DH2i DxEnterprise 21.0 Software: SQL Server Availability Groups for Kubernetes StatefulSets on Azure Quick Start Guide

DH2i Company

Support: +1 (800) 380-5405 ext. 2
https://dh2i.com/support/
eFax: +1 970-295-4505
Email: support@dh2i.com
https://dh2i.com

SQL Server for Kubernetes StatefulSet on Azure

This quick start guide describes how to create and deploy a DxEnterprise + SQL Server container image to an Azure Kubernetes Service (AKS) cluster for Availability Groups. Using this guide, the user will create a DxEnterprise + SQL Server container image and push the image to a repository, deploy and configure an AKS cluster and a StatefulSet, and use DxEnterprise to create and configure the Availability Group.

***NOTE:** This guide covers the deployment of three AG replicas. When deploying availability groups, please consider the Microsoft SQL Server quorum requirements for automatic failover described in this KB: Quorum Considerations for SQL Server Availability Groups.*

Prerequisites

- A machine with the Azure and Kubernetes CLIs installed and logged into an Azure account. For information about installing the Azure CLI and logging in, view Microsoft documentation. To install the Kubernetes CLI, run the command `az aks install-cli`. Alternatively, the Azure Cloud Shell may be used.
- Docker installed on a Linux VM and logged in to a Docker account with push and pull privileges. This can be the same VM that has the Azure and Kubernetes CLIs installed. For more information about installing Docker, view Docker documentation. Use `docker login` to log into a Docker account.
- A valid repository for the image on a registry service such as Docker Hub. For more information about creating repositories on Docker Hub, view Docker documentation. This guide relies on Docker instructions, but it is perfectly acceptable to use another registry service such as Azure Container Registry.
- A valid DxEnterprise license with availability group management features and tunnels enabled. A fully featured developer edition is available for free for non-production use at https://dh2i.com/dxenterprise-dxodyssey-developer-edition. To purchase DxEnterprise software for production workloads, visit https://dh2i.com/store/.

Create the SQL Server + DxEnterprise Container Image

1. Download the dxemssql Dockerfile from DH2i's GitHub to the current working directory.
    - `curl https://raw.githubusercontent.com/dh2i/dxemssql/main/dxemssql.dockerfile -o dxemssql.dockerfile`
2. Build the image using the Dockerfile. The DxEnterprise + SQL Server container image layers DxEnterprise software on top of the base SQL Server image. The Dockerfile contains commands to install .NET Core 3.1, add the DxEnterprise tarball from DH2i's website, and add some additional permissions. Ensure that the image is tagged with a `<Docker_ID>` and `<repository>` (e.g., dh2i/dxemssql).
    - `docker build -t <Docker_ID>/<repository> -f dxemssql.dockerfile ./`
3. Push the image to Docker Hub, or any other registry service.
    - `docker push <Docker_ID>/<repository>`

Configure Kubernetes and Start the StatefulSet

1. Create a resource group in Azure for the Kubernetes cluster.
   - `az group create -n <resource_group> -l <region>`
2. Create a two-node Azure Kubernetes Service (AKS) Cluster in the resource group. For more information about CLI parameters, view Microsoft documentation.
   - `az aks create -g <resource_group> -n <cluster_name> -c 2 --disable-rbac --generate-ssh-keys`
3. Obtain the access credentials for the AKS cluster.
   - `az aks get-credentials -g <resource_group> -n <cluster_name>`
4. Download the example YAML deployment from DH2i's GitHub.
   - `curl https://raw.githubusercontent.com/dh2i/dxemssql/main/dxemssql.yaml -o dxemssql.yaml`
5. Use sed to insert the Docker ID and repository into the YAML configuration. Replace the `<Docker_ID>` and `<repository>` with those used in steps 2 and 3 of the previous section. Note that the backslash after the `<Docker_ID>` is required as part of the command.
   - `sed -i 's/image:.*/image: <Docker_ID>\/<repository>/' dxemssql.yaml`
   - `e.g., sed -i 's/image:.*/image: dh2i\/dxemssql/' dxemssql.yaml`
6. Create a credential for the SQL instance.
   - `kubectl create secret generic mssql --from-literal=SA_PASSWORD="<password>"`
7. List the nodes, then apply the primary label to one of the Kubernetes nodes (VM).
   - `kubectl get nodes`
   - `kubectl label node <node_name> role=ags-primary`
8. Apply the secondary label to the remaining node.
   - `kubectl label node <node_name> role=ags-secondary`
9. Apply the StatefulSet configuration.
   - `kubectl apply -f dxemssql.yaml`
10. Check the status of the pods using one or both of the commands below. Proceed to the next section after the primary pod's status updates to "running".
    - `kubectl get pods`
    - `kubectl describe pods`

