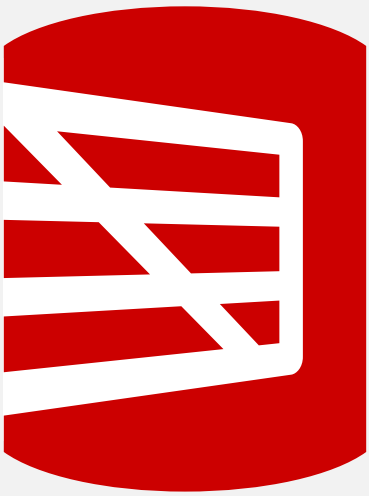
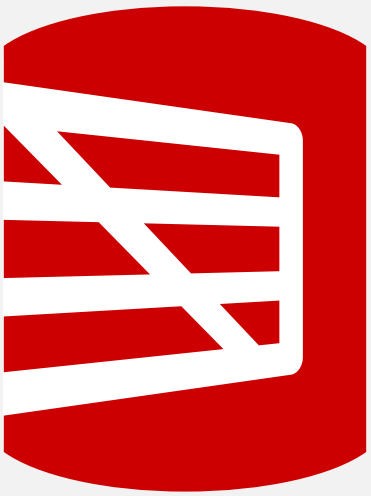


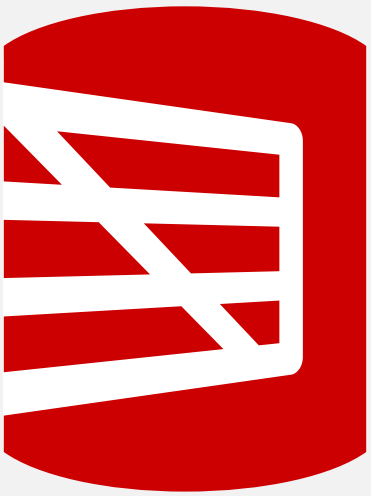
GUIDE

Best practices for SQL Provision

redgate



Contents



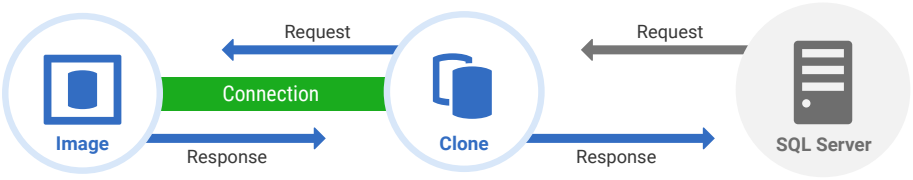
Introduction

In order to get the best out of SQL Provision, it's a good idea to think about how it would best fit into your workflows and network architecture. We've put together a set of guidelines which can help you to optimize efficiency, performance, security, and reliability, and recommend that you consider these when planning a proof-of-concept or rollout of SQL Provision in your organization.

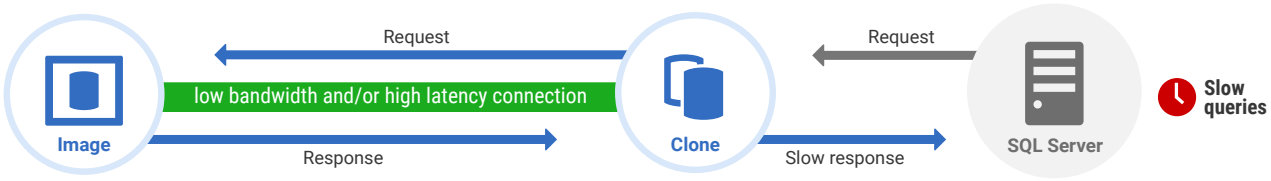
Network architecture

SQL Clone is designed to work within a LAN environment, using a single Windows server running SQL Clone Server, one or more file shares on which to store images (point-in-time database copies), and one or more machines running SQL Clone Agent, which allows you to create images and clones using installed copies of SQL Server on the same machine. You can either access SQL Clone Server from its web interface, or issue commands using PowerShell from any machine on which you've installed the cmdlets.

