

## 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 <u>SQL Server containers with database clones</u>, 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.

# 🛄 WinDocks

← → C 0 52	iole 👌 Blog Template 👌 leads 🍃 Order list 🚽 G	ioogle Analytics 🍵 AWS Marketplace Ma	🞯 Windocks docker win:	☆ 🖸 Th HTML guide 🞯 WinDocks > Log In
52 Connect				
		Images		
Choose Files No file chosen	Name image to create Build			
Repository         Tag         Image Id         Cre           dotnet-4.5         none         dotnet-4.5         12/31	ated I/2014 Create Name container to create			
mssql-2012 none mssql-2012 12/31	I/2014         Create         Name container to create           I/2014         Create         Container1			
	3	Containers		
Container ID Name Image 06cca4dc8737 Container1 mssql-201	Created Status Ports SQL Info 2 6/24/2017 Started 10001 Prl5ptnRdoaOC	Om Start Stop Delete	Save as image N	ame of image to save

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.

<b>9</b>		Objec	st Explorer - Microsoft S
File Edit View Debug Tools \	s. <sup>j</sup>	Connect to Server	×
🗄 🛅 🕶 🔤 🖝 💕 📕 🎒 🔔 New C			
Object Explorer ×	Microsoft SQI	Server 2014	
Connect 🕶 🛃 🔳 🍸 🛃 📓			
	Server type:	Database Engine	~
	Server name:	52.404001119,10001	~
	Authentication:	SQL Server Authentication	~
	Login:	sa	~
	Password:	*****	
		Remember password	
	Connect	Cancel Help	Options >>

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.



<del>5</del>	(	Object Explorer - M	licrosoft SQL	Server Manage	ment Studio	
File Edit View Debug Tools Window Help	-	Restore	e Files and Fi	legroups		- 🗆 🗙
Object Explorer ×	Select a page	🔄 Script 👻 🎼 Help				
Connect 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	Cheeris	Destination to restore Select or type nam To database: Source for restore Specify the source From database: () From device:	ne of new or existi es and location of	ng database for your n Customers backup sets to restore C:\Windocks\dbb	estore operation	✓ fullbak
		Select the backup	sets to restore:			
	Connection Server: 22-minit 19,1001 Connection: 32 View connection properties Progress Ready	RedName customerd	lata-Full Data	File Type	Type Full OK	Fie L
Ready						

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).

						Image	S				
Choose Files	No file choser	Nam	ne image to creat	e Build							
Repository	Тад	Image	Id	Created							
dotnet-4.5	none	dotnet-4	4.5	12/31/2014	4 Create	Name contain	er to create				
windows	none	window	ws	12/31/2014	4 Create	Name contain	er to create				
mssql-2012	none	mssql-2	012	12/31/2014	4 Create	Container1					
newsqlimage	none a0276	372-ba08-4386-a	a87-116bd6b5e5	5e 6/24/2017	Create	Name contain	er to create				
						Contain	ers				
Container ID	Name	lmage C	Created Statu	s Ports	SQL Info						
06cca4dc8737	Container1	mssql-2012 6/2	24/2017 Starte	d 10001 Pr!5	õptnRdoaO0m	Start	Stop	elete	Save as image	newsqlimage	

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



# 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.

👪   🕞 🗓 ≂   testAddDb							
File Home Share View							
(€) → ↑ ↓ This PC → OS (C:) → Windocks → samples →	testAddDb				v 0	Search testAdd	
☆ Favorites	Name	Date modified	Туре	Size			
🛄 Desktop	📄 cleanseData	5/15/2017 9:17 AM	Microsoft SQL Ser	1 KB			
💹 Recent places	📴 customerdata	5/9/2017 1:03 PM	SQL Server Databa	4,096 KB			
	dockerfile	6/26/2017 5:46 PM	File	1 KB			
		dockerfile - Notepad					
File Edit Format View Help							
FROM mssql-2014 ADDDB customers customerdata.me COPY cleanseData.sql . RUN cleanseData.sql	df						

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."

				Windocks Management Tool
127.0.0.1	C	onnect		
Choose Files	3 files	customsql	Build	Images
Repository	Tag	Image Id	Created	
dotnet-4.5	none	dotnet-4.5	12/31/2014	Create Name container to create
windows	none	windows	12/31/2014	Create Name container to create
mssql-2014	none	mssql-2014	12/31/2014	Create Name container to create

#### 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></directory></imagename>	Builds an image only. This command supports SQL Server images with cloned databases.
>docker build <directory></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><contai< th=""><th>nerid&gt; <imagename></imagename></th></contai<></instruction></option>	nerid> <imagename></imagename>
	Commits a container to create a new image. Does not support creating images with cloned databases.
>docker create <option> <image/></option>	Creates a new stopped container.
>docker exec <containerid> <command/></containerid>	Executes a command in a container.
>docker images	Lists available images on the host.



