC2 VM configured in a single AWS region	99.99%*
RDS	Multi-AZ instances	99.95%**

* <https://aws.amazon.com/rds/sla/>
 ** <https://aws.amazon.com/compute/sla/>

IDENTIFYING THE MIGRATION TARGET

As we discussed earlier, there are several different options when it comes to moving your databases to the cloud. One of the key differences between the AWS and Microsoft Azure offerings is that Amazon uses on-premises SQL Server to underpin both the EC2 IaaS and RDS for SQL Server PaaS offerings. The upshot of this is you have multiple SQL Server versions available, much as you would on-premises. This also means there are far fewer blockers to migrating to PaaS, as it's just regular SQL Server albeit with a few limitations around the implementation.

SIZING THE CLOUD SYSTEMS

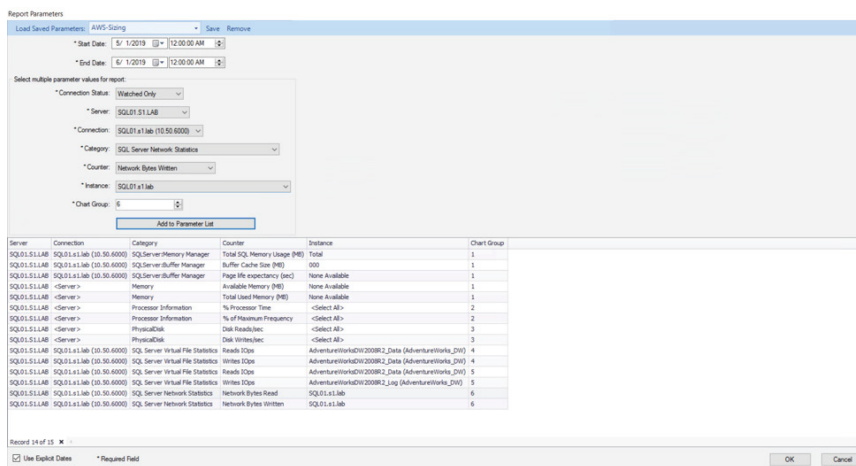
Sizing EC2 VMs and RDS for SQL Server instances is relatively intuitive. Both options work on the premise of vCPU, memory, IOPS/storage, and network throughput. As with many cloud vendors, there are pre-sized instance options to pick from for different combinations of resources.

AMAZON RDS FOR SQL SERVER AND EC2

AWS has an array of instance sizes available for use with SQL Server in RDS. However, the images you can use will depend on the version and edition of SQL Server you want to deploy. More information on the images available for each version can be found in the [RDS user guide](#).

Once you've identified the images you can use with the version of SQL Server you want to deploy, you need to pick which image you want to deploy. When picking the appropriate image size, it's important to understand the resources available to each, details of which can be found in the [RDS user guide](#).

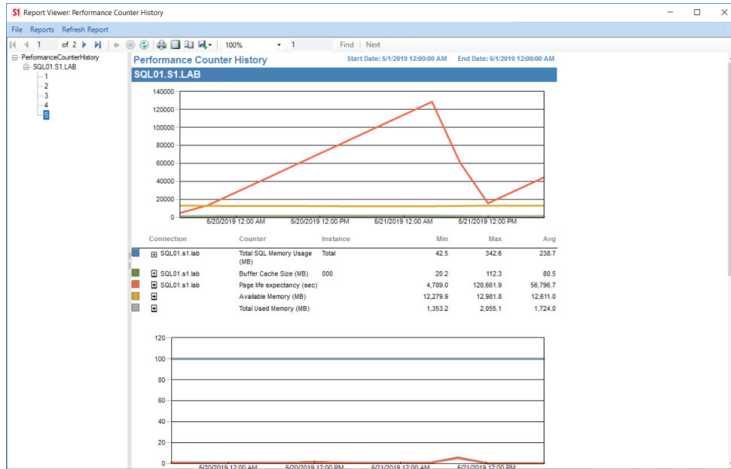
Because of the model used by RDS for SQL Server to allocate CPU, memory, and IOPS, it's relatively simple to model what you currently have to the new environment. This is where using the Performance Counter History report in SQL Sentry can help in the planning phase.



SQL Sentry performance counter history report configuration

By grouping the key performance counters into logical groups around CPU, memory, IO, and networking, you can see what your historical performance baselines are. This historical information allows us to select the appropriate size AWS instance you need. The upshot of this approach is you can select a machine size that can run your workload but isn't oversized, saving you from unnecessary costs.

This grouping of performance counters means you have a very clear report (example shown below) focused on what you're interested in.



SQL Sentry report—performance counter history

ADOPTING A CLOUD MINDSET

When you provision on-premises systems, you do so by estimating the lifespan and factoring in growth over time, which often means you'll over-provision capability for the system just in case. If you take this approach with cloud systems, it will cost more money than you need to spend, ultimately reducing ROI.

When working with cloud-based systems, whether IaaS or PaaS, you want to run with an overall higher resource utilization. Whereas you would see many on-premises systems running with 30% to 40% utilization for CPU, in the cloud, you want to push CPU utilization as high as you can go while allowing for performance spikes. Doing so minimizes additional spend on unused resources. It's easy to plan for scaling up a VM from one class to another, and it's easy with PaaS to move from one service tier to the next. Although these operations are typically offline events, with effective planning, you can minimize the impact to end users.

