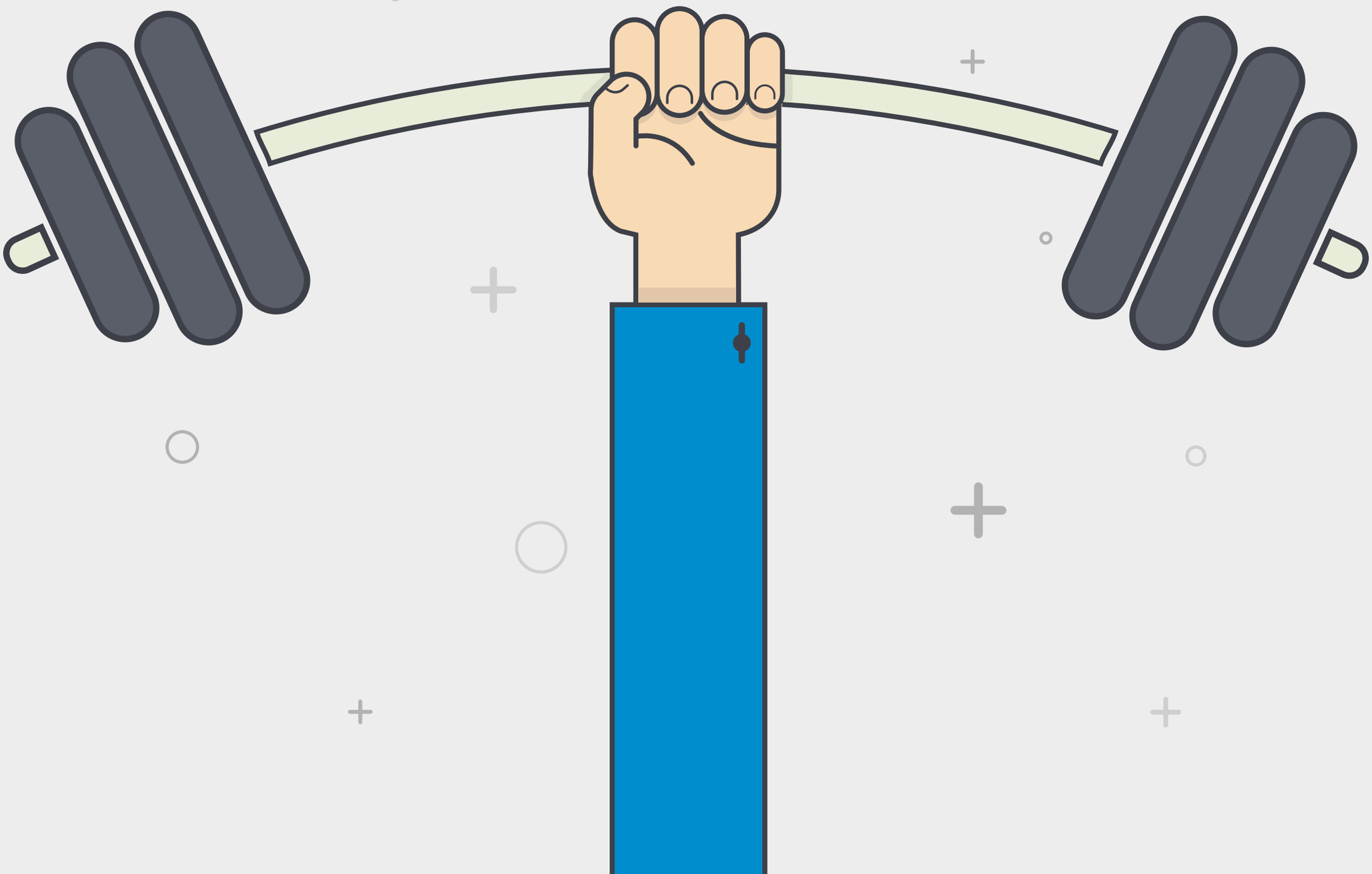
HOW TO INCREASE YOUR SHAREPOINT PERFORMANCE

An eBook by **Sharegate**



Abstract

As a SharePoint Consultant, I get to see dozens of SharePoint farms every year, and one of the most common complaints I get from clients is that SharePoint is slow. A lot of people think that SharePoint is slow because the SharePoint servers are lacking resources, or simply, because SharePoint is a slow product. Although resources allocated to the SharePoint servers (e.g. Web Front Ends and App servers) are important, not a lot of people realize that SharePoint performance is directly related to the database, SQL Server. In fact, 94% of SharePoint data is stored in SQL.

In this whitepaper, we will learn how to plan things before the installation, explore options on how to optimize SQL Server 2012 for SharePoint 2013, and what to do after the installation is completed

AUTHOR

Vlad Catrinescu

Vlad is a SharePoint and Office 365 Consultant specializing in SharePoint and SharePoint Online deployments as well as hybrid scenarios. As a Pluralsight Author, Microsoft Certified Trainer and recognized international speaker, Vlad has helped thousands of users and IT Pros across the globe to better understand and to get the most out of SharePoint.

Vlad is also a Microsoft Most Valuable Professional (MVP) in SharePoint since 2013 and is known in the community for his technical abilities and for co-founding the largest and most active SharePoint Community that you can find at www.SharePoint-Community.net. Vlad also has his own blog at www.absolute-sharepoint.com and he often shares his knowledge by speaking at local conferences and community events.

In Addition, Vlad is the recipient of the "Top 25 Office 365 Influencers" award, showcasing him as one of the most influential Office 365 specialists in 2015, demonstrating his expertise and passion for Microsoft's shift towards the cloud in the past years.

@vladcatrinescu

http://ca.linkedin.com/in/vladcatrinescu

www.absolute-sharepoint.com

pluralsight.com/authors/vlad-catrinescu

REVIEWERS

Special Thanks

Vlad would like to thank the following people for taking the time to review this whitepaper to ensure its quality prior to publication.

Mark Jones
founder, **Collaboris**

Gokan Ozcifci
Microsoft Community Contributor

Jesper M. Christensen
SharePoint Architect and founder of **extri:co a/s**

Jasjit Chopra
SharePoint Architect & Founder of **SharePointPower.com**

Sharegate

More Checkmarks Mean Better SharePoint Performance

Here's a printable **Checklist** of everything we'll cover.

1. Getting prepared

- ☐ Set the NTFS Allocation size to 64k
- ☐ Check the Network Speed available

2. Installing SQL Server 2012

- ☐ Select the drive for the SQL binaries
- ☐ Configure the SQL Server instance
- ☐ Set the accounts for the Agent and Database Engine Services
- ☐ Customize the Server Collation
- ☐ Database Engine Configuration

3. Post Installation Configuration

3.1 Server Properties

- ☐ Set the maximum SQL Server Memory allocation
- ☐ Set the minimum SQL Server Memory allocation
- ☐ Specify the Index Data storage Fill Factor

3.2 Model Database

- ☐ Define the Initial Size of the Model Database
- ☐ Adjust the auto growth size
- ☐ Disable Auto-Create Statistics
- ☐ Set the max degree of parallelism
- ☐ Enable Instant File Initialization

3.3 Tembdb

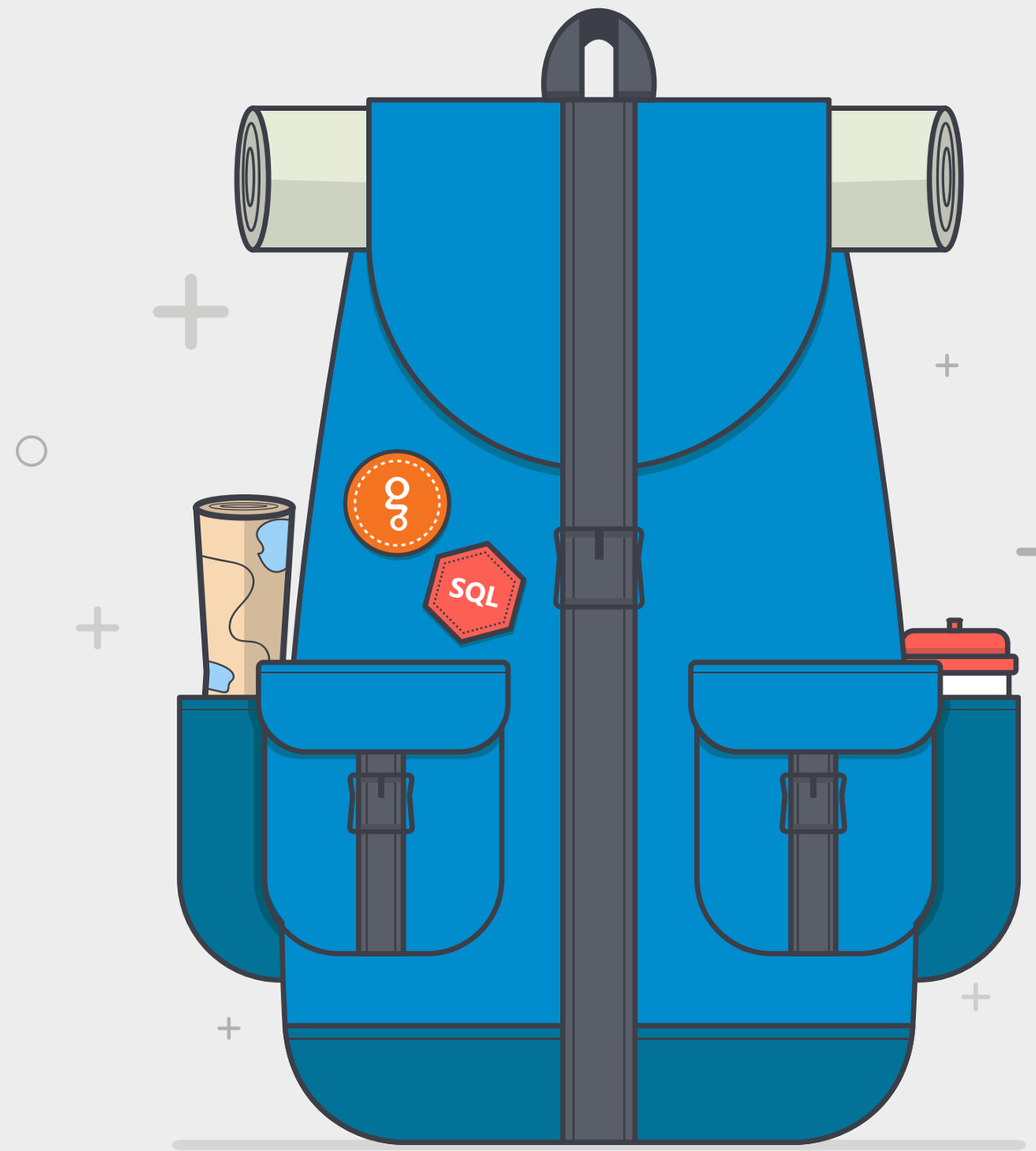
- ☐ Configure Initial Size & Autogrow
- ☐ Set the Multiple Files ratio

INSTALL SHAREPOINT 2013

Table of Contents

CHAPTER **ONE**

Getting Prepared



Before installing SQL Server 2012, we first have to plan how to configure it. This step is very important because, if you don't spend enough time planning and preparing, this will lead to problems further down the line.

