WinDocks

SQL Server containers with in-container data

This article provides step-by-step instructions for working with SQL Server containers with databases running in the container's private file system ("in-container"). This method works well with databases up to 3-5 GB. For large databases, refer to SQL Server containers with database clones, where writable environments are delivered quickly with minimal storage.

Windocks supports two methods for working with SQL Server containers, using standard Docker client software or a web UI. This article provides a step by step guide to using both the web UI and the command line interface, with databases running in the container's private file system.

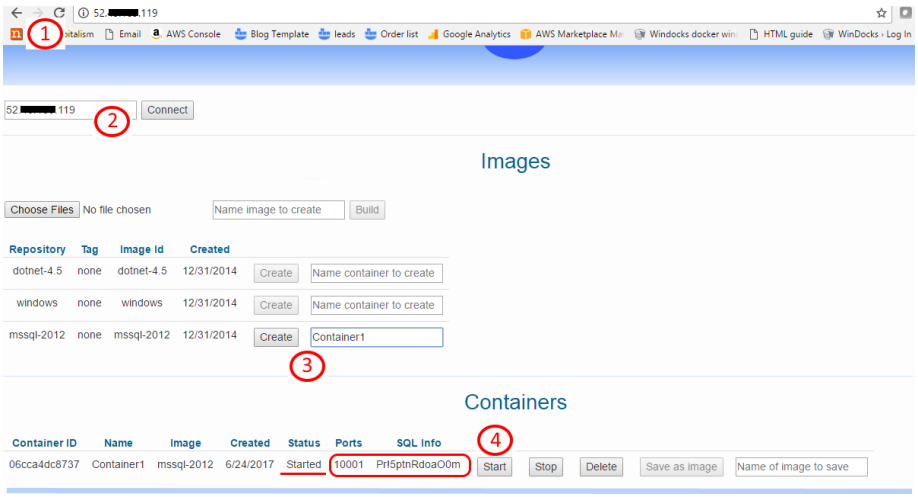
Start the Windocks daemon and web UI

Following the Windocks install, the Windocks daemon will prompt one-time for the Administrator credentials and will thereafter "run as Admin" for login users that belong to the local Administrative group. For machines that lack an Administrator account, or prefer to start the daemon manually, the desktop icon can be used by right-clicking and selecting "run as Admin." See **Windocks Installation and Configuration Guide** for more details on Windocks daemon operation.

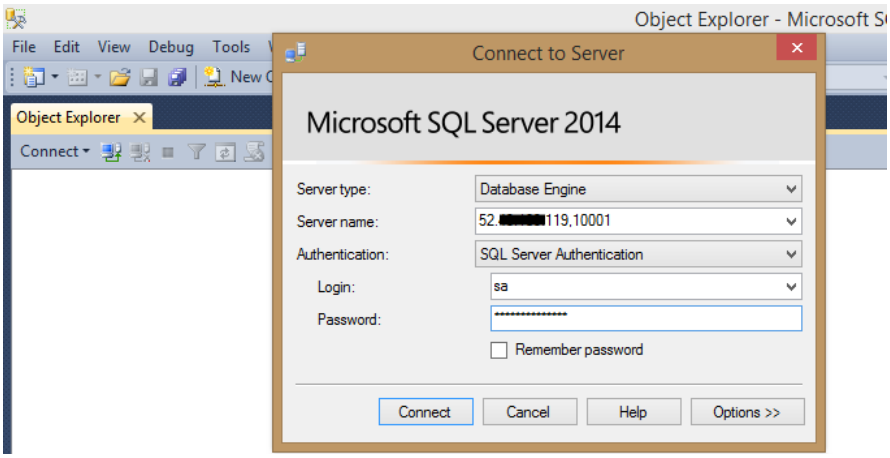
With the daemon running, open a Firefox or Chrome browser and direct the URL to localhost (IE is not supported). When the web UI resolves enter the local loopback address: 127.0.0.1. The Windocks web UI is now visible, and displays the images available (Steps 1 and 2 below). For access from a remote workstation enter the URL of the Windocks host IP address. Ensure the Windocks host firewall allows inbound traffic to ports 10001 to 10200 (container ports), 2375 for the Windocks daemon port, and SQL Server port of 1433. Enter the hosts IP address in the IP selection window to "connect."