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

<b>2</b>		Command Prompt		
C:\Users\Pau>docker REPOSITORY dotnet-4.5 windows mssql-2014	images TAG none none none	IMAGE ID dotnet-4.5 windows mssql-2014	CREATED 2 years ago 2 years ago 2 years ago	VIRTUAL SIZE O B O B O B
C:\Users\Pau>docker ContainerId = <u>f7488</u> = prison_oo_eeqYg71 ssword = <u>Pr!5Z62c722</u>	run -d mssql-2014 21539b4c0d1e1b1ec3660 & ContainerPassword 2tTa	66c3d980153da9363692 d = Pr!5ЈуQQd9Zs4H &	<u>8df523abe235e7a57b1</u> MSSQLPort = <u>10001</u> &	& ContainerPort = MSSQLServerUserNa
C:\Users\ <u>Pau&gt;docker</u> CONTAINER ID MES f748821539b4 lly_euclid/windocks-	<u>ps</u> IMAGE mssql-2014 -id:cf493bd2-7578-48	COMMAND "" f0-b5c8-b012739f87ef	CREATED 2 minutes ago	STATUS Started
C:\Users\Pau>docker	commit f7 newsql			
C:\Users\Pau>docker REPOSITORY dotnet-4.5 windows mssql-2014 newsql C:\Users\Pau>	images TAG none none none none	IMAGE ID dotnet-4.5 windows mssql-2014 5cf5841f-869	CREATED 2 years ago 2 years ago 2 years ago 20 seconds ago	VIRTUAL SIZE O B O B O B O B

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.



		Command Prompt		- 🗆 🗙			
(c) 2013 Microsoft (	Corporation. All rig	hts reserved.					
		\	- +				
C:\Users\Pau>docker	C:\Users\Pau>docker_buildt_hewsql2_c:\windocks\samples\testadddb						
Sending build contex	xt to bocker daemon 4	4.198 MB					
Sending build contex	xt to Docker daemon						
Step 0 : FROM mssql-	-2014	16					
Step 1 : ADDDB custo	omers customerdata.m	dŤ					
Step 2 : COPY cleans	seData.sql.						
Step 3 : RUN cleanse	eData.sql						
ContainerId = $19a4f$	<u>3e6/0</u> 5/00ca043646†4c	2b93ad120de12319ccd9	3bc02b09c2414f/9112	& ContainerPort =			
10002 & ContainerUse	erName = prison_oo_U	ZqaFIC & ContainerPa	ssword = Pr!5eLF6rNd	g1P & MSSQLPort =			
10002 & MSSQLServerl	JserName = sa & MSSQI	LServerSaPassword =	Pr!58ewJ5/pttF & Doc	kerFile output:			
C:\Users\Pau>docker	ımages						
REPOSITORY	TAG	IMAGE ID	CREATED	VIRTUAL SIZE			
dotnet-4.5	none	dotnet-4.5	2 years ago	0 В			
windows	none	windows	2 years ago	0 В			
mssql_2014	none	mssq1-2014	2 years ago	0 в			
newsql	none	5cf5841f-869	About an hour ago	0 В			
newsq12	none	a1e7353f-609	11 seconds ago	0 В			
C:\Users\Pau>docker run -d -p 10100name pauls newsql2							
ContainerId = a192/1	fb45889c8351ab3a7df7	582594†a411b5701bb77	31789cbcef493185369 <	& ContainerPort =			
10100 & ContainerUse	erName = prison_oo_j	44a4TX & ContainerPa	ssword = Pr!5588N7cZ	47o & MSSQLPort =			
10100 & MSSQLServeru	JserName = sa & MSSQ	LServerSaPassword =	Pr!5r4QF7g2zma				

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) <u>SQL Server containers and database clones</u>
- 2) Installing and Configuring Windocks
- Windocks containers operate with DNS: <u>https://windocks.com/blog-2/docker-windows-</u> <u>containers-and-DNS</u>
- To understand Windocks licensing options for organizations: <u>https://windocks.com/files/WinDocks\_Licensing\_and\_Support.pdf</u>
- 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: <a href="mailto:support@windocks.com">support@windocks.com</a>
- 8) For information on how Windocks compares to Microsoft's new containers in Windows Server 2016: <u>https://windocks.com/blog-2/Windows-Containers-Compared-Windocks-Microsoft</u>