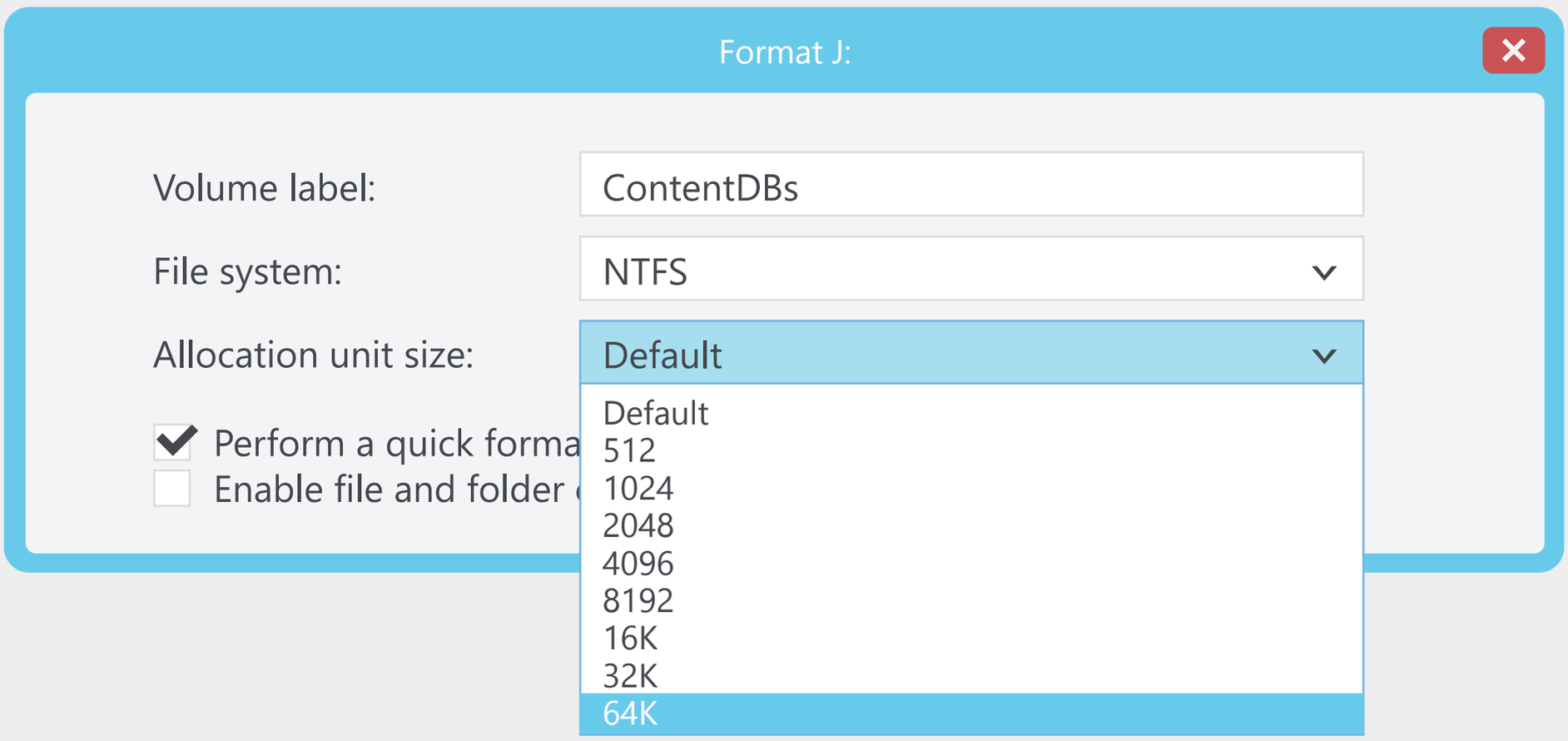
NTFS Allocation Size

SQL server reads and writes 64k at a time, therefore it's a best practice to make sure your disks are formatted with an allocation size of 64K and not the 4K default. This small change alone will improve your SQL performance by up to 30%.

To check what your allocation size is, in an Administrator Command Prompt type: "chkdsk c:" (Where C: is your disk letter). You'll end up seeing something like this:

```
    4096  bytes in each allocation unit.
58607103  total allocation units on disk.
20257161  allocation units available on disk.
```

In this example, you see that the allocation size is 4K on this disk! To change it, you'll have to reformat your hard drive, which isn't always possible. However, if you're installing your SQL now, you can change the default allocation size when formatting the disk.



If you check your disk again afterwards, you'll see it's been successfully set in 64K! (64*1024=65536)

```
    65536  bytes in each allocation unit.
  9762031  total allocation units on disk.
  7737514  allocation units available on disk.
```

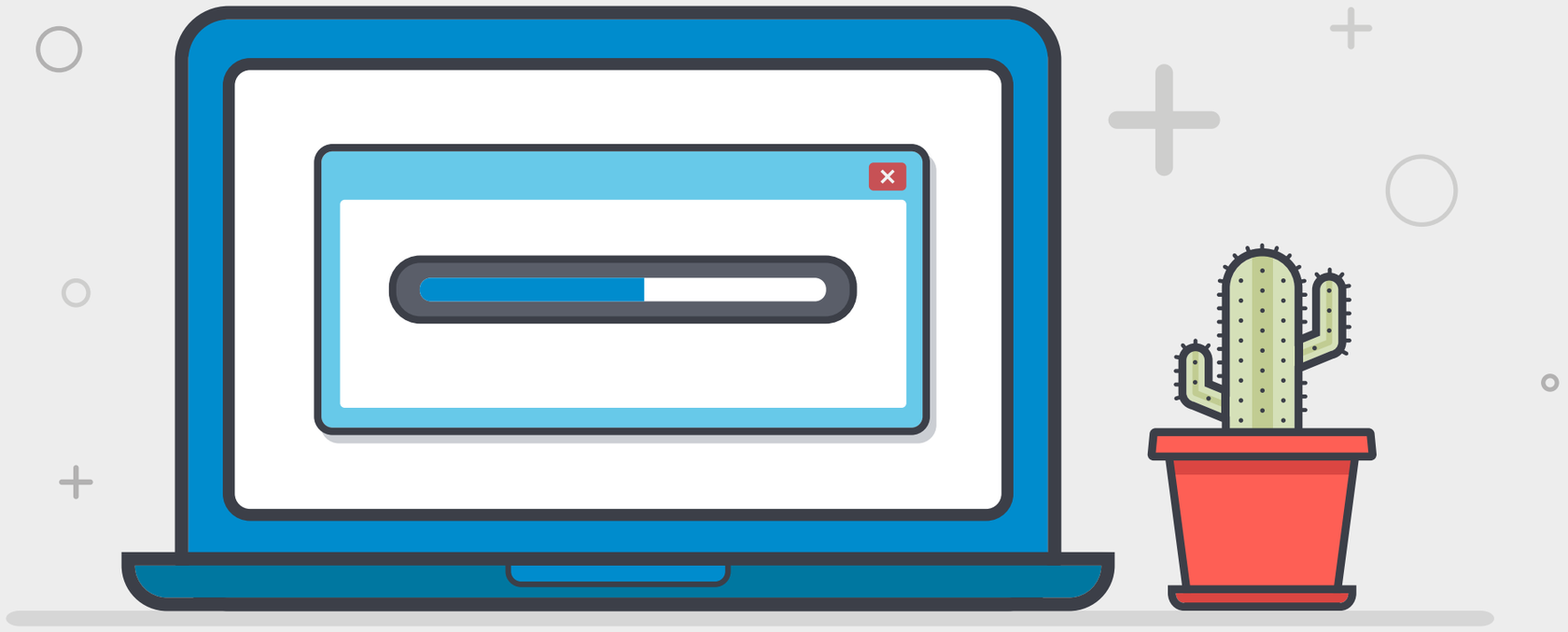
Network Speed

As you already know, everything between SQL and SharePoint travels through the network. If both your SQL and SharePoint are all on SSD Drives with 20GB of RAM, but you only have a 1Mbs cable between the two, that also answers client requests, you will still hit a bottleneck.

You need to make sure you have fast transfer speeds between your SQL and SharePoint. Furthermore, use a dedicated network adapter strictly for SQL-SharePoint communication, and a different one for client requests (if you can, obviously). You can also use Windows Server 2012 NIC Teaming feature to make them faster!

CHAPTER **TWO**

Installing SQL Server 2012



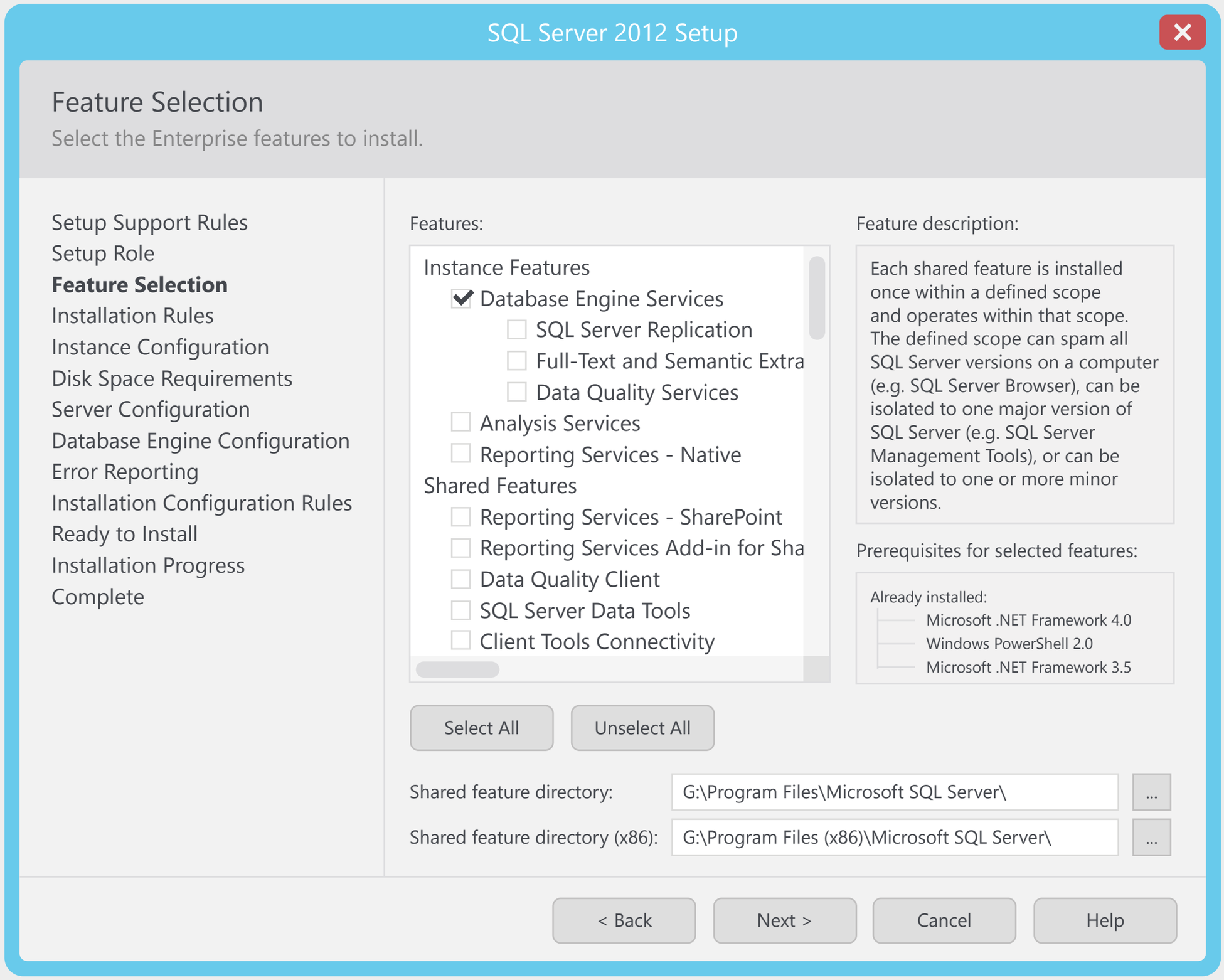
It's true that starting on the right foot is very important, especially in SQL, since some settings can only be done once and not changed afterwards!

