Working with a Pure Storage Arrays with SQL Server Containers and Instances

Windocks 3.0 introduces support for sourcing, managing, and delivering SQL Server data from storage arrays to any SQL Server target environment.   This article provides background on the design, and instructions for working with storage arrays.

Windocks 3.0 responds to a number of customer requests:

- Database administrators can create and manage a catalog of data images on the storage array
- Automated workflows and end-user access are supported through command line and the Windocks web UI
- Support for Microsoft SQL Server containers (both Windows and Linux), and conventional SQL Server instances are added, in addition to Windocks SQL Server containers
- Support for data migration from SAN to SAN, or from SQL instance to SAN is a new capability

Windocks runs on Windows 8.1 and 10 Pro and Enterprise editions, Windows Server 2012 R2, and Windows Server 2016.   All editions of SQL Server 2008 forward are supported, and Windocks continues to support creation of Windows based database clones based on Virtual Hard Drives (VHDs).

Enhanced Data Images and Dockerfiles:

Previous Windocks releases support containers built at run time with Dockerfiles.   Windocks 3.0 images can now include Dockerfiles that are applied at run time, that define data sources and targets. Dockerfile commands identify these images, their sources and targets, and their associated parameters.

Many parameters are defined in the Dockerfile during build time, such as the storage array IP address, volume name, and others that are fixed for the life of the image. Other parameters are set at run time with $ environment variables.

Windocks 3.0 is designed to accommodate a range of storage arrays.   For a complete listing of supported storage arrays contact Windocks at support@windocks.com.

Pure Storage array to SQL Server instance:

This Dockerfile builds an image, that clones a Pure array volume for mounting to a SQL Server instance on the LAN (or locally on the Windocks host).   SQL Server uses two accounts on the target instance, including a SQL user/password and a domain user, with both included in the SQL target instance **Sysadmin** group.

The final step for mounting SQL Server databases to network instances involves the remote machine Domain user executing commands to mount the databases. The Domain user is provided access to the network share created by Windocks, with permissions set either in the \windocks\config\node.conf file, or in the Dockerfile as shown below (domain\user is Windocks\Support1). Refer to Windocks Clones and SQL Server Instances for detailed setup of SQL Server instances.

Images are built with **>docker build -t**, and **ENV USE_DOCKERFILE_TO_CREATE_CONTAINER** ensures the Dockerfile is applied at run time. **RUN SourceClone_San_Pure** identifies the data source as a Pure Storage array, with the required parameters. **RUN TargetAttach_SQLWindows** identifies the target as a SQL Server instance on a Windows host, with the parameters needed for delivery. Images can include scores of databases, with each described with a separate **MOUNTDB** command and file path.

The Docker command and sample Dockerfile is included below, with parameters listed on separate lines for readability. Refer to **\windocks\samples\puretoinstance** for a sample that is ready for editing.

Note: when delivering Pure Storage array data to instances located on the Windocks host, the use of Network Shares is not needed. The parameters highlighted in blue below are only needed when supporting delivery to instances on the LAN.

>docker build -t <imagename> c:\windocks\samples\puretoinstance

```
FROM mssql-20XX
ENV USE_DOCKERFILE_TO_CREATE_CONTAINER
RUN SourceClone_San_Pure
        SourceVolume|SQL-DATA01 DestinationVolume|$ContainerId
        MountPoint|C:\windocks\$ContainerId HostName|c220m4-server
        ArrayIP|10.21.XXX.XX
        ArrayUser|username
        ArrayPassword|userpassword
        NetworkShareName|$ContainerId
        NetworkSharePath|C:\windocks\$ContainerId\data
        NetworkShareUsers|domain\User
RUN TargetAttach_SqlWindows
        InstanceName|$SqlInstanceName
        SqlUserName|$SqlInstanceUserName
        SqlPassword|$SqlInstancePassword
MOUNTDB customers c:\windocks\$ContainerId\data\customerdata.mdf
c:\windocks\$ContainerId\data\customerdata_log.ldf
MOUNTDB operations c:\windocks\$ContainerId\data\operations.mdf
c:\windocks\$ContainerId\data\operations_log.ldf
```

The image is used to deliver mounted environments through a **>docker create** command. Each "create" includes an associated "container" which represents the data environment that is mounted to the instance. Care must be taken to detach or delete the databases from the instance prior to refreshing with a new image. The Docker command line and parameters involved in delivering the image is summarized below.

```
>docker create -e $SqlInstanceName="Windocks1\SQL2016"  -e $SqlInstanceUserName="username" -e $SqlInstancePassword="Strongpassword" <imagename>
```

Windocks records the list of storage array volumes, snapshots, and mount points, and cleans up the resources on deletion of the "container."

Pure to Windocks SQL Server container

This Dockerfile builds an image that clones a Pure array volume, and mounts the databases to a new Windocks SQL Server container. In this case the target is on the Windocks host, simplifying the Dockerfile and does not require a **RUN TargetAttach.** The sample Dockerfile below is located in the \windocks\samples\puretowindocks folder.

```
>docker build -t <imagename> c:\windocks\samples\puretowindocks
```

```
FROM mssql-20XX
ENV USE_DOCKERFILE_TO_CREATE_CONTAINER
RUN SourceClone_San_Pure
        SourceVolume|SQL-DATA01
        DestinationVolume|$ContainerId
        MountPoint|C:\windocks\$ContainerId
        HostName|c220m4-server1
        ArrayIP|10.21.XXX.XX
        ArrayUser|pureuser
        ArrayPassword|pureuserpassword
MOUNTDB customers C:\windocks\$ContainerId\data\customers.mdf
C:\windocks\$ContainerId\data\customers_log.ldf
MOUNTDB operations C:\windocks\$ContainerId\data\operations.mdf
C:\windocks\$ContainerId\data\operations_log.ldf
```

Once the image is available, environments are delivered by using >docker create, followed by >docker start <containerid>, or by a >docker run –d <image>.

Work with a subset of the databases

Developers commonly choose to work with a subset of the databases included in a full data image. This is supported with a run time environment variable. In the case of delivering a Pure array image to Windocks containers, the command line takes the form:

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

When working with SQL Server instances there are times when users will choose to mount an updated database alongside older copies. Use the **DB_NAME_OVERRIDES** to add a second or third copy. To avoid database name collisions, the databases have the date/time stamp appended to the database name. This behavior is not implemented when Windocks delivers databases to containers, as Windocks only provides a one-time image delivery (so there is no risk of naming collisions).

Protecting the Pure Array Password

Dockerfiles require use the Pure Array password which should not be exposed as plain text. Windocks supports encrypted passwords for this purpose. Navigate to the \windocks\bin directory, open a command prompt, and enter "encrypt." Windocks encryption is based on the Windows Data Protection API (DPAPI), and prompts for the password. Once entered the encrypted password is written to an "encrypted.txt" file located in the same directory.

Open the encrypted.txt file, and copy the complete string into the Dockerfile for the ArrayPassword. Once encrypted all references to the password require the encrypted form.

ArrayPassword|1,2,0,0,0,56, . . .

Dockerfiles with the encrypted password can be safely used without compromising the security. The hashed string can also be decrypted by using the decrypt.exe program, so only trusted users should be provided access to the Windocks server when working with sensitive credentials.