ANALYZING WORKLOADS

A crucial part of the testing and validation phase is ensuring when databases are moved to cloud-based systems, they perform as expected. This analysis isn't limited to hard performance counter metrics such as transactions per second, IO throughput, and response times. It also includes confirming queries return the correct results, and the correct data is being written to your databases.

Workload analysis should be completed in several iterations, starting with the most basic tests, to ensure applications can connect, and simple transactions can be processed. In many cases, you might need to perform an action on the dev/test version of the system on-premises, repeat the action on your new platform, and then compare the results.

Validating data can be a painstaking task—automation should be applied whenever possible.

Writing formal tests in code at this stage will greatly help when performing more in-depth workload testing. Also, automating the data validation process ensures tests are repeatable, you can trust the output, and test runs can be compared to one another. When you manually run tests at scale, small discrepancies can creep in and invalidate the results.

WORKLOAD REPLAY OPTIONS

When performing workload testing, it's best to use a production workload, which allows you to validate the new systems against a known system and production baselines to ensure parity. However, capturing and replaying workloads isn't a simple task. There are several options available to help with this task. These include OStress in the [RML Utilities](#) suite from Microsoft or SQLWorkload in the [WorkloadTools](#) open-source project by Gianluca Sartori ([t|b](#)). These tools have similar functionality but complement one another.

WorkloadTools

WorkloadTools is useful for live streaming a workload from one server to another. If you leverage Extended Events (XE), workloads can be streamed from one server to another with low overhead. It's also possible to perform workload streaming via SQL Trace on earlier versions of SQL Server where there isn't parity between Trace and XE for events. You can also capture a SQL Server workload with SQL Trace and output to Trace files. These files can then be converted to a replay format SQLWorkload can process. This approach lets you capture several different workloads and store them for repeated use throughout the analysis phase as well as a baseline.

OStress

OStress is a replay tool that has been around for a long time and works on both SQL Trace and XE files. The process for capturing and converting a workload is a bit more complex than SQLWorkload. However, OStress includes an additional capability for the replay mode: multi-user simulation for testing concurrency limits. You can achieve this by combining OStress with ORCA in the RML Utilities suite.

The workload is captured using native XE or SQL Trace to files; these files are processed into the Replay Markup Language (RML) using the ReadTrace tool. OStress then reads the RML files to replay the workload against a target database.

OStress can be used in conjunction with the Orca tool in RML Utilities, making it possible to coordinate the replay of the workload over multiple clients. This approach allows you to simulate a multi-user environment to determine if there are potential concurrency issues when the workload is run on the new system.


WORKLOAD REPLAY PROCESS

The most crucial element of the workload replay task is ensuring a consistent database that can be reset on the target systems. You should take a full backup of the source database(s) for the workload you'll capture. Then, create a **marked transaction** right before starting the workload capture, which allows you to take a transaction log backup and restore the database to the marked transaction, after which you can replay the workload.

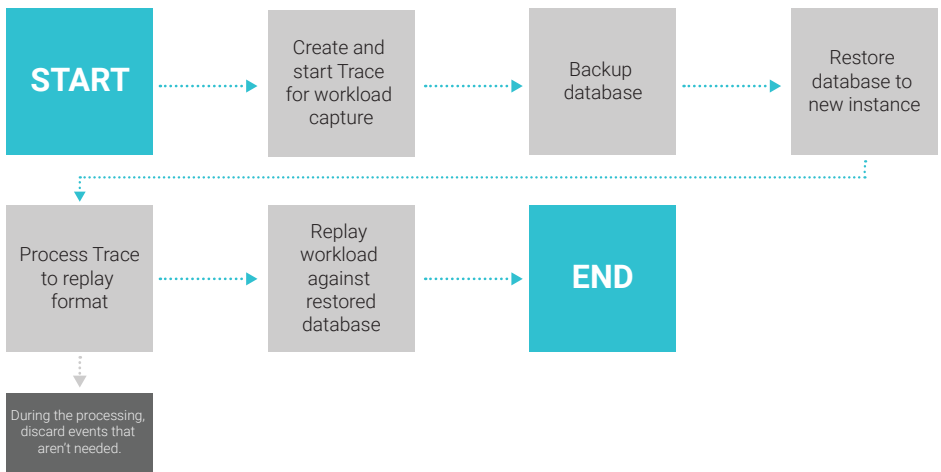
This assumes, however, you're working with a full or bulk-logged recovery model. If you're using simple recovery, it can be more difficult to coordinate the backup with workload capture. In this instance, a potential solution would be to trace the backup as well as the workload, and then manually modify the workload records to remove the ones prior to the completion of the backup. However, this isn't an exact science and can be more difficult to do.

When replaying the workload, the database should be restored, workload replayed, and system monitored. This monitoring session can then be compared to the baseline established during the workload capture or discovery process baseline. This database can be restored, and the workload replayed as needed, based on different configurations for the target system.

The Microsoft Database Experimentation Assistant can help with workload replay. However, there's a licensing overhead for this tool due to the reliance on SQL Server Distributed Replay.