However, if your SQL is already installed, don't worry! There are many optimizations you can still do. They are outlined in Section 3:

Post Installation Configurations.

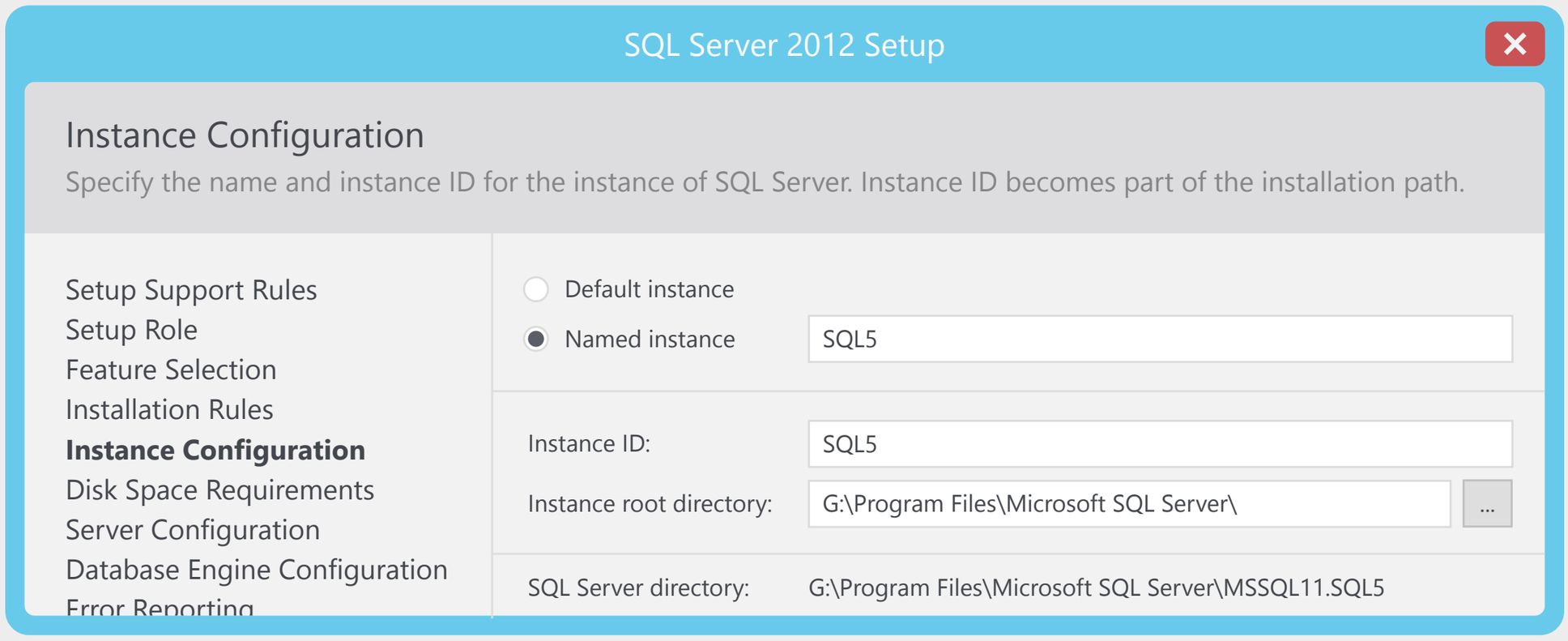
Feature Selection Screen

In the feature selection screen, you can change the location of the SQL binaries. As a rule of thumb, you want to keep the C: drive only for the OS, so we'll install the binaries on a separate drive (the G: drive in this case).



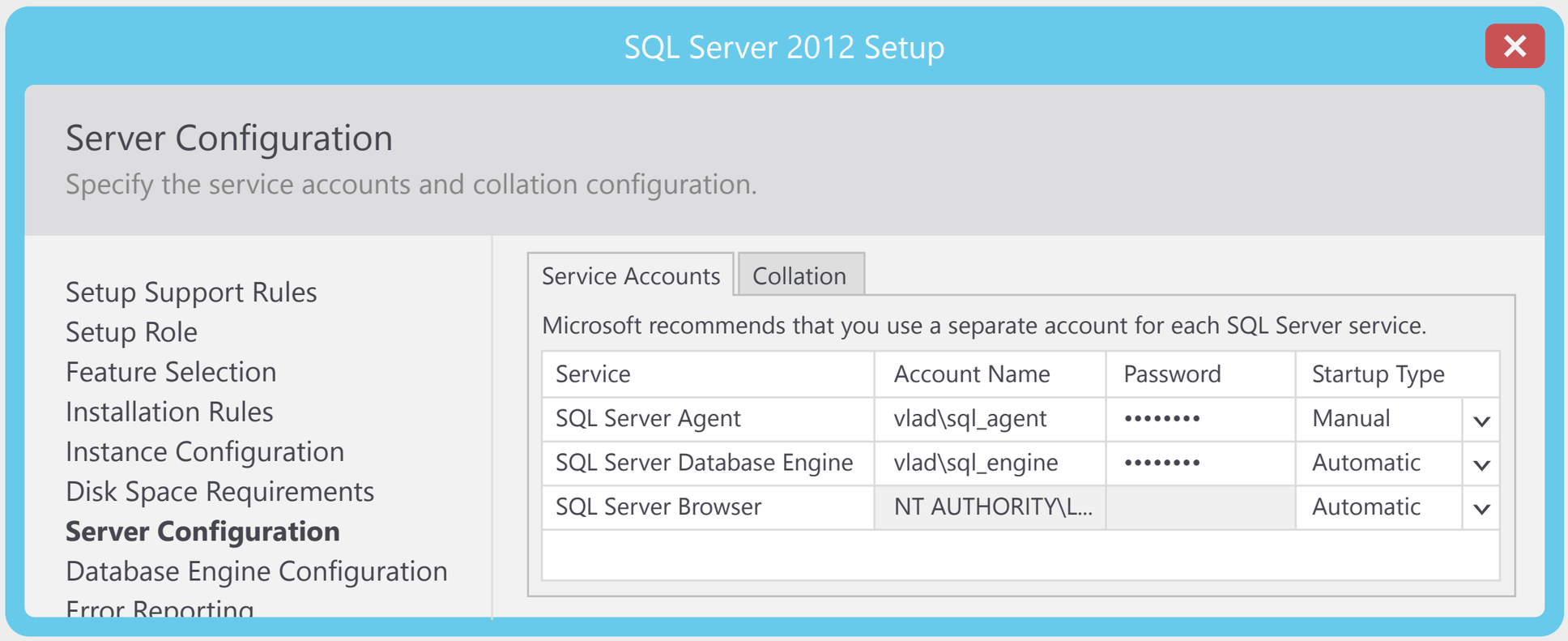
Instance Configuration

In the instance configuration page, you can either specify to use the default instance (if you don't already have one) or use a named one. Using named instances is always neat, but you don't have to! Furthermore, in this screen, you also need to change the Instance root directory to match the settings from the "Feature Selection" screen. Using a named instance vs Default Instance doesn't improve performance.



Server Configuration – Service Accounts

In this screen, we will set the Service Accounts for the Agent and Database Engine Services. It's recommended to have at least one account for both (ex: SQL_Services), or one account per service. This is also not for performance, but for security.



Server Configuration – Collation

95% of the time people have never changed this setting, and it's not fatal. As a matter of fact, SharePoint is built to accept any SQL Server 2012 Collation, however that's not the collation SharePoint Server 2013 uses. SharePoint 2013 uses **Latin1_General_CI_AS_KS_WS**. This is a quote from Microsoft to prove it:

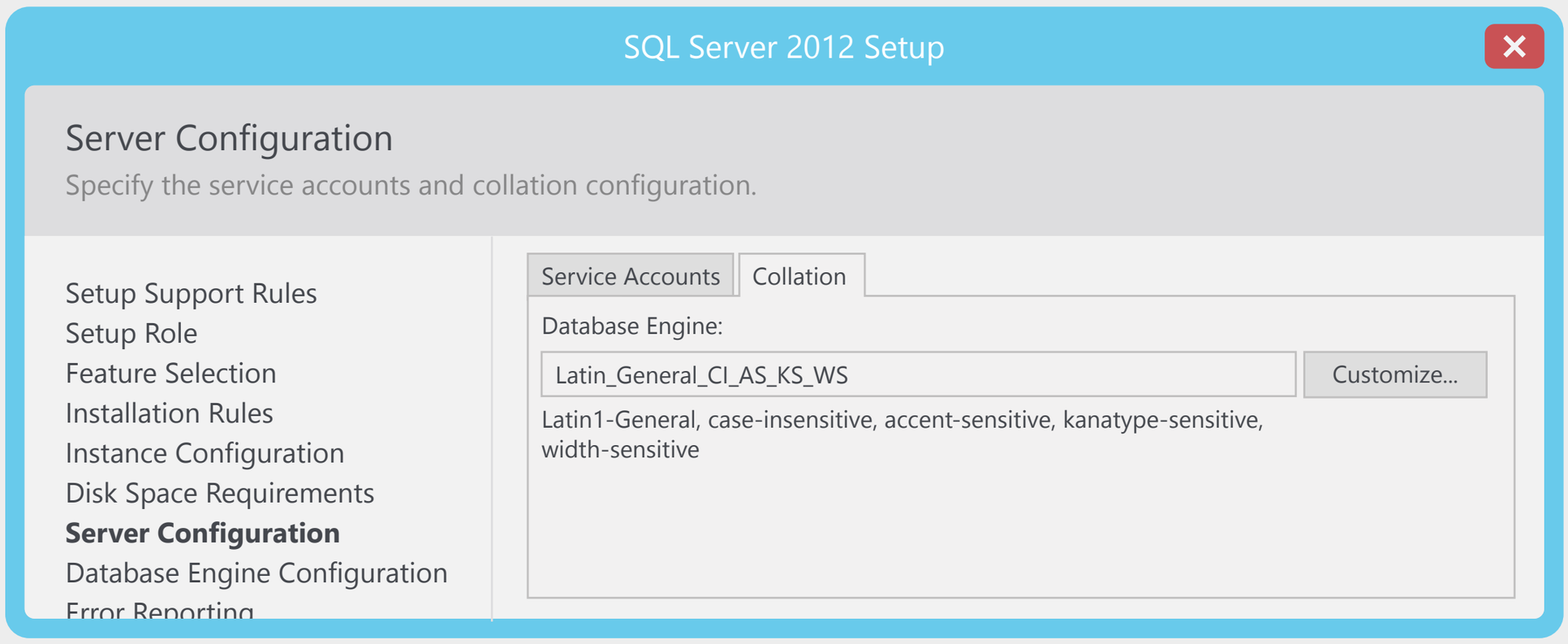
"We support any CI collation for the SQL instance **(for master, tempdb databases).** However, we recommend using **Latin1_General_CI_AS_KS_WS** as the instance default collation **(master, tempdb databases)**."

But, what does **CI_AS_KS_WS** stand for anyway?