Use the web UI to create a Custom SQL Server image from an empty container

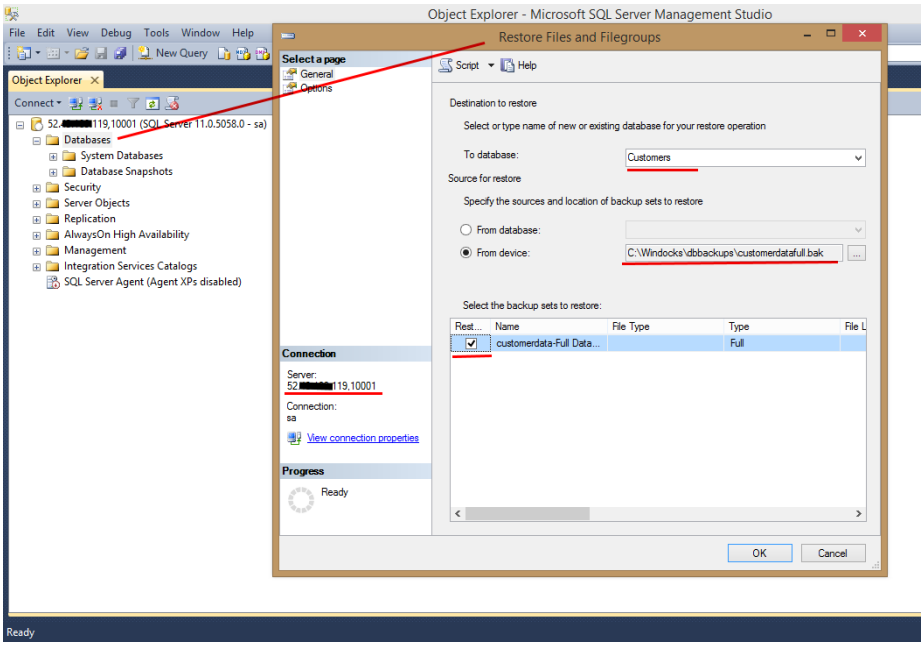
During installation Windocks configures an existing SQL Server instance for support of containers. Use the web UI to enter a name for a new SQL Server container, and click "create." When the container is delivered note the container port and click on "start." Steps 3 and 4 below.



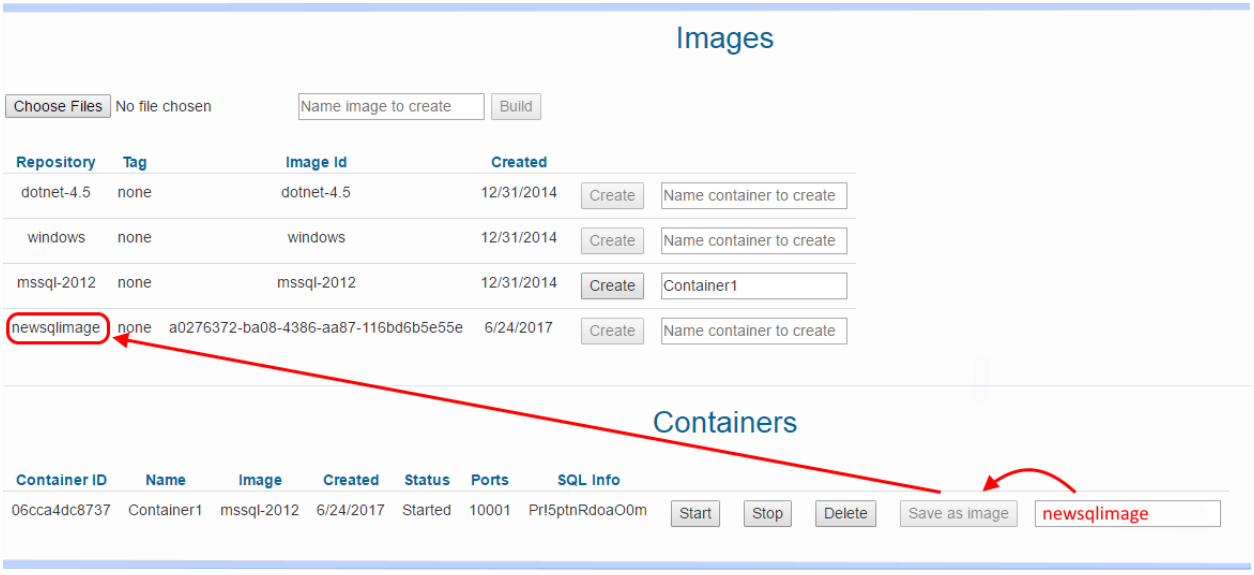
Open SQL management studio with the loopback address and port (127.0.0.1,10001 note a comma is used as a separator), or the remote host IP address. The SQL sa password is needed for remote access.