For more information
about how SQL Server backups work, so you can decide which events might need to be discarded from the trace based on the transactions in the backup, see Paul Randal's blog posts "[More on how much transaction log a full backup includes](#)" and "[Debunking a couple of myths around full database backups.](#)"

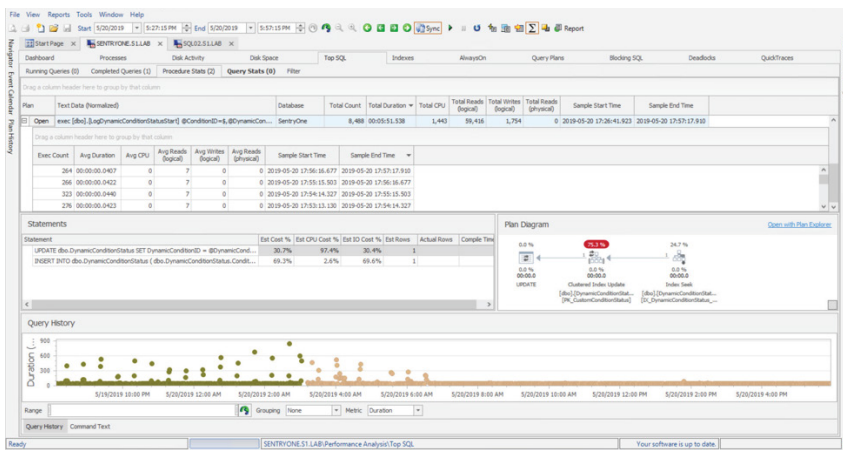


Workload replay process

MONITORING FOR WORKLOAD ANALYSIS

When performing workload analysis, you're likely to move through several iterations as you tune the configuration and make changes to code to work around any issues or incompatibilities.

Monitoring the source system via SQL Sentry allows you to create custom baselines for the periods of interest. The detailed performance metrics captured also enable you to track resource utilization over time for many counters. SQL Sentry's Top SQL functionality also provides deep insight into how queries are executing. You can identify problematic large queries and spot the rapidly executing large volume queries that can cause problems with overall resource usage.



SQL Sentry Top SQL showing procedure execution history with multiple execution plans in history

With this source data, you can monitor the new target systems with SQL Sentry to compare against your baselines. The deep insight into the way the workload is executing and the execution plans in use allow you to identify both areas of improvement due to the latest version of SQL Server in play. Or, more importantly, performance regressions that need to be resolved to ensure end users aren't affected, and system performance isn't compromised.

When migrating from older versions of SQL Server to new versions in AWS, it's important to understand the changes that have taken place in the engine over time. SQL Server 2014 introduced a new cardinality estimator. SQL Server 2016 altered several default behaviors and the optimizer hot fixes were enabled by default. SQL Server 2017 introduced the new adaptive query processing features to optimize queries more efficiently. SQL Server 2019 will add capabilities that can dramatically alter the way scalar UDFs work.

You must understand the potential effect of these changes to how your queries and workloads function as part of any migration effort.

VALIDATING THE DATA

In addition to validating the performance and behavior of workloads on the SQL Server you'll be running, you must ensure data is written to the database correctly and query output is as expected. This complex and repetitive task is best suited to automation rather than human operators.

By clearly defining the tests for a predefined input, you can ensure with every workload run, it's possible to validate the data written to the database and spot any issues. This is important for two reasons:

1. The rate at which bugs are fixed in SQL Server has accelerated dramatically. Behavioral changes introduced with these fixes could manifest in the results you see in your queries. Automated testing for data helps identify these changes before they become a problem.
2. Microsoft has altered the default behavior of the SQL Server engine. For example, in SQL Server 2016, there were changes to the level of accuracy for values returned when there were implicit conversions between numeric and datetime types. Depending on the tolerances in your system, these variances could be something you need to detect and then update code to maintain behavior.

Microsoft is very diligent when it comes to documenting changes and making information available. It's possible to see the breaking changes to SQL Server 2016 database engine features in the [Microsoft documentation](#).

The most fundamental capability for workload validation is the ability to compare two data sets drawn from different databases.

In this scenario, having a static data set that's a snapshot of the database(s) as of the end of the workload capture allows you to then create tests to validate the data contained in the snapshot versus the state of your new database platform after the workload has been replayed. This test will help you understand whether there's a discrepancy in the data written in the new system.

PLATFORM CAPABILITY VALIDATION

Although you must understand the performance characteristics of the workload and data validation, the operational manageability of the cloud platform must meet the business needs you documented during the discovery phase.

Migrating from an on-premises system, where you have a high degree of control over how backups and maintenance operations are undertaken, to a cloud platform, where many of these actions are abstracted or handled by the platform and are therefore outside of your control, requires you to adapt your operational practices.

HIGH AVAILABILITY CONFIGURATIONS

The recommendation from Amazon when deploying either RDS for SQL Server or EC2 VMs is to create a Multi-Availability Zone (AZ) deployment. This essentially means you're deploying to multiple AZ within the Amazon cloud platform, which is the same principal you use in a data center you own by distributing hardware over a wider area to minimize the chance of an outage taking out all members of an Availability Group.