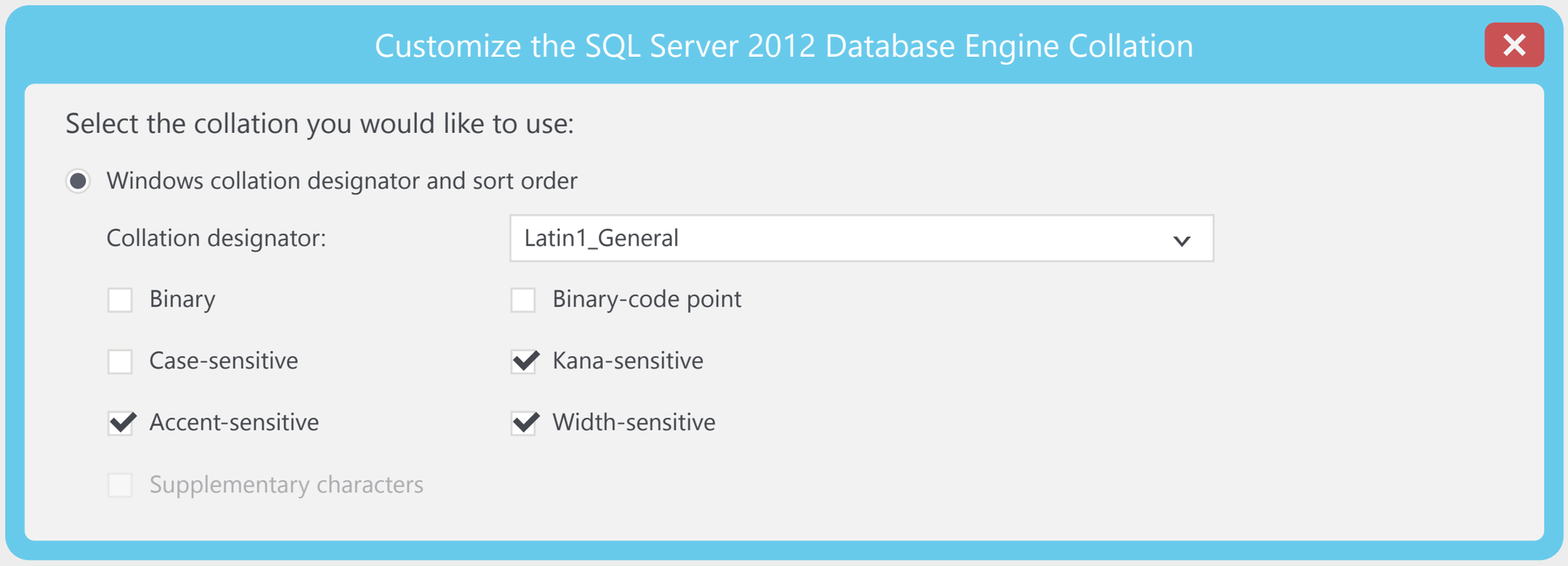
- **CI – (Case Insensitive)** "A" and "a" **ARE** treated as the same character.
- **AS – (Accent Sensitive)** "a" and "á" are **NOT** treated as the same character.
- **KS – (Kana Sensitive)** Japanese Hiragana and Katakana characters which look the same are **NOT** treated as the same character.
- **WS – (Width Sensitive)** Single-Byte and Double-Byte versions of the same character are **NOT** treated as the same character.

This is another one of those settings that must be done correct from the start as you cannot change the Instance Collation after it's installed. But, how do we do it?

On the Collation page, click the Customize button.



In the following Screen, select Latin1_General as Collation designator, and check the following boxes:

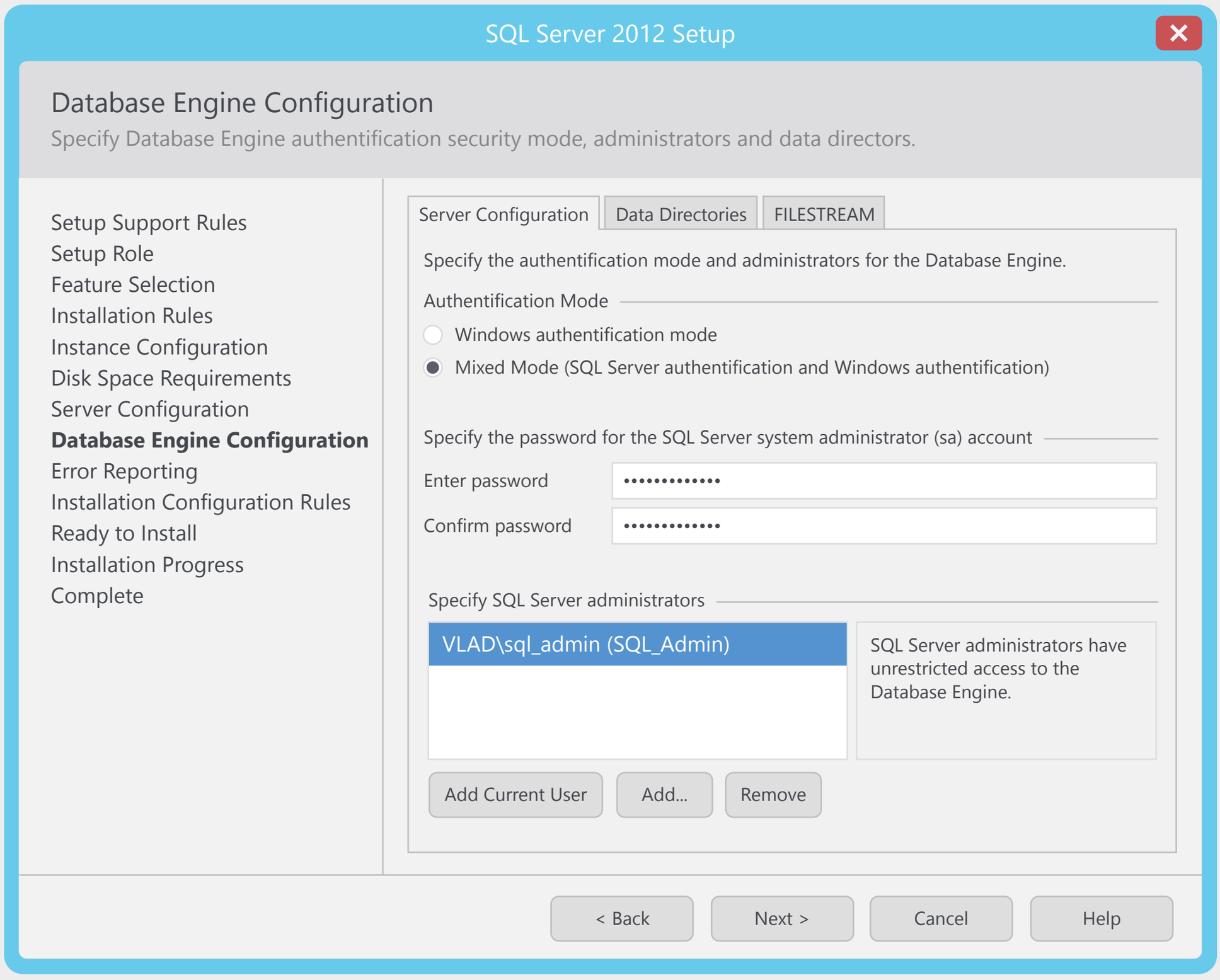


Now, your collation should look like this:

If you didn't do it initially, don't worry because SharePoint 2010/2013 will automatically set the correct collation during database creation. However, if you want your DBA to pre-provision your databases, they'll need to use the SharePoint collation when creating the databases.

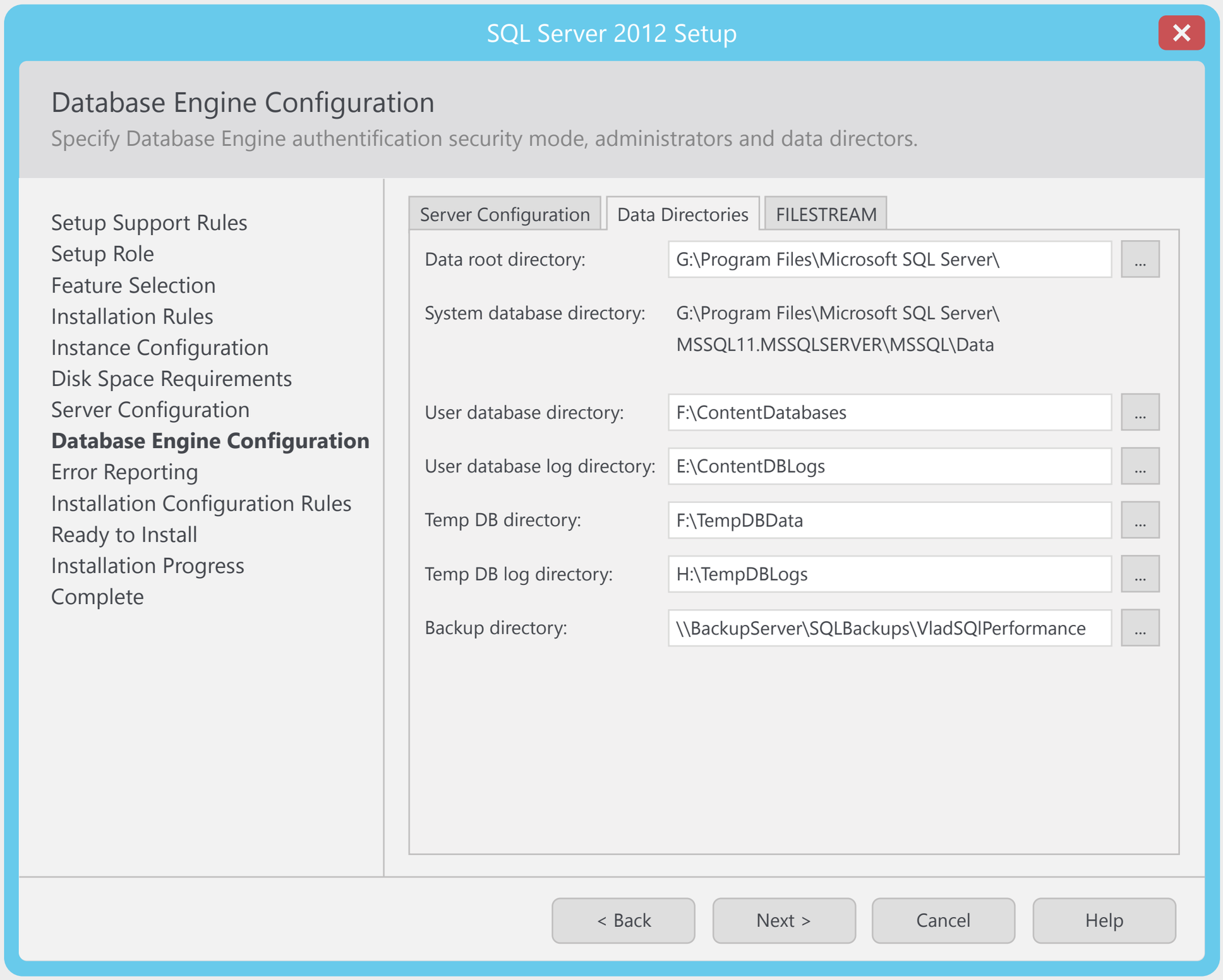
Database Engine Configuration – Server Configuration

In this screen, we decide if we only use Windows authentication or Mixed Mode as well as your SQL admins. It's strongly suggested to use Mixed Mode and to keep the sa account password safe. Use sa only when there's no other way to log into an instance of SQL (for example, when other system administrators are unavailable, or have forgotten their passwords). This is again only for security and not for performance!