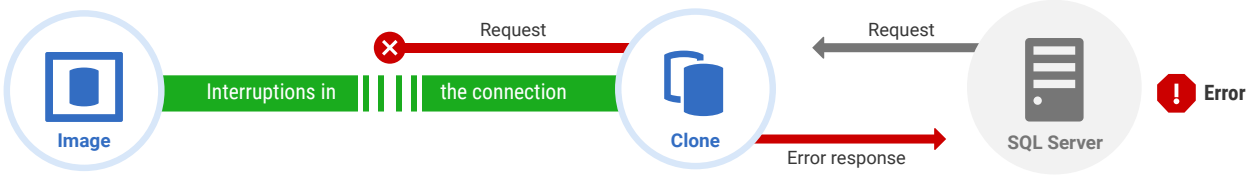
SQL Clone uses disk-based virtualization technology to allow many lightweight clone databases to be made from a single image. There are many benefits to this, particularly in how quickly you can create new databases and how they can access large data sets without having to consume additional disk space, but there are also some trade-offs associated with this technology which can be mitigated by using an appropriate network architecture.



When SQL Server accesses a clone database, the virtual disk driver first checks to see if the data it is requesting is available in the clone's local file, where changes are stored, and returns the data from there if so. Otherwise, the data is accessed from the image file. This means that the bandwidth, latency, and reliability of the connection between the clone and the image location is very important. Best performance and reliability can be obtained by hosting the image file share on a SAN on the same LAN as the machines which will be used to host clones.



Separating the image by a low bandwidth and/or high latency connection such as a VPN risks low clone performance, as whenever SQL Server attempts to read unmodified data, it must be transmitted across this connection. This will be exacerbated by usage patterns which cause SQL Server to read or modify large numbers of pages, such as table scans or updates affecting many rows.



Interruptions in the connection between the image and clone can prevent clones from being accessed until SQL Server is restarted. If SQL Server attempts to read data from the image, but the virtual disk driver cannot reach the image at that time, it returns an error to SQL Server, which then refuses to work with the database. SQL Clone attempts to detect this condition and detach the database, but it is not always possible to do so before SQL Server attempts access.

It is therefore very important that the connection between images and clones is not interrupted. This is best achieved by using a stable LAN connection between them. Depending on your network architecture, it may make sense to create a number of clones on central servers rather than distributing them across machines which are sometimes disconnected from the network or are separated from the image location by a VPN.

Workflows

SQL Provision permits efficient, compliant database provisioning for dev/test by using Data Masker and SQL Clone together.

Masking

Making a production database available for dev/test use usually requires masking sensitive data, altering permissions, and sometimes modifying static or config data. Using Data Masker, someone familiar with the sensitivity characteristics of the database can generate a masking set appropriate for a particular database, creating a set of rules which will replace personal or otherwise sensitive data with generated replacements. A masking set which takes into consideration the distribution of the data in the live system and the implicit relationships between columns and tables can be used to ensure the masked database behave similarly to the real production database, while being compliant with data protection regulations and best practices by not sharing sensitive data in dev/test environments. Masking sets should be stored in a location accessible by SQL Clone users who create images or where automated image-creation scripts will be run, and will need to be updated if the database schema changes.

Although Data Masker can connect to a production database in order to read its schema, testing the masking set is an important part of this process, and so it must be re-targeted to a temporary database in order to safely do this. Performance will be higher if the machine executing the masking set has a fast, low-latency connection to the SQL Server hosting the database. You may want to consider creating an image of the database using SQL Clone and testing the masking set against clone databases, as these can be quickly reset back to the original image state to test further changes.

Creating images

Creating clones takes very little time and initial disk space, but creating images requires more consideration as each image is a full point-in-time copy of the database. Images can be created from a live database (provided the server is not part of a cluster), which may be appropriate if you already maintain a pre-production environment which you are looking to create additional copies of. However, imaging from live should generally not be used directly against production, as this would require running the SQL Clone Agent service on your production system, and may result in temporary I/O interruptions and performance degradation during the imaging process. Instead, it is generally better to use backups when working from production, setting up a temporary instance within your dev/test environment to use for preparing images. As a SQL Server instance cannot work with database versions newer than itself, this temporary instance should be the same version of SQL Server as that used in your dev/test environment, or an older version.

Any changes which you want to be present in all clones, such as masking data, should be made during image creation. This means that they will only have to be run once, and the image will not be made available for cloning until the modifications are complete.

In many cases, it is useful to automate image creation using the SQL Clone PowerShell cmdlets, which can be used to execute a Data Masker masking set. PowerShell scripts can be scheduled to run overnight on a schedule appropriate to your project, using the Windows Task Scheduler, SQL Agent, or another scheduling tool.