In the case of EC2 machines, you can leverage the Always On Availability Group capabilities within SQL Server to provide high availability. One of the key differences between an on-premises and AWS EC2 deployments is you need to assign multiple IP addresses to the server hosts, one of which will then be used for the listener on that server. More details about how to configure an Availability Group in EC2 can be found in the [AWS documentation](#).

DISASTER RECOVERY SCENARIOS

Currently, RDS for SQL Server doesn't support a multi-region, multi-AZ deployment model. As such, you must ensure you have a robust DR process in place that will allow you to recover if your primary site goes offline. RDS has several options to help with designing a DR solution to meet your needs. The most important pieces of information to have are the RPO and RTO of the databases involved.

RDS, as a PaaS offering, provides an element of built-in backup automation. By default, RDS for SQL Server systems will be backed up with a default retention of one day. You can customize the default retention period based on the needs of your business. These backups allow for a point-in-time restore of the database to be completed if needed over a longer period of time. The default backup schedule is daily full backups with transaction log backups every five minutes.

However, the default automated backups are only available within a single region for restore purposes. If you have a requirement for implementing a multi-region DR scenario, then you should look at using snapshots you can copy to another region. Unfortunately, RDS for SQL Server doesn't support Read Replicas for this purpose.

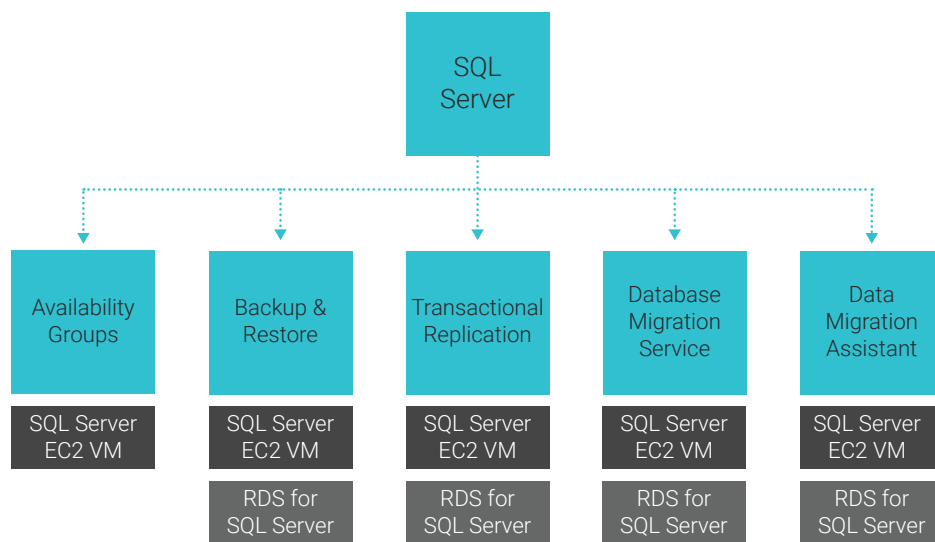
AWS operates what are referred to as Availability Zones (AZ). These distinct implementations don't share physical infrastructure. That way, if an AZ has a power outage, your services can failover to the other AZ they're configured in. Learn more about [Multi-AZ deployments](#).

MIGRATION DRY RUN

Toward the end of the analysis phase, you should be ready to test the methods you'll use to migrate your workload from on-premises to the cloud.

As with anything, there are multiple ways to achieve the objective. However, rather than immediately falling back to tried-and-tested methods you're familiar with, you need to evaluate all the options and make a final decision based on gathered data and facts to ensure the migration methodology is appropriate and meets the needs of the business.

When migrating SQL Server workloads from on-premises to AWS, there are several options, as illustrated in the diagram below.



Migration paths to AWS for SQL Server

ANALYSIS AND VALIDATION SUMMARY

The analysis and validation phase of the migration effort is vital. If you don't complete this activity, there's a higher degree of risk that the migration effort won't succeed. The outputs you should have in place are as follows:

1. Comprehensive documentation
 - Covering the source system
 - Analysis from the tests that justifies the migration method selected
 - Any changes that have been or need to be made for the migration to succeed
 - A detailed migration plan with rollback steps if there's a failure
2. Test automation for validating the success of the migration checkpoints to provide details to the decision makers for continue/rollback quality gates
3. Platform monitoring and analysis in place for source and destination systems

Migration to AWS

Now it's time to review the actual migration process. In addition to the technology aspect, we'll cover how to implement the changes and decision-making process. Let's look at the migration options for moving workloads from on-premises systems to AWS.

RDS FOR SQL SERVER

RDS for SQL Server is aimed at those who want the minimal management overhead of PaaS but with the choice of versions/editions of SQL Server you're familiar with. By offloading management of the underlying OS, patching, backups, and infrastructure requirements reduces the overall management overhead for organizations making use of RDS.

There are several options when it comes to migrating databases to RDS for SQL Server; let's look at three of the most common methods.