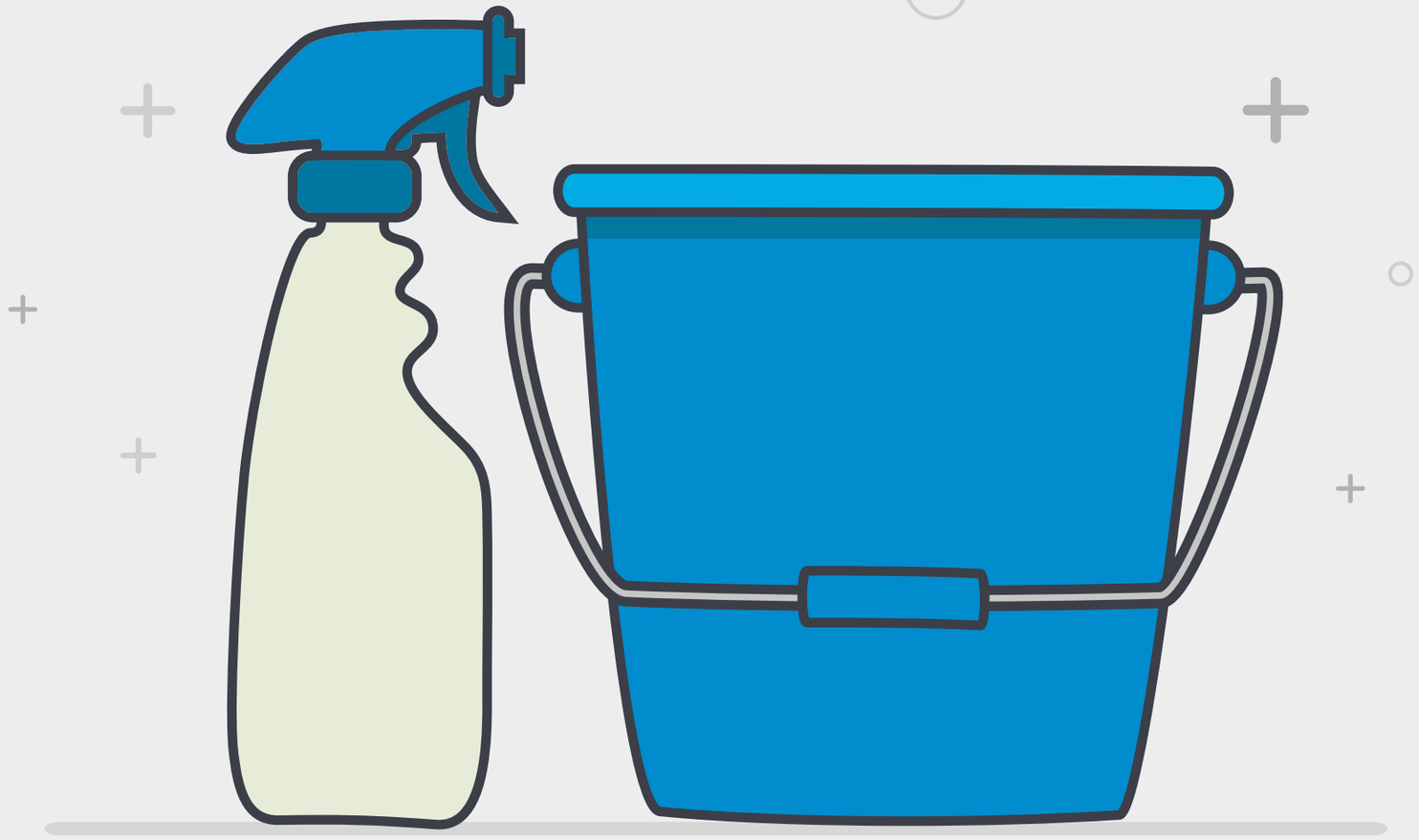
Database Engine Configuration - Data Directories

This screen will put into practice everything we planned in the first part of this White Paper. Here's where we tell the SQL server the default location for the TempDB, ContentDB's , …, data, and logs. Use what we planned in the first section and configure it!



CHAPTER **THREE**

Post Installation Configurations



After the SQL Server installation is done, there are some changes we need to do before installing SharePoint Server 2013 to make sure everything is set up. If you've already installed your SharePoint, you can also make those changes now to make it faster.

Server Properties

To access the Server Property, right click on the server name, and click properties.

Maximum Server Memory

By default, SQL Server is set to use max 2TB of RAM, although it's doubtful you have that much! This effectively means that SQL can consume all of the RAM in your server leaving nothing for the OS or other applications. This can cause performance issues. Here's how Thomas Larock, from SQL Rockstar explains it:

SQL Server (and other database systems such as Oracle and Sybase) needs to read data pages into their internal memory before they can be used. Of course your server needs memory to operate as well. When your database engine and your server are competing for the same memory resources, you get bad performance. You want your server and your database engine to be like dancing partners, and less like my kids fighting over the last cupcake.

There's a nice formula to define how much RAM you should dedicate to all the SQL instances on the server, to make sure there's enough left for the OS but... unfortunately it's not easy!

- **SQL Max Memory** = TotalPhyMem - (NumOfSQLThreads * ThreadStackSize) - (1GB * CEILING(NumOfCores/4))

- **NumOfSQLThreads** = 256 + (NumOfProcessors*- 4) * 8 (* If NumOfProcessors > 4, else 0)

- **ThreadStackSize** = 2MB on x64 or 4 MB on 64-bit (IA64)

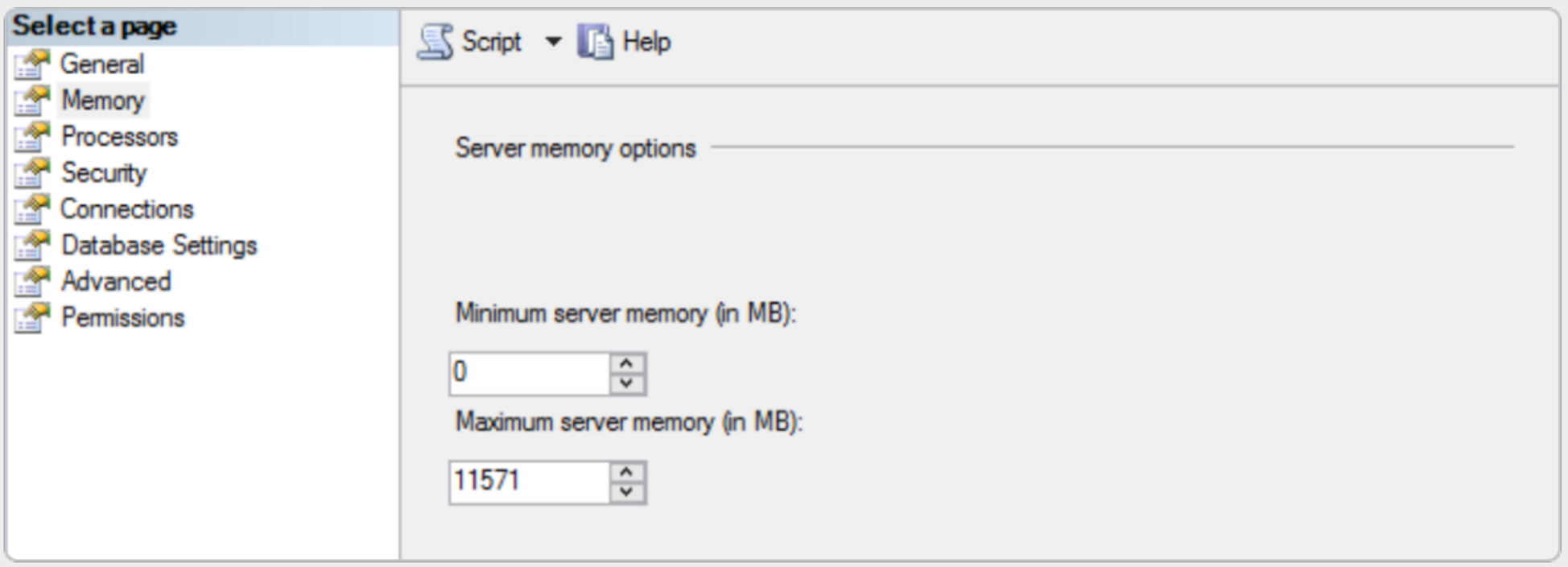
It's not easy, and it actually misses a part really dedicated for the OS. To help you set the correct values, I developed a utility that automatically calculates this figure. There are some examples shown below.

Here are some examples based on a quad core server on x64 architecture!

Server Ram (IN GB)	Ram for all SQL Instances together (IN MB)
8	5012
12	8294
16	11571
24	19968
32	27136
64	55808

So, for example, if you have an 8GB, 4 core, x64 server with two instances, you'll need to split the 5017 MB of RAM between the 2. Ex: instance 1: 2000, instance 2: 3017

Here's where you set it:



However, since SQL 2008 R2, you don't have to set the Max Server Memory if you only have one instance. The memory manager component of Microsoft SQL Server eliminates the need for manual management of the memory available to SQL Server.

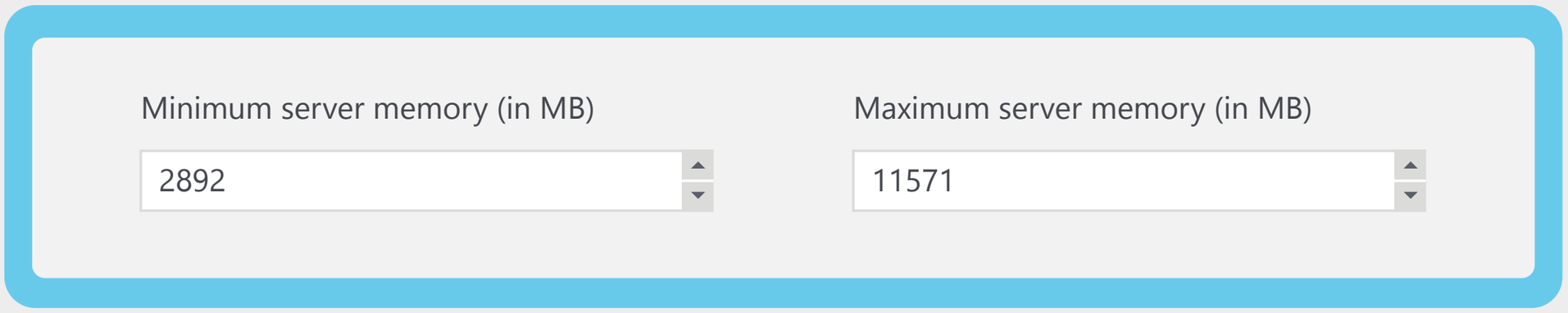
When SQL Server starts, it dynamically determines how much memory to allocate based on how much memory the operating system, and other applications, are currently using. As the load on the computer and SQL Server changes, so does the memory allocated.

Minimum Server Memory

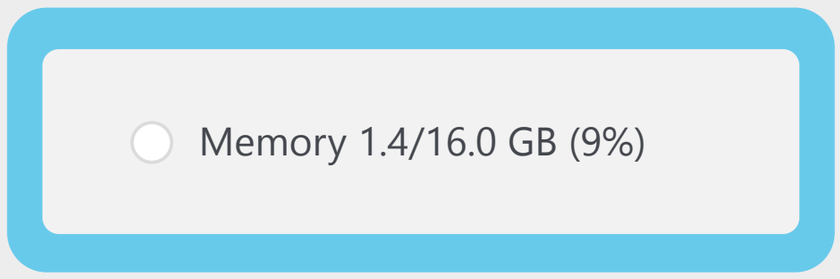
The **Minimum Server Memory** server configuration option can be used to ensure that SQL Server doesn't release memory below the configured minimum server memory once that threshold is reached. This configuration option can be set to a specific value based on the size and activity of your SQL Server. If you choose to set this value, set it to some reasonable value to ensure that the operating system doesn't request too much memory from SQL Server, which can affect SQL Server performance.