Configure the SQL Server instance to include the desired databases, by restoring a database backup, or copying or moving databases into the container file system and attaching. Each container has a private file system that can be inspected in the \windocks\containers folder. Sample SQL Server 2012 database backups are included in the \Windocks\dbbackups folder. In the example a Full backup of customerdatafull.bak is restored from \windocks\dbbackups. Each container has a private file system, located in the \windocks\containers directory.



Once the container is configured it can be committed to create a custom image. Enter a name in "image to save" and click "save as image." When the web page refreshes the new image is listed in the image section of the web page. A running container will be stopped momentarily while the image is created, but is returned to a running state once the image is complete (a stopped container can be committed, and is returned in stopped state).

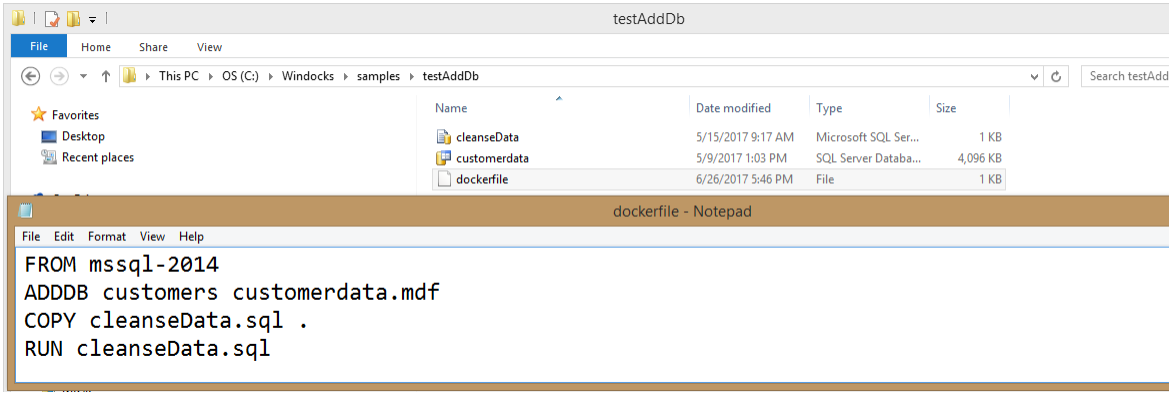


The new image now supports delivery of scores of identical environments in seconds.

WinDocks

Creating a custom SQL Server image using a Dockerfile

Custom images are also built with a Docker configuration file (Dockerfile). The Dockerfile can include databases and scripts that are copied into the container and image during a build. A sample Dockerfile is located in \windocks\samples\testadddb, is shown below.

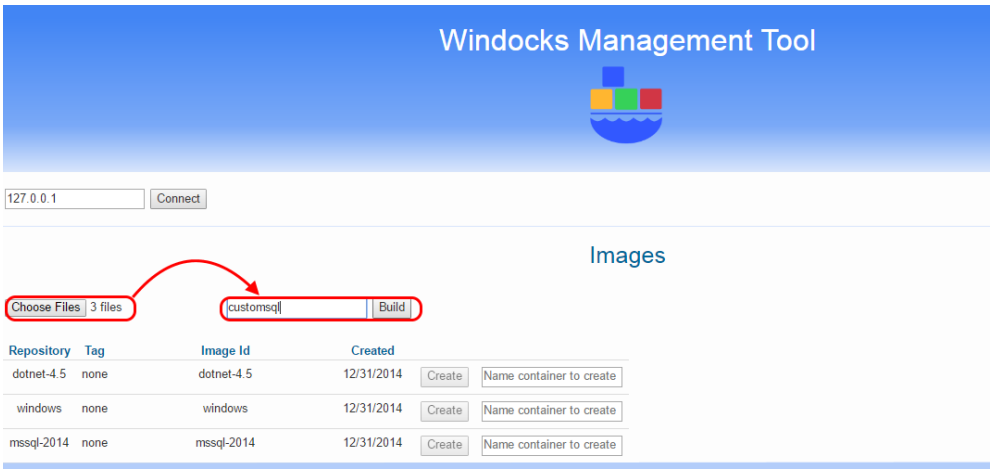


Dockerfiles begin by referencing a base image (**FROM mssql-2014**). Additional commands include **ADDDB**, which copies databases into the container. SQL scripts can be added with a **COPY**, and are

RUN during build. Dockerfiles support building of containers and images with scores of databases, each described with a separate line.

The Dockerfile and all files and folders located in the same directory are copied to the host for processing, so ensure the databases are located in the same directory as the Dockerfile. To build an image, click on the “select files” tool and navigate to the Docker file and associated files. Highlight the files, right-click and “select.” Enter an image name and “build.”