Configure the Primary and Create the Availability Group

All DxCLI commands can be run from outside the container using the command format `kubectl exec <pod_name> -- dxcli <command>`. This format will be used for all of the subsequent DxCLI command examples.

1. Activate the DxEnterprise license using the command `dxcli activate-sever`.

   Syntax

   ```
   dxcli activate-server <license_key> [node]
   ```

Parameters

Name	Description	Required
license_key	The license key.	True
node	The name of the node.	False

Example

```
kubectl exec dxemssql-primary-0 -- dxcli activate-server AAAA-BBBB-CCCC-DDDD
```

2. Add a Vhost to the cluster using the command `dxcli cluster-add-vhost`.

Syntax

```
dxcli cluster-add-vhost <vhost> <vips> <nodes | VHOST:vhosts> [autofailback] [1-5] [ilb_ports]
```

Parameters

Name	Description	Required
vhost	The name of the Vhost.	True
vips	The virtual IP(s) for the Vhost (comma separated list for multiples). The use of a loopback address (127.0.0.1) is supported, but must be preceded by an asterisk (*).	True
nodes | VHOST:vhosts	The node(s) or Vhost(s) to add to the Vhost (comma separated list for multiples).	True
autofailback	Set autofailback or leave blank if autofailback is not desired.	False
priority	The priority order of failover between Vhosts (1 is the highest and 5 is the lowest)	False
ilbports	The ports to use for internal load balancer probing (comma separated list for multiples).	False

Example

```
kubectl exec dxemssql-primary-0 -- dxcli cluster-add-vhost vhost1 *127.0.0.1 dxemssql-primary-0
```

3. Encrypt the SQL Server sysadmin password for DxEnterprise using the command `dxcli encrypt-text`. The encrypted password will be used to create the availability group in the next step.

Syntax

```
dxcli encrypt-text <value>
```

Parameters

Name	Description	Required
value	The text to be encrypted.	True

Example

```
kubectl exec dxemssql-primary-0 -- dxcli encrypt-text p@ssw0rd
```

4. Add an availability group to the Vhost using the command `dxcli add-ags`. The SQL Server sysadmin password must be encrypted using the command `dxcli encrypt-text` from the previous step.

Syntax

```
dxcli add-ags <vhost> <ags_name>
<node_name|instance_name|sql_login|sql_pass|mirror_port|availability
_mode|[tunnel_port]>
```

Parameters

Name	Description	Required
vhost	The name of the Vhost.	True
ags_name	The name of the availability group.	True
node_name	The name of the node.	True
instance_name	The name of the instance.	True
sql_login	The user name used to login to SQL Server. The SQL Server login can be any SQL login or Windows domain account that has been assigned the sysadmin role for the instance being managed. When supplying a domain account, the username needs to follow the UPN format (for example: user@domain.com).	True
sql_pass	The password used to login to SQL Server (encrypted using `dxcli encrypt-text`).	True
mirror_port	The mirroring port to use for the availability group (default is 5022).	True
availability_mode	synchronous_commit, asynchronous_commit, or configuration_only.	True
tunnel_port	The port to be used for the tunnel connection. This port should be unique per node.	False

Example

```
kubectl exec dxemssql-primary-0 -- dxcli add-ags vhost1 ags1
"dxemssql-primary-
0|mssqlserver|sa|6pnFaDrRS+W/F+dkRuPKAA==|5022|synchronous_commit|40
001"
```

5. Set a One-Time PassKey (OTPK) using the command `dxcli set-otpk`. The output from this command will be used to join the other nodes to the DxEnterprise cluster.