Pro Tip: Set it at 25% of the Max server memory for a farm where data requested by SharePoint changes a lot, and to 60% of the Max server memory when the data requested by SharePoint is almost always the same.

Also, note that even if you set the minimum server memory, it doesn't mean SQL will automatically take all that memory. It will only take it when it needs it!

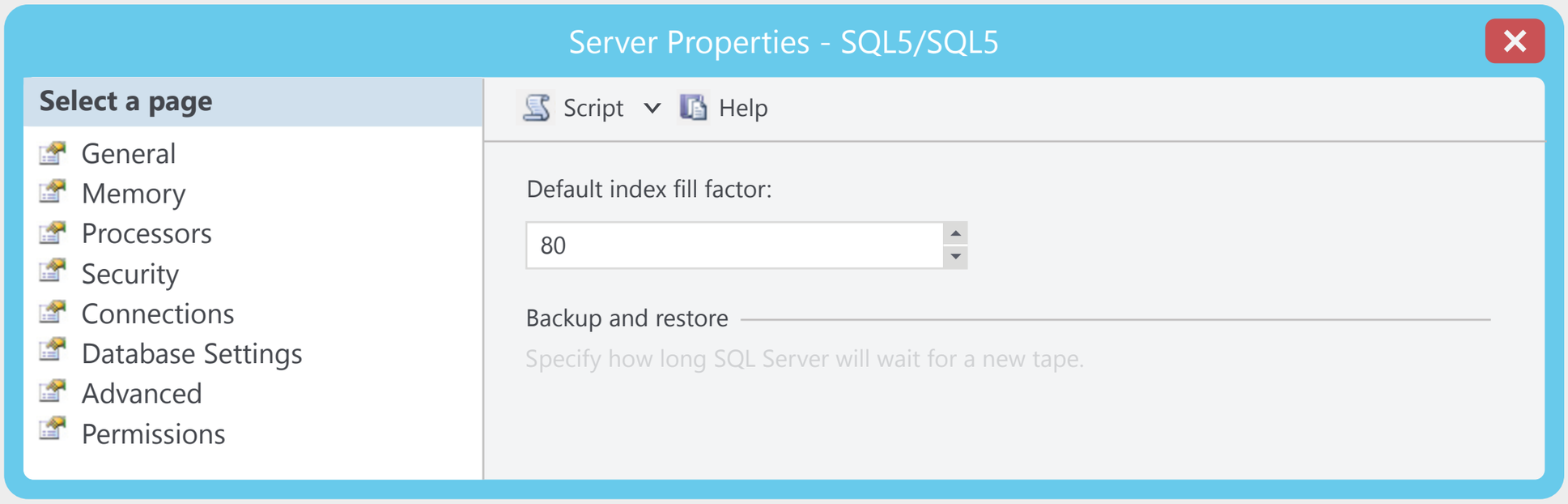


In Task manager, you can also see the Memory usage made by SharePoint. (In our example, because SharePoint isn't actively used at the moment, the server only consumes 1.4 GB.)



Fill Factor

To further improve Index Data storage and performance, use fill factor. When indexes are created or rebuilt, the fill factor value (1-100) determines the percentage of space that can be filled with data on each leaf level page. The remaining space is reserved for future growth. For many situations, the default server-wide fill factor level of 0 (fill each page to 100% full) is optimal. However, for SharePoint, a server-wide setting of 80 is optimal to support growth and minimize fragmentation.



Model Database

SQL Server uses the Model database as a template for creating new user Databases. When a new user Database is created, SQL Server copies contents of the model to the new Database, and fills the rest of the space with empty data pages.

Before even installing SharePoint 2013, we will make the Model DB as good as possible, so all future Databases will inherit these settings. However, be careful, SharePoint doesn't use ALL the settings, so there are some things you'll need to modify later.

Initial Size

The initial size is the size of the Databases when they'll be created. The default is only set at 3MB. That means that when your content Databases are created, they only take up 10MB. This means every time you do something in SharePoint, such as adding a document, the Database will have to grow (more info later) before being able to write that data.

This is an operation that can be avoided by correctly setting a Database size upfront. It's a best practice to set your initial size to how much data you expect to have per content Database in a year. Yes, it'll take more space on your disks initially, however your performance will be far better. In our example, I estimated that my Content Databases will reach 500MB after the first year, so my Initial Size is 500MB. Furthermore, the log should be at about 25% of the Database initial size.

Auto growth

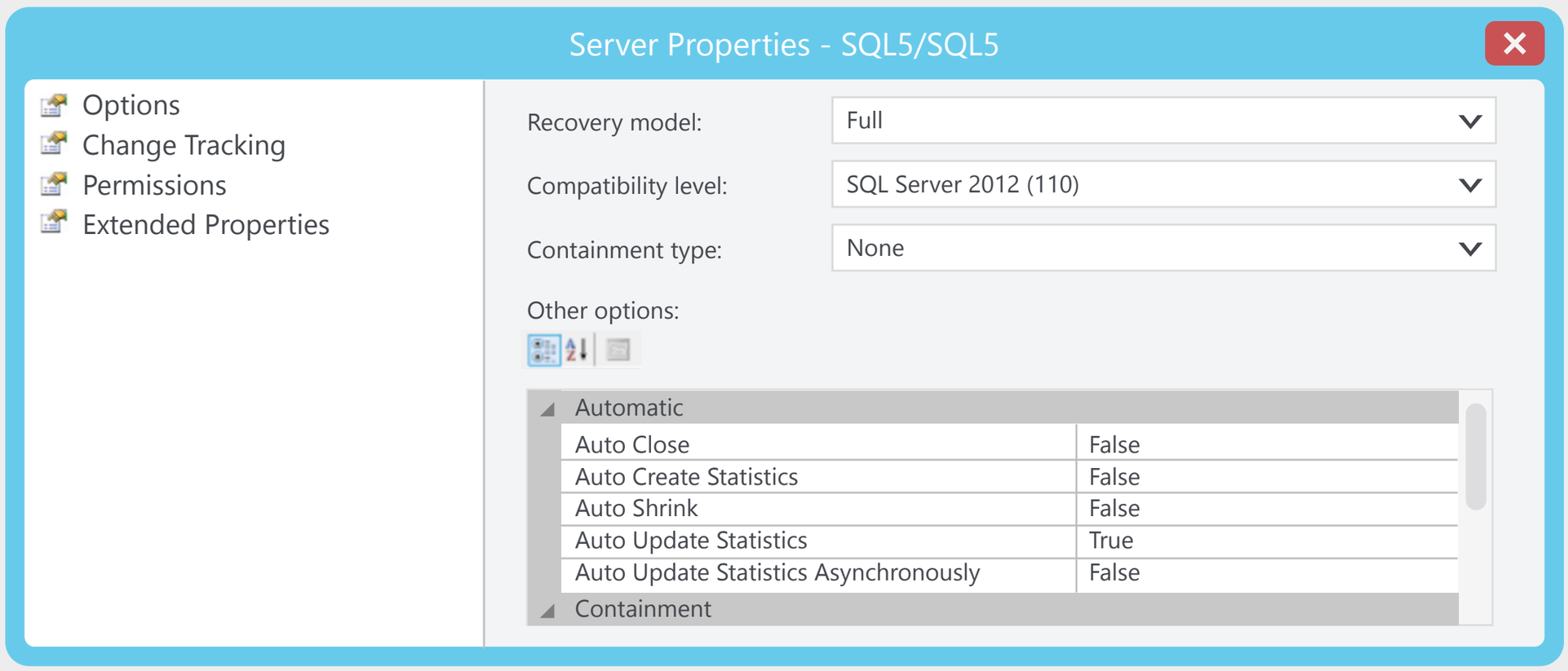
The auto growth is your insurance policy if your current database size has been reached. Instead of making the Database read only and stopping users from writing to it, SQL allows you to set by how much you want your database to grow when DB size has been reached. By default, for data files, it's set at only 1MB, but that isn't optimal at all. Why? Imagine you're 1MB away from your size, and user uploads a 10MB document. Your SQL will have to grow 9 different times in order to be able to put the document in the database.

That adds tasks to the SQL which will make it slower for the user. Now you might ask, but if we put it at a bigger number, ex: 200MB, won't the user have to wait for it to grow 200MB before being able to write his document? Yes, that's exactly true. Read on for a trick which will make this auto grow almost instant, and you'll only have to grow the Database once instead of 9 times.

Pro Tip: Set the Auto Growth in MB and not in % because it allows you to monitor and know by exactly how much it'll grow. Furthermore, the Auto Growth should be at about 50% of the Initial Size!

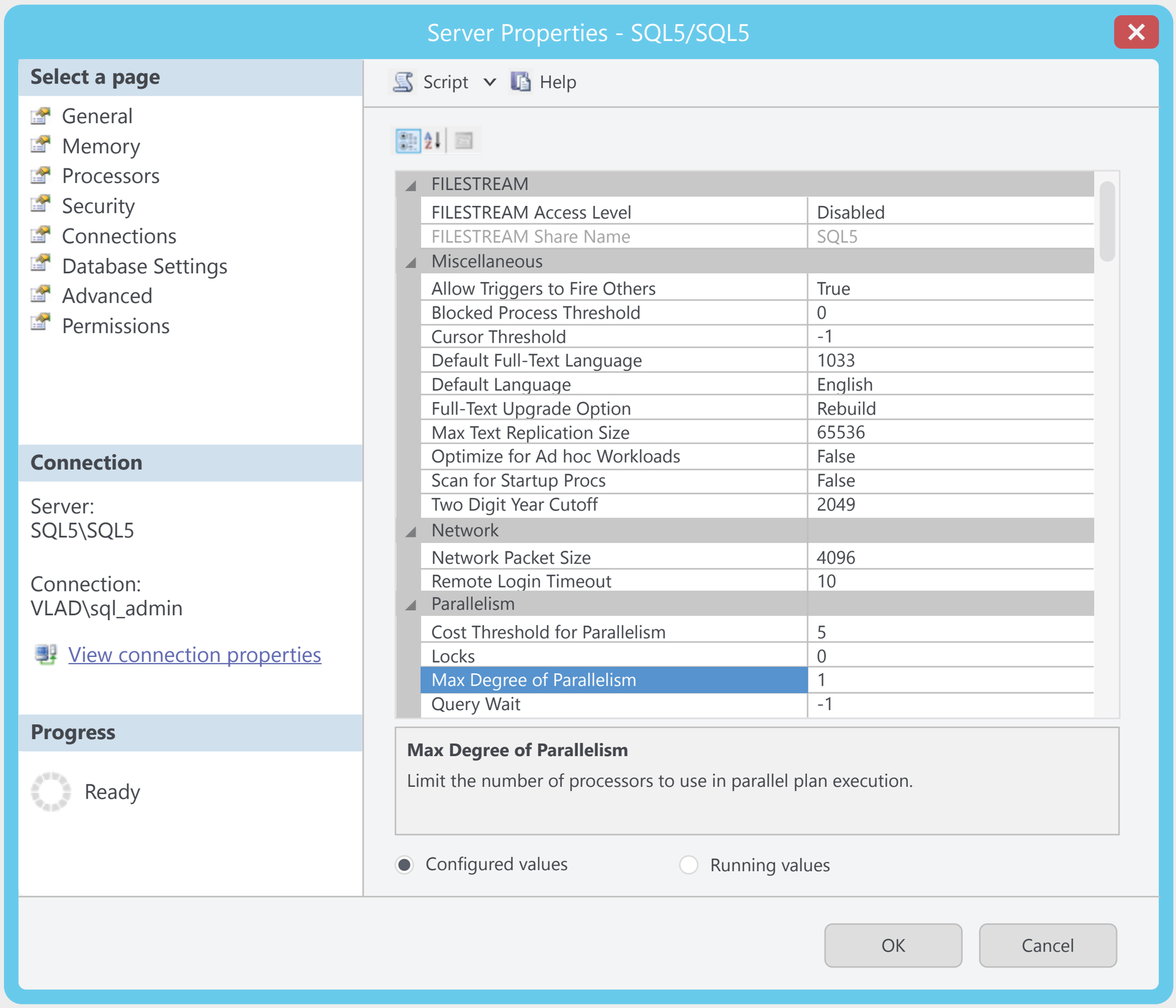
Auto-Create Statistics

Do not enable auto-create statistics on a server that hosts SQL Server and SharePoint Server. Enabling auto-create statistics isn't supported for SharePoint Server. SharePoint Server configures the required settings during provisioning and upgrade. Manually enabling auto-create statistics on a SharePoint Database can significantly change the execution plan of a query. The SharePoint Databases either use a stored procedure that maintains the statistics (proc_UpdateStatistics) or rely on SQL Server to do this.