Note: at the time of this article this step generatea an error, which will be resolved shortly.



The build using the file selector tool delivers a new image, but not a container. The image includes the databases and scripts that were run during the build. The image can be used to generate as many identical instance as needed. To use the new image, simply enter a container name, and select "create." The container will appear at the bottom of the container list. Select "start" to make the container available. The SQL sa password is presented for network access.

Working with the Command Line Interface

Windocks is an independent port of Docker's open source to Windows, and uses standard Docker client software. The Windocks installation includes a Docker.exe that can be copied to any Windows client, and can be used locally on the Windocks host. Docker clients for mac and linux machines are also available, see **Additional Resources** at the end of this article for more information.

The docker client on the Windocks host is included in the system path, so is available via a standard command prompt window or PowerShell on any directory path. For client machines either setup the system path, or navigate to the directory where the Docker.exe is located to work with the following commands.

The Windocks CLI supports a standard set of Docker commands. When working from a remote client, the syntax takes on an added complexity, and requires the host firewall to be configured to allow inbound traffic on ports 10000 – 10200, 2375, and SQL Server port 1433.

>docker -H tcp://ip.address.of.host:2375 images Remote client call for >docker images

Docker commands used on the Windocks host use the simpler syntax shown below:

>docker build -t <imagename> <directory>	Builds an image only. This command supports SQL Server images with cloned databases.
>docker build <directory>	Builds an image and container, with the image named with the container name. Does not support building an image with cloned databases.
>docker commit <option> <instruction><containerid> <imagename>	Commits a container to create a new image. Does not support creating images with cloned databases.
>docker create <option> <image>	Creates a new stopped container.
>docker exec <containerid> <command>	Executes a command in a container.
>docker images	Lists available images on the host.

>docker ps	Lists containers on the host.
>docker rm <containerd>	Removes the containers. Use 2-3 digits, sufficient to achieve a unique match to the full container id.
>docker rmi <imagename>	Removes the image. Use the full image name.
>docker run -d <option> <image>	Creates a running container.
>docker start <containerid>	Starts the container
>docker stop <containerid>	Stops the container

Containers created can include assigned ports and names. A commit can include a Dockerfile instruction to **ADD, COPY, or RUN**, with the **--change** option.

-p <port>

--name <containername>

--change "instruction file"

--cidfile=<path to folder>

DockerFiles:

Dockerfiles are plain text configuration files that support the creation of new containers and images. A number of examples are included in the \windocks\samples directory.

The Docker client will copy all files and folders that are in the Dockerfile location to the host. It is important to include only files that are desired in the container or image to be located with the Dockerfile.

Supported Dockerfile commands include:

FROM<image>

ADD <file>

ADDDB <dbname> <mdf> <ndf> <ldf>

COPY <file>

EXPOSE <port>

RUN <file>

SETUPCLONING FULL <dbname> <pathto Full backup>

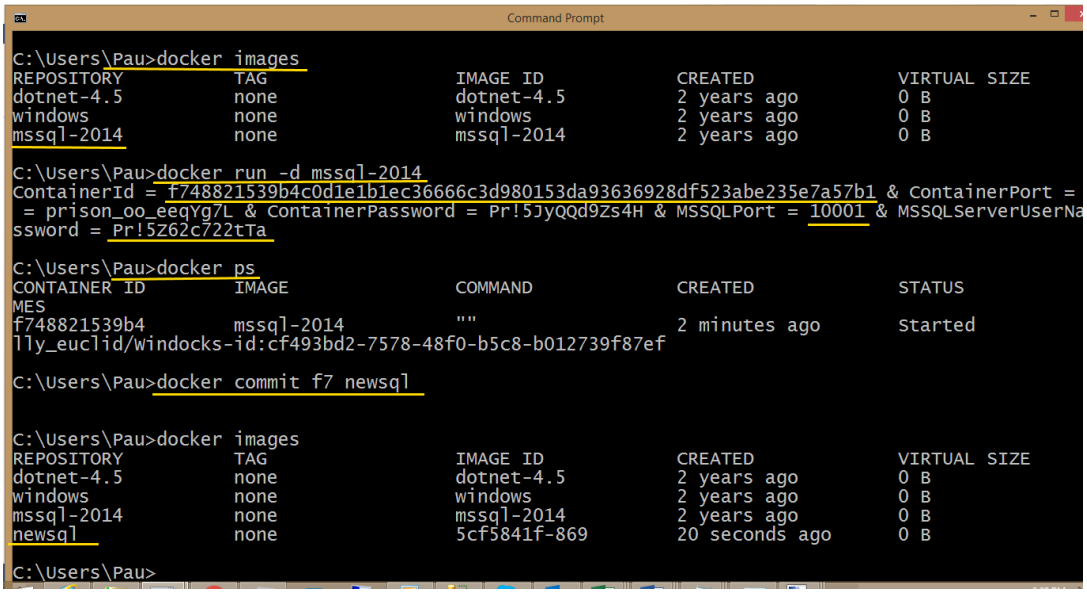
SETUPCLONING DIFF <dbname> <pathtoDifferential backup>