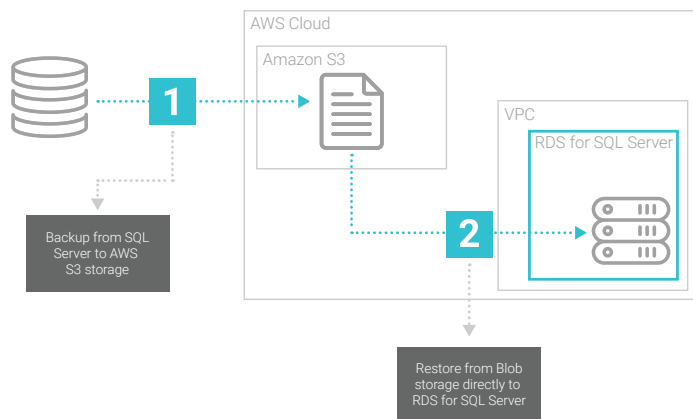
Backup and Restore

RDS for SQL Server supports the backup and restore of native SQL Server backups. This means you can take a backup of an on-premises database and restore it to RDS. To perform this operation, you first need to create an Amazon S3 bucket and present it as a mapped drive in the Windows OS of your source. Details of how to add an S3 bucket to Windows as an SMB mapped drive can be found in the [AWS Storage Gateway](#) user guide.

Once the drive is mapped, then it's simply a case of backing up the SQL Server database as you would typically. However, in this case, you're targeting the mapped drive. Because the mapped drive is in an Amazon region, there will be higher latency and throughput will be constrained by your internet bandwidth.

Once the backup is in the S3 bucket, you need to configure the RDS instance to be able to access the storage to read the backup files. For more information about how to map an S3 bucket to an RDS for SQL Server instance, see the [AWS documentation](#).

One of the key benefits of using RDS for SQL Server is it uses on-premises SQL Server under the covers. So, you can easily take a native backup and then restore it to the same version or higher of SQL Server on-premises. This way, you avoid the potential lock-in of other PaaS SQL Server implementations.



RDS for SQL Server migration—backup and restore

Consideration—Backup to S3 Performance

When taking a backup to Blob storage, the limiting factor will typically be bandwidth on your internet connection. However, if that isn't the case, then striping a backup over multiple files will help drive higher throughput on the network due to multiple writer threads. RDS for SQL Server can also make use of multiple threads when performing the restore operation, which results in higher throughput for the restore operation.

Database Migration Service

The Amazon Database Migration Service (DMS) is a service that will automate a large portion of the process when moving your databases from on-premises to AWS (EC2 or RDS for SQL Server) services. There are several options when it comes to using the Amazon DMS. It's possible to do the schema, schema and data (static or data change), or data only to an existing database in Amazon.

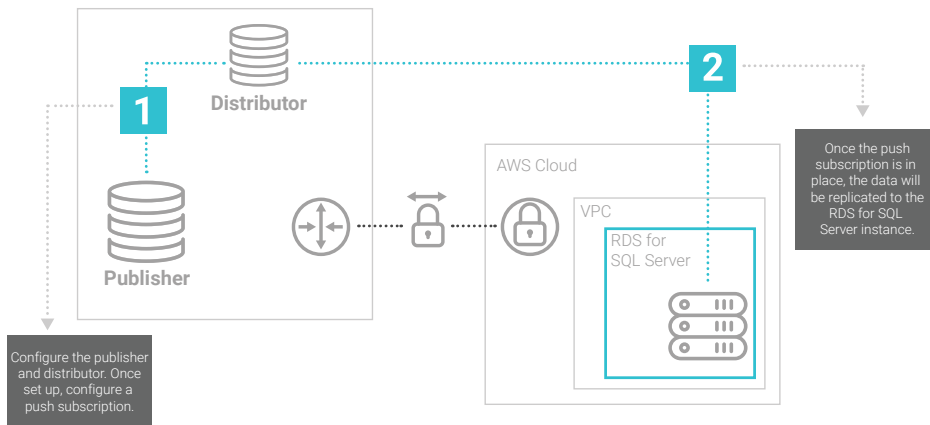
There are a few infrastructure prerequisites for a successful DMS migration, including network connectivity between the source and VPC the destination systems are located in.

Transactional Replication

RDS for SQL Server has a high degree of parity with on-premises SQL Server. One of the features supported is Transactional Replication. RDS for SQL Server can be configured as a subscriber using a push architecture.

The use of Transactional Replication for database migrations does rely on the database schema supporting the use of this configuration. More details on Transactional Replication usage restrictions can be found in the Microsoft documentation article, "[Considerations for Publishing.](#)"

If Transactional Replication is a viable option for your system, then it can be used to provide a migration mechanism that minimizes downtime during the migration event. Post-migration operations needed to decommission the source and replication topology will be minimal. The diagram below shows an example of the architecture for using Transactional Replication to migrate a database to RDS for SQL Server. More information about how to configure push subscriber replication with RDS for SQL Server can be found on the [AWS Database blog](#).



Transactional Replication to RDS for SQL Server—push subscription

Database Migration Assistant

The Microsoft Database Migration Assistant (DMA) has two project types—the analysis project, which helps you identify blockers to migration to selected targets, and a migration project. The migration project allows you to copy the database schema and data from the source system to the target server. In the case of RDS for SQL Server migrations, the DMA will only be of use in performing an analysis of the source system to identify behavioral changes. You won't be able to migrate a database to RDS for SQL Server with DMA because the account used for the migration needs to be a member of the SysAdmin group. Unfortunately, SysAdmin isn't a group supported in RDS for SQL Server; it uses a bespoke security configuration.