Syntax

```
dxcli set-otpk [ttl] [otpk]
```

Parameters

Name	Description	Required
ttl	The time to live.	False
otpk	The one-time passkey in base64	False

Example

```
kubectl exec dxemssql-primary-0 -- dxcli set-otpk
```

Join the Second Container to the DxEnterprise Cluster

1. Activate the DxEnterprise license for the second node using the command `dxcli activate-server`.

   Example

   ```
   kubectl exec dxemssql-secondary-0 -- dxcli activate-server AAAA-BBBB-CCCC-DDDD
   ```

2. Join the second node to the DxEnterprise cluster using the command `dxcli join-cluster-ex`. Use the default NAT proxy of **match.dh2i.com**.

   Syntax

   ```
   dxcli join-cluster-ex <target> <passkey> [do_nat [true|false]]
   ```

   Parameters

Name	Description	Required
target	The IP address of the target server or match.dh2i.com if joining via NAT matchmaking.	True
passkey	The passkey for the target cluster or OTPK if joining via NAT matchmaking.	True
do_nat	Use the NAT matchmaking service instead of a cluster passkey.	False

   Example

   ```
   kubectl exec dxemssql-secondary-0 -- dxcli join-cluster-ex match.dh2i.com 331bc8bf-7096-99bc-05e5-0dd097393600 true
   ```

3. Add the second node to the existing availability group using the command `dxcli add-ags-node` command. The SQL Server sysadmin password must be encrypted using the command `dxcli encrypt-text`.

   Syntax

   ```
   dxcli add-ags-node <vhost> <ags_name>
   <node_name|instance_name|sql_login|sql_pass|mirror_port|availability_mode|[tunnel_port]>
   ```

   Parameters

Name	Description	Required
vhost	The name of the Vhost.	True
ags_name	The name of the availability group.	True
node_name	The name of the node.	True
instance_name	The name of the instance.	True
sql_login	The user name used to login to SQL Server. The SQL Server login can be any SQL login or Windows domain account that has been assigned the sysadmin role for the instance being managed. When supplying a domain account, the username	True

	needs to follow the UPN format (for example: user@domain.com).	
sql_pass	The password used to login to SQL Server (encrypted using `dxcli encrypt-text`).	True
mirror_port	The mirroring port to use for the availability group (default is 5022).	True
availability_mode	synchronous_commit, asynchronous_commit, or configuration_only.	True
tunnel_port	The port to be used for the tunnel connection. This port should be unique per node.	False

Example

```
kubectl exec dxemssql-secondary-0 -- dxcli add-ags-node vhost1 ags1
"dxemssql-secondary-
0|mssqlserver|sa|6pnFaDrRS+W/F+dkRuPKAA==|5022|synchronous_commit|40
002"
```

Join the Third Container to the DxEnterprise Cluster

1. Activate the DxEnterprise license for the third node using the command `dxcli activate-server`.

   Example

   ```
   kubectl exec dxemssql-secondary-1 -- dxcli activate-server AAAA-
   BBBB-CCCC-DDDD
   ```

2. Join the third node to the DxEnterprise cluster using the command `dxcli join-cluster-ex` with the same parameters from step 2.