SETUPCCLONING RAW <dbname> <pathto DB files>

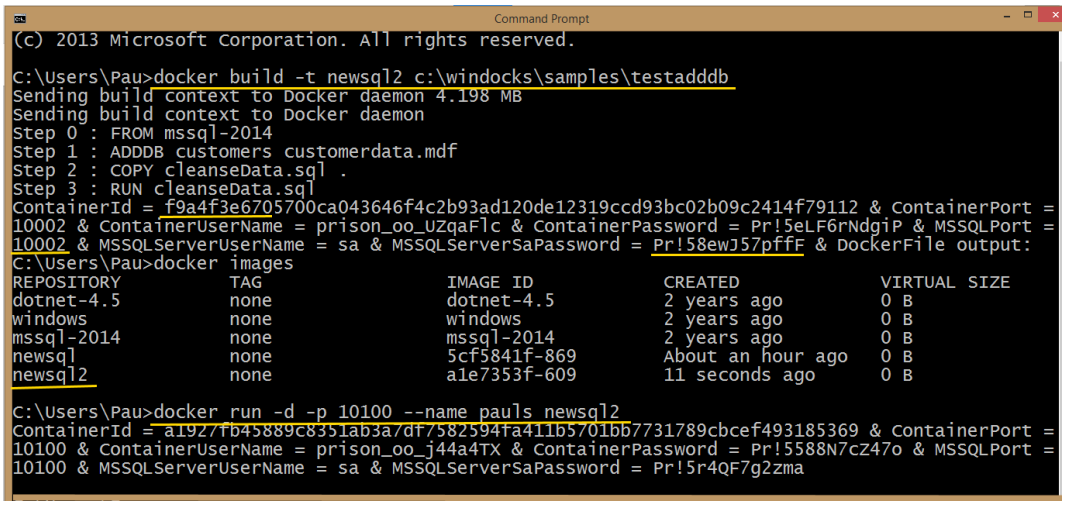
Note, **SETUPCLONING** instructions must be used with the **>docker build –t** command.

Database files referenced by the ADDDB should be located in the same directory as the Dockerfile.

The image illustrates use of docker client commands. A running container can be configured and committed to create a new image. When working with containers a unique match of 2-3 digits of the ID is generally sufficient. When working with images the exact name is required, and images are case sensitive. The process below creates a running container, which is committed to create the **newsql** image. The images command confirms the availability of the new image.



Images are also created by building a Dockerfile. A docker build delivers a container and new image. The “-t” parameter requires an assigned image name (newsql2). As before, we use a folder with a Dockerfile and any scripts or databases that need to be copied to the container.



The new image is confirmed with the **>docker images** command, and then used to deliver identical container environments. In the example the parameters **-p** and **--name** are used to assign a port and new container name.

Pros and Cons for SQL Server containers with in-container databases

The processes outlined are ideal for delivery of Dev and Test environments comprised of databases up to ~5 GB. Each container is delivered in seconds, varying on the size of the databases to be copied, and maintenance is easy as removal of containers. A team can work with scores of identical environments on a single shared server, and save hours in VM maintenance (and on costs of VM infrastructure). Containers are created quickly, so this workflow is ideal for short-lived instances needed for Dev and Test. The web UI provides Developers an easy to use self-service environment.

The negative of in-container databases is that data does not persist beyond the container instance. Each container and image is a full byte copy, so storage is consumed as instance count grows.

Additional Resources and Notes:

1) SQL Server containers and database clones
2) Installing and Configuring Windocks
3) Windocks containers operate with DNS: https://windocks.com/blog-2/docker-windows-containers-and-DNS
4) To understand Windocks licensing options for organizations: https://windocks.com/files/WinDocks_Licensing_and_Support.pdf
5) Windocks + Jenkins CI pipeline support Email: info@windocks.com
6) For information on working with multi-tier environments, including .NET see: https://windocks.com/lps/gitbuildtest
7) For technical support email: support@windocks.com
8) For information on how Windocks compares to Microsoft's new containers in Windows Server 2016: https://windocks.com/blog-2/Windows-Containers-Compared-Windocks-Microsoft