AMAZON EC2 VM

The Amazon PaaS offering provides a wealth of capabilities. But there are still going to be reasons to migrate to an IaaS solution with on-premises SQL Server due to constraints such as third-party vendor applications with support for only specific versions of SQL Server.

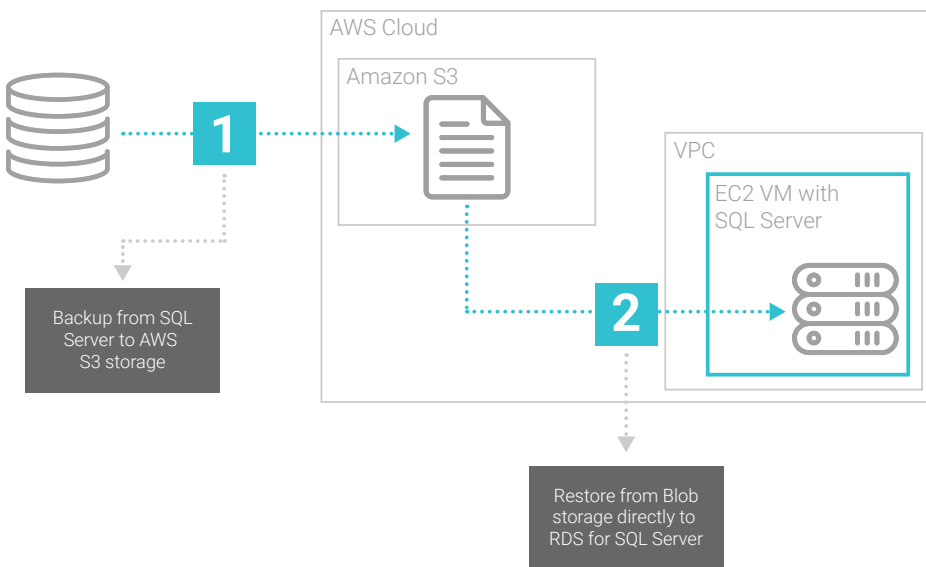
There are several routes to moving on-premises databases to IaaS. You can use both DMA and DMS to perform a migration to AWS EC2 VMs. If you're using SQL Server 2012 or newer, you can utilize Availability Groups to replicate the data. This migration model also helps with the application configuration, as the listener can be used to ease the application migration from a connectivity perspective.

Backup and Restore

You've likely been using backup and restore as a method for moving databases between SQL Servers for an exceptionally long time. This versatile options can help with moving some of the largest databases by making use of full, differential, and transaction log backup options.

There are two main routes for using backup and restore to move SQL Server databases between servers. The first is to perform it manually via scripted methods such as T-SQL or PowerShell with the **dbatools** module. The second is to make use of log shipping within SQL Server, which allows for the priming of the target environment and can drastically reduce downtime for migration activities.

The option you choose to use will depend on the connectivity between on-premises systems and AWS. If the two environments are connected via either site-to-site VPN or Direct Connect, then both options are open, regardless of whether you've extended your Active Directory (AD) domain into EC2 VMs. If the two environments are distinct, then you are restricted to the first option and have to manually perform the migration activities via Amazon S3 storage. It's possible to automate these activities, but it will involve additional services and configuration to make work smoothly.



Backup and restore a SQL Server database to EC2 VM

Native Backup and Restore

When using native backup and restore methods for migrating a SQL Server database to an EC2 VM, the best route is via Amazon S3 storage. The use of S3 storage allows for both connected and disconnected environment configurations. You should use the **dbatools** PowerShell module, which gives you a wider range of options when scripting, and subsequently automating, the process.

For connected hybrid environments in which the network and AD have been extended to an Amazon Virtual Private Cloud (VPC), having connectivity to both the source and destination makes this process easier. S3 storage would take the place of a network share you would use. At this point, the process is the same you would use for any on-premises migration between servers. You can even use PowerShell—if the version of SQL Server at the source supports backup to URL,

Using Amazon S3 storage as part of this process means there's no need to allocate temporary storage to your SQL Servers or stand up dedicated infrastructure to handle the backups. S3 storage can be created and destroyed with negligible impact to the target environment.

then use [Backup-DbaDatabase](#) from the source to the mapped drive and [Restore-DbaDatabase](#) from the same S3 bucket mapped to the EC2 machine.

In disconnected environments, where the Amazon portion is distinct from the on-premises environment, the method is the same. However, the implementation differs, as you'll need to connect to the target systems via a public route such as RDP to perform the restore operations. It isn't advisable to expose the SQL Server systems to the internet via public IP addresses on EC2 VMs.

Transaction Log Shipping

If you have a connected hybrid environment, then transaction log shipping is a viable option for migrating databases to EC2 VMs. Log shipping is a tried-and-tested technology for creating copies of databases; it can leverage many of the new features, such as backup encryption introduced with SQL Server 2014.31

There are three options to configure log shipping regarding storage for backups:

- Provision the required storage on the SQL Servers that will hold the transient backup files.
- Stand up a dedicated file server within the environment that can be accessed by both source and target servers.
- Make use of [AWS Storage Gateway](#) for files to use S3 as a storage area for transferring files between on-premises and the cloud.

