SQL Clones for SQL Server 2017 Linux Containers

Windocks supports use of SQL Server database clones with Windocks SQL containers and instances. This document details the setup, and use of SQL Server database clones with SQL Server 2017 Linux containers.

Building SQL Server images with Dockerfiles

Dockerfiles are plain text configuration files that define the data source and target. Windocks 3.0 adds support for external storage arrays, or Windows file system database cloning. Windows based SQL Server images are built with Full or Differential backups, or database files, with each being a full byte copy of the data. Once created an image supports creation and delivery of clones in seconds with full read/write support, with each requiring less than 40 MB on delivery.

This Dockerfile defines a clonable image that delivers cloned databases to a new SQL Server 2017 container on a Linux host.

```
FROM mssql-2017
ENV USE_DOCKERFILE_TO_CREATE_CONTAINER=1
RUN TargetAttach_MSContainerSqlLinux MSDockerIp|192.168.XX.XXX:2375
MSSqlImageName|microsoft/mssql-server-linux
MSDockerClientPath|C:\docker\docker.exe MSContainerPort|$MSContainerPort
MSContainerSaPassword|$MSContainerSaPassword MSLinuxMountPathForMountDb|None
MSLinuxMountPathForSetupCloning|/windocks/dataexternal/$ContainerId/$ContainerImageName
SETUPCLONING FULL customers C:\windocks\dbbackups\customerdatafull.bak
```

ENV USE_DOCKERFILE_TO_CREATE_CONTAINER ensures the Dockerfile is applied at run time for each container/environment. Delivery to SQL Server 2017 Linux containers is accomplished with **RUN TargetAttach_MSContainerSqlLinux** with the parameters shown. SQL Server clones are built with **SETUPCLONING**, with **FULL**, **DIFF** (differential) backups, or **RAW** database files.

Support for delivery of database clones over the network is based on SMB, with a file share created on the Windocks host mapped to the Linux host (c:\windocks\data to /windocks/dataexternal as shown above). The Linux setup involves installing SAMBA, creating the mapped folders, and setup of the Docker daemon to allow for remote commands. Finally, copy the desired Windows Docker client executable into a folder as called for in the Dockerfile (C:\docker\docker.exe).

A one-time build creates an image that supports an unlimited number of clones. The example below shows the build, followed by a command to deliver the clone to a new SQL 2017 container. Most of the

parameters are defined in the image, and only two are required for container creation, including the target port and SQL sa password.

```
>docker build –t <image> c:\windocks\samples\TestWindocksClonetoMSLinuxSQLContainer

> docker create -e $MSContainerPort="16000" -e $MSContainerSaPassword="Pa55word!!" <image>
```

Management of the combined environment is handled by the Windocks container. When it's time to refresh the environment the removal of the Windocks container removes the Linux container and associated mounts.

Working with sensitive credentials

Windocks 3.0 introduces encrypted credential support for Windocks images and containers. The workflow described above involves clear text SQL sa passwords, which is the current practice in use of SQL Server 2017 on Linux. When working with the Windocks SQL Server containers, credentials can be secured using the following methods:

1) Windocks containers support Windows authentication.
2) Windocks Windows SQL Server containers are created by cloning a SQL Server instance that is configured for use by Windocks. Each container inherits SQL logins configured on the parent instance, enabling users with these accounts.
3) Windocks also includes configurable SQL sa credentials for each created SQL container, including an option for no sa passwords, encrypted sa passwords, or passwords in clear text. The three options are configured in the Windocks config folder, node file as SHOW_SA_PASSWORD="0" or 1, or 2, for no password, encrypted, or clear text, respectively. Restart the Windocks Service following changes to the Windocks configuration.

Windocks also provides password encryption based on the Windows Data Protection API (DPAPI). To encrypt a password navigate to \Windocks\bin and open a command prompt and enter "encrypt." The program prompts for a credential string, and writes the hashed result to encrypted.txt in the same directory. Open the text file and copy the string into the Dockerfile that requires the particular password. For example, when working with storage arrays, a required parameter will include an array user and password:

ArrayPassword|1,0,0,0,208,140,157,223,1,21,209,17,140,122,0,192,79,194,151,235,1,0, . . .

Adding encrypted passwords into Dockerfiles allows them to be saved and reused securely. Once a credential is encrypted the hashed result or environment variable needs to be used in any references to that credential.