Max Degree of Parallelism

This setting was optional in SharePoint 2010. However, in SharePoint 2013, you have to do it yourself or you won't be able to run the Configuration Wizard. Set max degree of parallelism (MAXDOP) to 1 for instances of SQL Server that host SharePoint Databases, to make sure that a single SQL Server process serves each request. Setting the max degree of parallelism to any other number can cause a less optimal query plan to be used that will decrease SharePoint Server 2013 performance.



Instant File Initialization

When SQL Server increases the size of a file, it must first initialize the new space before it can be used. This is a blocking operation that involves filling the new space with empty pages (zeroes). That means before SQL can create or auto grow, it must first write the size required with zeroes, and then it can save the data.

However, since SQL 2005 we can enable “Instant File Initialization”. It's a feature that's seemingly simple; it allows file allocation requests to skip zero initialization on creation. As a result, file allocation requests can occur instantly – no matter the file size.

Here are some tests done by Kimberly L., author at www.sqlskills.com:

Performance Test with Zero Initialization

Hardware: Dell Precision 670 Dual Proc (x64) with Dual Core, 4 GB Memory, RAID 1+0 array w/4-142 GB, 15000rpm disks

- CREATE DATABASE with 20 GB Data file = **14:02 minutes**
- ALTER DATABASE BY 10 GB = **7:01 minutes**
- RESTORE 30 GB DATABASE (EMPTY Backup) = **21:07 minutes**
- RESTORE 30 GB DATABASE (11GB Backup) = **38:28 minutes**

Performance Test with Instant Initialization

Hardware: Dell Precision 670 Dual Proc (x64) with Dual Core, 4 GB Memory, RAID 1+0 array w/4-142 GB, 15000rpm disks

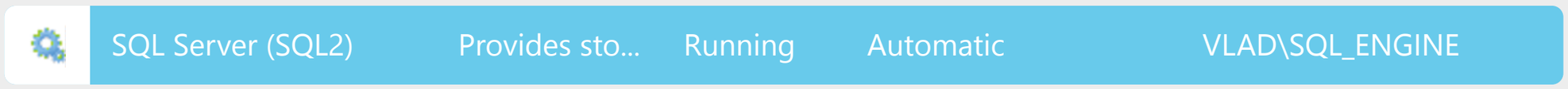
- CREATE DATABASE with 20 GB Data file = **1.3 seconds**
- ALTER DATABASE BY 10 GB = **.4 seconds**
- RESTORE 30 GB DATABASE (EMPTY Backup) = **5 seconds**
- RESTORE 30 GB DATABASE (11GB Backup) = **19:42 minutes**

However, Instant file Initialization doesn't apply to Log Files, and that's why we still have to set a decent amount as initial size!

As you can see, the differences in time are HUGE, but how do we enable this? It's super easy.

Instant file initialization is only available if the SQL Server (MSSQLSERVER) service account has been granted SE_MANAGE_VOLUME_NAME. Members of the Windows Administrator group have this right and can grant it to other users by adding them to the Perform Volume Maintenance Tasks security policy. For more information about assigning user rights, see the Windows documentation.

1. Open Services.msc and check what service account is running your SQL Server (Instance Name) service.



2. Open Local Security Policy> Local Policies > User Rights Management and go to the Perform Volume Maintenance Task. By default, only Local Admins have access to it. If the service account above isn't in your local admins (and it shouldn't be), add it.

3. You will then need to restart the SQL Server service in order for the changes to take effect.

Tempdb

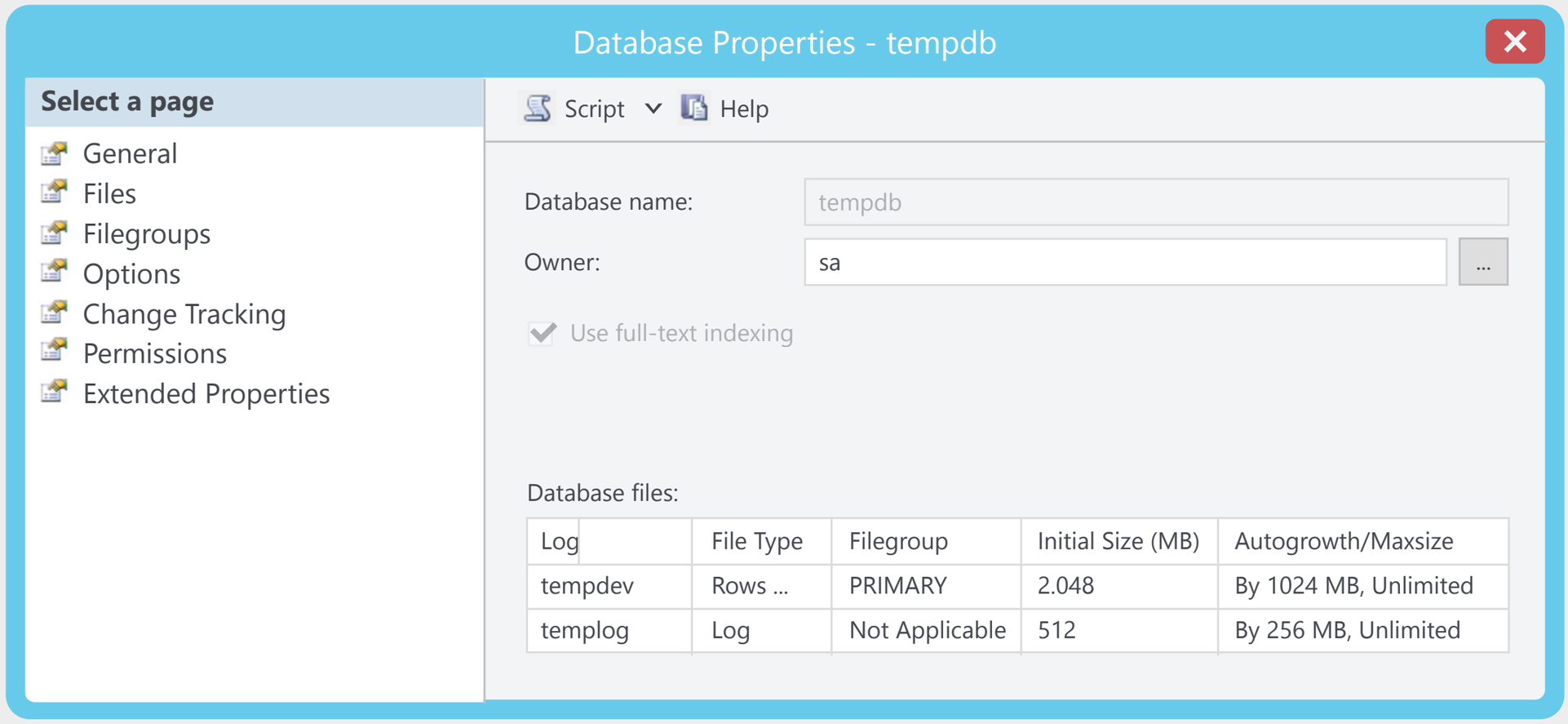
The **tempdb** system database (found in SQL5/Databases/System Databases/tempbd) is a global resource that's available to all users connected to the instance of SQL Server and is used to hold the following:

- Temporary user objects that are explicitly created, such as: global or local temporary tables, temporary stored procedures, table variables, or cursors.
- Internal objects that are created by the SQL Server Database Engine, for example, work tables to store intermediate results for spools or sorting.
- Row versions that are generated by data modification transactions in a database that uses read-committed using row versioning isolation or snapshot isolation transactions.
- Row versions that are generated by data modification transactions for features, such as: online index operations, Multiple Active Result Sets (MARS), and AFTER triggers.

Initial Size & Autogrow

The size of **tempdb** can affect the performance of a system. For example, if the **tempdb** size is too small, the system processing will be too occupied with autogrowing the database to support your workload requirement every time you start the SQL Server. You can avoid this overhead by increasing the size of **tempdb**.

This Database is the busiest database of your instance, so don't be shy to give it a good initial size. Here, we gave it a 2048MB Initial size and 1024MB autogrow. Thanks to Instant File initialization, the autogrow is done almost instantly and because it pretty high, the database can actually be busy writing/reading files instead of autogrowing!