Creating clones

Clone creation and recreation for shared environments can also be automated. Ad-hoc clone creation can be useful for particular developers, branches, or work items. This can be handled by a DBA in response to tickets, but SQL Clone can also be used to delegate this responsibility to developers by giving them appropriate permissions and either access to the SQL Clone web UI or scripts designed to create relevant clone databases.

If per-instance or per-environment configuration is required, users with image creation privileges can create clone templates which consist of a set of SQL scripts to run after clone creation.

Managing images and clones

Since clone databases depend on their image, images cannot be deleted while they have clones. Many users therefore find it useful to keep a rolling set of images available so that developers can continue to work against a particular clone until they've finished their current task, then get a fresh clone based on an up-to-date image to start the next piece of work.

Deleting of old images which no longer have clones can be automated using PowerShell. The SQL Clone web UI can be used to get an overall view of the images and clones in your environment, and more advanced workflows can be constructed by building on top of the PowerShell cmdlets.

Security

SQL Clone provides a web interface, by default on port 14145, which can be locked down to specific Windows users or Active Directory groups. Only users who are permitted to manage users, licensing, and updates should be given Admin privileges. Regular users who are allowed to create images and clones can be given Standard user permissions, while any users who are only permitted to consume available images to create clones can be granted Clone only privileges. This final permission level can be used to permit developers to manage their own clones but not to create custom images, thereby ensuring that any required modification scripts and masking sets have been run correctly on images they can use for development purposes.

To ensure security, we would also recommend setting up an HTTPS certificate on this port as described in our documentation.

The SQL Clone Server must also be accessible on port 14146 for SQL Clone Agents to connect. This port is always protected by SQL Clone's own HTTPS certificates.

The SQL Clone Server service user only needs to be able to access its own configuration database using Windows Authentication and listen on these ports. It does not require local administrative access. Because the SQL Clone Agent service needs to be able to mount disks and manage databases, it needs local administrative privileges and membership of the sysadmin fixed role on any local instances of SQL Server with which you want to create images or clones.

The location where images are stored must be accessible to all applicable SQL Clone Agent service users, and it is best practice to restrict this from other non-administrative users to ensure that images are only accessed through the SQL Clone system.

Using clone databases

In most ways, clone databases work just like normal databases. However, their performance and growth characteristics can be significantly affected by how they are used.

Performance

As clones use disk-based virtualization, their performance characteristics are influenced by how SQL Server accesses its data files and the underlying Microsoft virtual disk driver – data does not directly pass through the SQL Clone service. While clones should not be used to generate realistic performance data, most users find clone performance to be similar in most development scenarios. However, operations which involve accessing or changing large number of pages will result in a large number of requests to the virtual disk driver, many of which may need to be served over the network. Index scans of a large table or updating many rows may therefore be noticeably slower than with a local database.

As changes are made to the clone database, they are saved locally, and can therefore be accessed without making requests across the network. This means that if clones are used for longer, although they will use more disk space, they will also tend to have better performance.

Given that your network setup and the how SQL Server responds to queries affect the performance of clone databases, it is best to test using typical workloads in your environment.

Size

When clones are initially created, the clone data files themselves contain very little data because unmodified data is stored in the image. As changes are made to the database, the clone data files will grow. Any processes, either automated or manual, which cause writes to the database files can potentially cause clone growth.

If you create a clone which was originally created from an older version of SQL Server onto a newer instance, SQL Server will upgrade the database version, causing writes which will increase the size of the clone. This can be avoided by using the same version of SQL Server for image and clone creation. Initial clone size (and time to create clones) may also be higher if there is a large log file.

Operations causing a large number of writes, particularly if they occur across many database pages, will cause clones to grow in size significantly as the changes must be saved locally. In particular, re-indexing clones should be avoided, and data masking should be performed on images rather than clones.

Generally, clones are best used for short-lived development tasks or temporary environments which are regularly refreshed, and should be regularly replaced rather than maintained over long periods of time.

Maintenance

SQL Clone uses a SQL Server database (by default called SqlClone_Config) for storing its state and history information. This database should be backed up regularly.

Redgate releases regular product updates for SQL Clone. Provided that the SQL Clone Server can communicate with update.red-gate.com, it will inform you when updates are available, and these can be installed from the web UI by administrators.

Agents automatically update when the SQL Clone Server they are attached to is updated.

.