Whether migrating to a single EC2 VM or to a multi-replica Always On Availability Groups configuration, log shipping can add value to the process. Once the storage is provisioned for the backup files, then it's simply a case of using the native log shipping feature in SQL Server to configure the primary and secondary servers.

Now you need to determine the timing for the shutdown of the application, wait for the final logs to be shipped, and then configure the target system. Once done, the application(s) can be reconfigured to work with the new servers and begin processing again.

MIGRATION SUMMARY

No matter which platform you plan to migrate to, the fundamental process is the same. You want to move databases from one place to another as effectively as possible with minimal input. DMS is often the first port of call for those looking to migrate to the AWS platform, no matter what your target is. The ability to manage the projects in the service and leverage the online migration option means it's one of the most effective options when looking to combine it with automation routines.

If you're not able to leverage DMS, there are still many other ways to migrate SQL Server systems to AWS.

SQL Server log shipping's ability to ship to multiple secondary systems to prime an Availability Groups configuration can simplify the deployment. It means all replicas will be at the same state, so when the primary replica is brought online, then the secondary replicas are simply added to the Availability Group.

Although this section of the guide has focused on the migration technology options, it's still important to note data testing and validation should be undertaken during the final migration. The tests and checks you defined to ensure the migration is successful need to be completed as part of the actual migration task, too. Data testing and validation will minimize risks to the migration process.

Post-Migration Operations

After the actual database migration has been completed, the work isn't done. Now is the time to watch carefully for any issues related to the migration. The sooner you can identify issues, the lower the overall impact is likely to be. The key to spotting issues early is to have effective monitoring in place with performance metrics from pre-migration to compare to regarding how the workloads performed prior to migration. The last thing you want is to have to deal with the dreaded "The application is slow" tickets.

Additionally, it's just as important to understand and ensure the maintenance routines are in place and functioning as well as Business Continuity testing to ensure if an outage occurs, you can restore service on the new platform.

Post-migration tasks that must be completed are as follows:

1. Ensure SQL Sentry monitoring has been updated to monitor the new targets and the old targets are no longer monitored.
 - It's prudent to initially stop watching the old targets to ensure historical monitoring data is retained for the old systems. Once you're comfortable with the new systems, then delete the old targets and associated data.
2. Update migration documentation in Database Mapper.
 - Once the migration is complete, it's important for the documentation to be updated to reflect the new environment. What was captured as a migration facilitator is now an important repository of support information that can assist in the platform running smoothly. It's now a living document and needs to be maintained. The documentation update can be scheduled via the automatic snapshot capabilities in Database Mapper.
3. Transition tests from migration scope to BAU data monitoring and validation.
 - As with the documentation you collected, the tests now have value to ensure the platform is functioning as required. The tests you build to verify the migration can be incorporated into application monitoring to help the application owners have increased confidence in their reports and applications.

By ensuring you carry forward the migration documentation and supporting systems, you can realize benefits in the daily operation of the hybrid or cloud platform you've migrated to.



If you encounter issues

in the post-migration phase, you should fix forward rather than roll back. Due to the complex nature of database systems, the rollback can result in incurring data loss, and the longer between migration and detection can mean the difference between remaining a viable business or not.

MONITORING

Earlier, we discussed how having SQL Server monitoring in place early in the migration planning process can solve many problems as you test and validate workloads on the new systems. You also need to ensure once the workload is active on the new platform, it's in line with your pre-migration baselines and performance thresholds.

It's also vital to monitor for discrepancies in the data contained in the migrated systems. This is especially true when you had to make modifications to the databases or applications to support the new version of SQL Server you're now running.

Performance Monitoring

The first thing end users will complain about is performance. Without metrics to back up your position that the system is functioning well within expected parameters, you face an uphill battle.

Therefore, it's important to have a platform spanning pre- and post-migration configurations. SQL Sentry provides powerful, scalable performance monitoring for SQL Server on Windows, Linux, and RDS for SQL Server, with all the data stored in one central database. As such, it's possible to have a single pane of glass spanning pre- and post-migration. Leveraging SQL Sentry Advisory Conditions, which look at the historical performance data and baselines and compare to the current workload profile to alert when a deviation is detected, is ideal.

Data Integrity Monitoring

When data changes go wrong and incorrect data is stored, you can encounter a lot of problems. This is one of the more insidious issues, as it isn't typically noticed for some time, at which point there's almost no chance of rolling back. The sooner you can spot this issue and remediate it, the better the situation will be. Incorrect data can be stored for any number of reasons, ranging from incorrect collation to the language used by the application login.

The quickest and easiest way to prevent data quality issues is to take the test suites you created as part of the migration validation and testing phase and transition them from integration tests to production data monitoring tests. Having regular data validation in place also means you're already ahead of the game when performing upgrades to the application or database.

Another advantage of transitioning your migration testing into production data monitoring is you can incorporate the test results into your reporting and ETL routines. One of the simplest things you can do is add functionality to your reports that gives a "confidence factor." If all tests complete successfully, then confidence is marked as high. If there are failures, the issues can be conveyed to the application owners. Confidence can be indicated by a traffic light system or percentages depending on the metric you want to use.