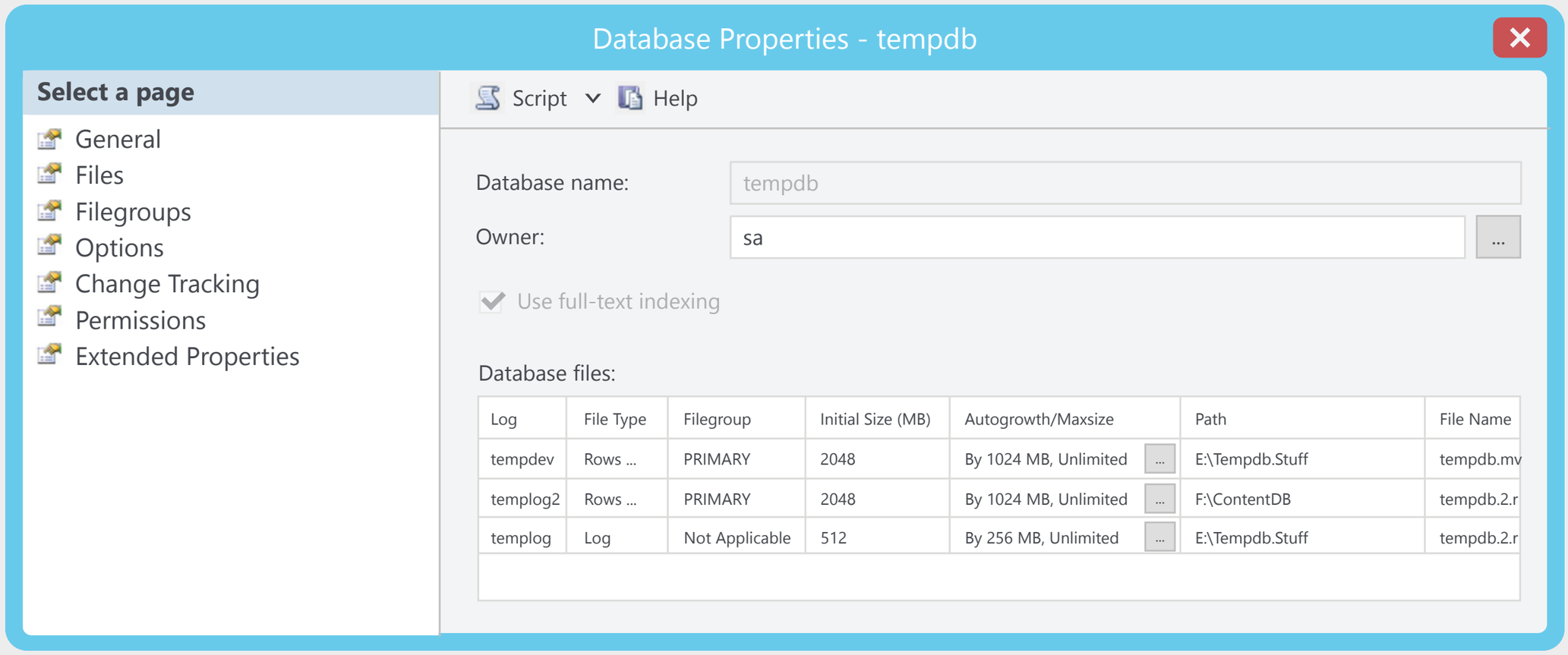


Multiple Files

By now, we know that the **tempdb** is the busiest database and needs to be on the fastest drive. But what if we create multiple files on multiple drives? That would allow the SQL Server to write in both files at the same time, therefore increasing performance. (Think of it as a TempDB Raid 0).

The general recommendation is 1 File/ CPU. Therefore, for a Dual Core CPU, you would have 2 Files. However, although Microsoft still suggests this, most SQL experts think you should only allocate 1 File per 2/4 CPU's.

Pro Tip: Make sure all your data files have the same initial size and Auto growth settings! This will allow optimal proportional-fill performance.



Other Considerations

- Avoid shrinking tempdb (or any database) files unless you are very certain you will never need the space again.

- Do not change collation from the SQL Server instance collation.

- Do not change the Database owner from sa.

- Do not drop the tempdb Database.

- Do not drop the guest user from the Database.

- Do not change the recovery model from SIMPLE.

- Ensure the disk drive's tempdb resides on have RAID protection i.e. 1, 1 + 0 or 5 in order to prevent a single disk failure from shutting down SQL Server. Keep in mind that if tempdb isn't available then SQL Server cannot operate.

Install SharePoint 2013

After all these optimizations, it's finally time to install SharePoint Server 2013!

CHAPTER **FOUR**

Maintenance Tasks

You have now installed SharePoint 2013 and your users are amazed by its speed. They're uploading, downloading documents, using SharePoint's social features every day, and they're happy!

Unfortunately, your job as a SharePoint Admin/DBA isn't over. Think of your SQL as a nice sports car, if you don't change the oil and maintain it properly, its performance will slowly degrade until it becomes slower than a scooter.

However, because we planned everything correctly and all the Post Installation Configurations were done right, we won't need to do this too often.

The SharePoint Databases Autogrow “Feature”

Remember when, in the Post Installation Configurations, we modified our model database and changed the autogrow settings? Well, when you create a SharePoint Content Database or Service Application Database, **SharePoint 2013 (and 2010) only copies the Initial size setting, but not the autogrow**.

However, since the initial size of our Databases was set to accommodate one year of content, you won’t have to modify the setting every time you create a Database, but try to do it monthly. You might ask, why aren’t the SharePoint Databases also copying the autogrow settings? Most SharePoint professionals would agree it’s a bug. However, if you would ask a developer, I’m sure they would answer this:

Now you might wonder if you'll have to check every Database's autogrow settings on a monthly basis because you don't remember which ones you changed last month. Good news! Greg Larsen from SimpleTalk posted a very cool script that shows all the Databases that had the default autogrow settings! Here it is:

```
-- Drop temporary table if it exists
IF OBJECT_ID('tempdb..#info') IS NOT NULL
       DROP TABLE #info;
-- Create table to house database file information
CREATE TABLE #info (
     databasename VARCHAR(128)
     ,name VARCHAR(128)
    ,fileid INT
    ,filename VARCHAR(1000)
    ,filegroup VARCHAR(128)
    ,size VARCHAR(25)
    ,maxsize VARCHAR(25)
    ,growth VARCHAR(25)
    ,usage VARCHAR(25));
-- Get database file information for each database
SET NOCOUNT ON;
INSERT INTO #info
EXEC sp_MSforeachdb 'use ?
select ''?'',name,  fileid, filename,
filegroup = filegroup_name(groupid),
''size'' = convert(nvarchar(15), convert (bigint, size) * 8) + N'' KB'',
''maxsize'' = (case maxsize when -1 then N''Unlimited''
else
convert(nvarchar(15), convert (bigint, maxsize) * 8) + N'' KB'' end),
''growth'' = (case status & 0x100000 when 0x100000 then
convert(nvarchar(15), growth) + N''%''
else
convert(nvarchar(15), convert (bigint, growth) * 8) + N'' KB'' end),
''usage'' = (case status & 0x40 when 0x40 then ''log only'' else ''data only''
end)
from sysfiles';
-- Identify database files that use default auto-grow properties
SELECT databasename AS [Database Name]
      ,name AS [Logical Name]
      ,filename AS [Physical File Name]
      ,growth AS [Auto-grow Setting] FROM #info
WHERE (usage = 'data only' AND growth = '1024 KB')
   OR (usage = 'log only' AND growth = '10%')
ORDER BY databasename
-- get rid of temp table
DROP TABLE #info;
```

Jesper M. Christensen

> Did you know that you can check the autogrow settings directly from your Windows 7/8 PC? I've written a nifty post explaining how it's done!

The Autogrow, Your insurance Policy

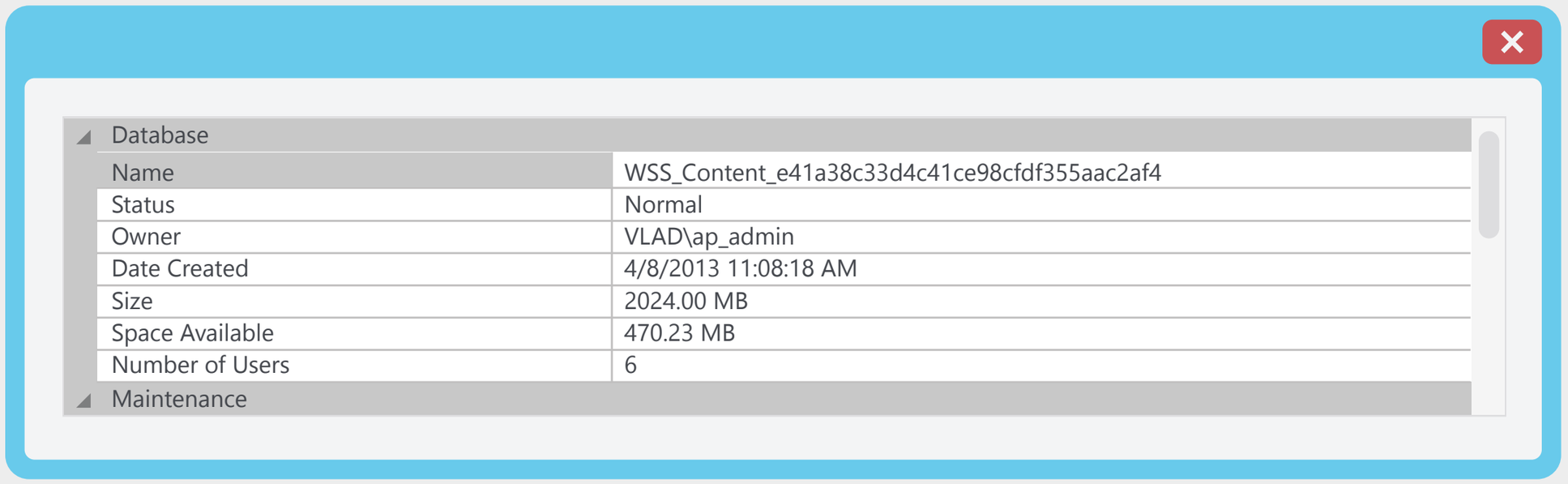
Now, we've talked about Autogrow and, hopefully, you understand how important it is to have it set correctly. Hopefully, you'll never have a need to autogrow your Databases! Yes, you read it right. Autogrow isn't a permanent solution, it's simply an insurance policy if your Databases reach your Initial Size.

If you have autogrow turned off, and your SharePoint Content Database reaches initial size you set, SQL will make it read only. All the settings were to make sure that even if your Content DB reaches initial size, your SQL will continue to operate fast by autogrowing a big enough chunk to last a while.

But how do we prevent our autogrow from kicking in and saving us? We have to pre-size the Databases. Remember, I told you to make your initial size big enough to accommodate how much data you'll have in one year. I could have said 1 month, 6 months, but I suggested one year so you won't have to check if you need to pre-size your Databases too often.

If planned successfully, you'll only need to do this once per year. However, checking it every 2-3 months will make sure nothing unexpected happens. Let's take a look at how to pre-size a database!

In SQL Manager, select a content database and check out the properties



We see that this Database takes 2024MB on disk, and it has 470MB free. That automatically tells us that during the year, it had to autogrow twice (1024 Initial + 500 + 500 autogrow). It means that, in one year, the content Database actually got to ~ 1.5GB instead of the 1GB I originally planned.

So, for the next year, we'll prepare for an additional 2GB of content. Go to the Files Tab, and change the Initial Size from 2024MB to 3072MB and click OK.

Shrinking the log files

The Database transaction log files seem to keep on growing - especially if you don't back these up. That's because every transaction is put in these, and not "flushed" until you perform a backup. When you perform a backup the file gets "emptied", and is ready for new transactions. Performing a manual Shrink creates fragmentation in the Database, and this decreases performance significantly. Creating a good maintenance plan is advised!

Works Cited

- Best practices for SQL Server in a SharePoint Server farm.
- Capacity management and sizing overview for SharePoint Server 2013.
- Database maintenance for SharePoint 2010 Products.
- Hardware and software requirements for SharePoint 2013.
- Instant Initialization - What, Why and How? - Kimberly L. Tripp.
- SQL Collation settings for SharePoint.
- SQL Server Database Growth and Autogrowth Settings.
- Storage and SQL Server capacity planning and configuration.
- Supportability regarding SQL collation for SharePoint Databases and TempDB.
- Tempdb Configuration Best Practices in SQL Server.
- Tempdb Database.
- Tuning SQL Server 2012 for SharePoint 2013 Jump Start.
- You’re Doing it Wrong: 5 Factors That Affect Database Performance.

About Sharegate

Sharegate simplifies management tasks for SharePoint, Office 365, and OneDrive for Business for thousands of administrators and IT professionals around the world. A privately-held company based in Montreal, Sharegate is trusted by more than 10,000 organizations.

As a leader in its industry, Sharegate lives by the motto:
“ Innovate and Keep Things Simple & Fun.”

Want to learn more?

Connect with us on twitter and visit share-gate.com for more SharePoint related content.

@sharegatetools

www.share-gate.com