When configured to deliver encrypted credentials, Windocks SQL container sa passwords are delivered in standard Docker client return strings (image below). To unencrypt the credential copy the complete string and save as an encrypted.txt file. RDP to the Windocks server, and copy the encrypted.txt into the \windocks\bin directory. Open a command prompt and enter "decrypt."

```
C:\Users\windocks>docker create mssql-2017ContainerId =
b748ec44e5005197ba2fcac3936d63e1095eb90d369ab95889fce96b5ad2dd52 & ContainerPort =
10001 & ContainerUserName = prison_oo_02N0b32 & ContainerPassword = Pr!50hUil1gMma &
MSSQLPort = 10001 & MSSQLServerUserName = sa & MSSQLServerSaPassword =
1,0,0,0,208,140,157,223,1,21,209,17,140,122,0,192,79,194,151,235,1,0,0,0,74,38,15,17,101,94,19
7,70,144,140,230,233,116,122,95,115,4,0,0,0,2,0,0,0,0,0,16,102,0,0,0,1,0,0,32,0,0,0,204,17,196,23
,176,117,186,46,74,114,251,145,206,253,4,177,209,91,1,202,160,75,47,212,34,242,160,145,16,80,
211,154,0,0,0,0,14,128,0,0,0,2,0,0,32,0,0,0,59,46,53,25,86,50,150,70,22,76,116,157,147,34,15,52,
161,36,225,9,148,56,60,249,168,7,24,30,225,153,234,200,16,0,0,0,82,13,58,208,252,214,197,176,
215,156,227,89,84,176,82,180,64,0,0,0,31,200,214,65,59,122,87,223,14,50,212,104,115,74,133,53
,35,137,117,170,252,50,75,8,65,12,76,133,102,20,165,230,65,103,239,192,240,236,159,16,143,82,
253,225,38,178,11,90,122,6,5,196,190,108,235,188,74,133,117,230,241,207,90,83

C:\Users\Windocks>
```

The program decrypts the text file and presents the password:

```
Original bytes:
 80   114   33   53   114   49   89   56   118   112   84   54   71   99
Original password:
Pr!5r1Y8vpT6Gc
```

Working with a subset of databases

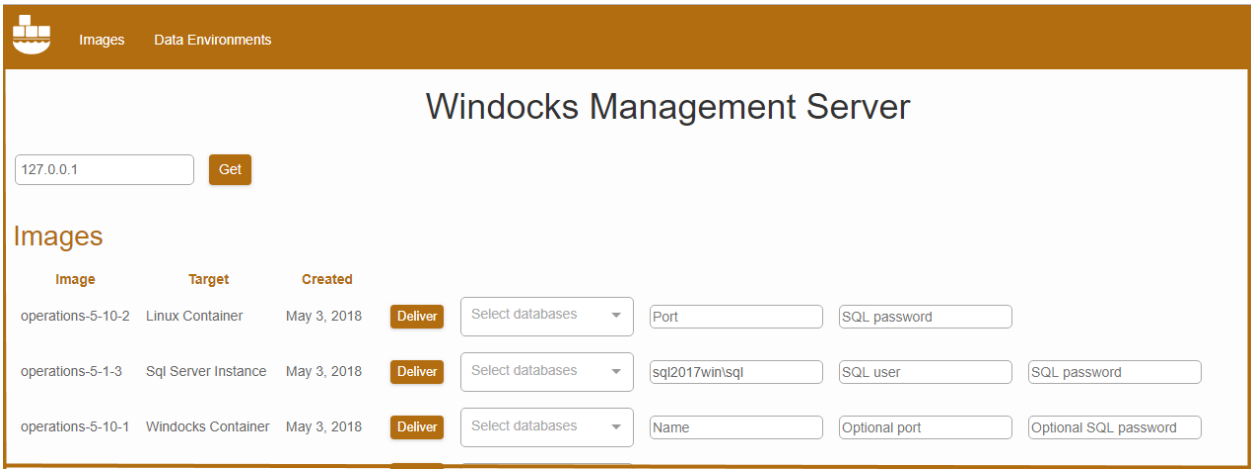
Users can work with a subset of the databases from an image by using a run time environment variable: **SQL_DB_NAME_OVERRIDES="dbname1, dbname2"**

>docker create -e SQL_DB_NAME_OVERRIDES="dbname1, dbname2" <imagename>

Working with a web UI

The Windocks web UI simplifies use for developers and other users. Open a Chrome or FireFox browser and point to the IP address of the Windocks server (local: 127.0.0.1). Images are displayed with required parameters, including the option to work with a subset of desired databases. The image

targeting Linux SQL containers only requires user input on the target port and SQL sa password, and includes a drop down selector for working with a subset of the databases in the image.



SQL Server cloning for SQL Server containers and instances

SQL Server 2017 Linux containers are drawing understandable attention in a world that is increasingly embracing Linux and open source technologies. Regardless of the form of SQL Server you use, database cloning is key to enabling an efficient workflow for development and test. The Windocks database cloning highlighted in this article will enable efficient upgrade testing, and work with large and complex data environments on the new SQL Server 2017 Linux containers.

Start to explore these capabilities today by downloading the free Windocks Community Edition, available at https://www.windocks.com/community-docker-windows