MAINTENANCE ACTIVITIES

Because the SQL Server offerings on AWS are based on on-premises SQL Server, you'll need to deploy maintenance tasks. These include CHECKDB, index and statistics maintenance, and, in the case of EC2 VMs, regular database backups to a secure location.

Likewise, backup operations are handled natively by RDS for SQL Server. But how do you know the backups are being completed? This is where monitoring provides a level of visibility. SQL Sentry monitors the backup operations not only on SQL Server in VMs but also on RDS for SQL Server. This monitoring provides a level of confidence that the backups are happening. It also allows you to identify if the backup operations impact the performance of your database workloads.

Index and Stats Maintenance

On PaaS solutions, index and statistics maintenance consistently get overlooked. There's a misconception that the cloud just handles everything. However, they're still based on the same engine as on-premises SQL Server, and they need reliable statistics to be able to make decisions about how to execute queries.

For SQL Server systems running in IaaS VMs, SQL Sentry can provide deep insight into index utilization as well as managing any fragmentation found by the analysis.

Leveraging SQL Agent jobs, in combination with deploying scripts such as Ola Hallengren's maintenance solution, allows you to manage maintenance operations effectively.

BUSINESS CONTINUITY TESTING

The final stage of the migration process is to ensure all staff are familiar with what it takes to bring the system back online if there's an outage from the cloud vendor. Outages can and will happen, so your database team needs to be prepared for it.

Effective documentation is essential to Business Continuity. Now that you've built a documentation repository for the systems you've migrated, you can use it to enhance your Business Continuity response.

Database Mapper offers scheduled environment scanning to detect changes and allows you to compare scans to help reduce identify drift in your environment. The added benefit of Database Mapper is it puts the ownership of maintaining the documentation in the hands of application owners and SMEs via a lightweight web interface.

The information Database Mapper provides will allow the operations staff to recreate an environment rapidly if there's a disaster. And, it allows the operations staff to identify affected components should an outage occur.

Summary

Whether you migrate to EC2 VMs or RDS for SQL Server, you now have established patterns and practices you can implement to minimize risk and increase the chance of success with your migration.

Through a combination of tried and tested methodologies and new AWS services supported by [SolarWinds SQL Sentry](#) and [SolarWinds Database Mapper](#), it's possible to accelerate and de-risk cloud migration activities to build reliable and efficient data platform solutions in the cloud.

Disclaimer

The sample code, processes and advice in this guide is provided "as is" and in any express or implied warranties, including the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services, loss of use, data, or profits; or business interruption) sustained by you or a third-party, however caused and on any theory of liability, whether in contract, strict liability, or tort arising in any way out of the use of this sample code, even if advised of the possibility of such damage.

ABOUT SOLARWINDS

SolarWinds (NYSE:SWI) is a leading provider of powerful and affordable IT management software. Our products give organizations worldwide—regardless of type, size, or complexity—the power to monitor and manage their IT services, infrastructures, and applications; whether on-premises, in the cloud, or via hybrid models. We continuously engage with technology professionals—IT service and operations professionals, DevOps professionals, and managed services providers (MSPs)—to understand the challenges they face in maintaining high-performing and highly available IT infrastructures and applications. The insights we gain from them, in places like our **THWACK** community, allow us to solve well-understood IT management challenges in the ways technology professionals want them solved. Our focus on the user and commitment to excellence in end-to-end hybrid IT management has established SolarWinds as a worldwide leader in solutions for network and IT service management, application performance, and managed services. Learn more today at www.solarwinds.com.



For additional information, please contact SolarWinds at 866.530.8100 or email sales@solarwinds.com.
To locate an international reseller near you, visit http://www.solarwinds.com/partners/reseller_locator.aspx

© 2021 SolarWinds Worldwide, LLC. All rights reserved

The SolarWinds, SolarWinds & Design, Orion, and THWACK trademarks are the exclusive property of SolarWinds Worldwide, LLC or its affiliates, are registered with the U.S. Patent and Trademark Office, and may be registered or pending registration in other countries. All other SolarWinds trademarks, service marks, and logos may be common law marks or are registered or pending registration. All other trademarks mentioned herein are used for identification purposes only and are trademarks of (and may be registered trademarks) of their respective companies.

This document may not be reproduced by any means nor modified, decompiled, disassembled, published or distributed, in whole or in part, or translated to any electronic medium or other means without the prior written consent of SolarWinds. All right, title, and interest in and to the software, services, and documentation are and shall remain the exclusive property of SolarWinds, its affiliates, and/or its respective licensors.

SOLARWINDS DISCLAIMS ALL WARRANTIES, CONDITIONS, OR OTHER TERMS, EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, ON THE DOCUMENTATION, INCLUDING WITHOUT LIMITATION NONINFRINGEMENT, ACCURACY, COMPLETENESS, OR USEFULNESS OF ANY INFORMATION CONTAINED HEREIN. IN NO EVENT SHALL SOLARWINDS, ITS SUPPLIERS, NOR ITS LICENSORS BE LIABLE FOR ANY DAMAGES, WHETHER ARISING IN TORT, CONTRACT OR ANY OTHER LEGAL THEORY, EVEN IF SOLARWINDS HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.