   Example

   ```
   kubectl exec dxemssql-secondary-1 -- dxcli join-cluster-ex
   match.dh2i.com 331bc8bf-7096-99bc-05e5-0dd097393600 true
   ```

3. Add the third node to the existing availability group using the command `dxcli add-ags-node`.

   - Example 1: Add the third node as an additional synchronous_commit replica.

     Example

     ```
     kubectl exec dxemssql-secondary-1 -- dxcli add-ags-node vhost1
     ags1 "dxemssql-secondary-
     1|mssqlserver|sa|6pnFaDrRS+W/F+dkRuPKAA==|5022|synchronous_com
     mit|40003"
     ```

   - Example 2: Add the third node as a configuration_only replica.

     Example

     ```
     kubectl exec dxemssql-secondary-1 -- dxcli add-ags-node vhost1
     ags1 "dxemssql-secondary-
     ```

```
1|mssqlserver|sa|6pnFaDrRS+W/F+dkRuPKAA==|5022|configuration_only|40003"
```

Add the Availability Group Database(s)

1. (optional) If a database is not already available, create it using `sqlcmd` on the primary.
   - `kubectl exec dxemssql-primary-0 -- /opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P p@ssw0rd -Q "create database db1"`

2. Add databases to the availability group from the primary using the command `dxcli add-ags-databases`.

   Syntax

   ```
   dxcli add-ags-databases <vhost> <ags_name> <db_name>
   ```

   Parameters

Name	Description	Required
vhost	The name of the Vhost.	True
ags_name	The name of the availability group.	True
db_name	The name of the SQL Server database (comma-separated list for multiples).	True

   Example

   ```
   kubectl exec dxemssql-primary-0 -- dxcli add-ags-databases vhost1 ags1 db1
   ```

3. Verify the details of the availability group using the command `dxcli get-ags-detail`.

   Syntax

   ```
   dxcli get-ags-detail <vhost> <ags_name>
   ```

   Parameters

Name	Description	Required
vhost	The name of the Vhost.	True
ags_name	The name of the availability group.	True

   Example

   ```
   kubectl exec dxemssql-primary-0 -- dxcli get-ags-detail vhost1 ags1
   ```

(Optional) Create an AG Listener and Load Balancer for the StatefulSet

Load balancers can be used by Kubernetes to provide access to cluster resources via a public IP address. The steps below describe how to apply a load balancer that will allow access to the DxEnterprise management port, the SQL Server management port, and the AGS listener port.

1. Set a cluster passkey using the command `dxcli cluster-set-secret-ex`. A cluster passkey is required for connecting to the cluster using the DxAdmin UI.

   Syntax

   ```
   dxcli cluster-set-secret-ex <passkey>
   ```

Parameters

Name	Description	Required
passkey	The passkey to use for the cluster.	True

Example

```
kubectl exec dxemssql-primary-0 -- dxcli cluster-set-secret-ex
p@ssw0rd
```

2. Set an AG listener port using the command `dxcli add-ags-listener`.

Syntax

```
dxcli add-ags-listener <vhost> <ags_name> <node_name>
```

Parameters

Name	Description	Required
vhost	The name of the Vhost.	True
ags_name	The name of the availability group	True
node_name	The name of the node.	True

Example

```
kubectl exec dxemssql-primary-0 -- dxcli add-ags-listener vhost1
ags1 14033
```

3. Download the example load balancer configuration from DH2i's GitHub.
   - `curl https://raw.githubusercontent.com/dh2i/dxemssql/main/lb.yaml -o lb.yaml`
4. Apply the load balancer.
   - `kubectl apply -f lb.yaml`
5. List the service and note the external IP address.
   - `kubectl get service`
6. Connect to the instance (port 1433) or AG listener (port 14033) via SSMS and the external IP address.
7. DxEnterprise's administration port of 7979 is used for DxCLI and DxAdmin, DxEnterprise's client UI. Use the external IP address from step 5 and passkey from step 1 to connect to the cluster from a remote machine. For more information about installing DxAdmin, view the DxAdmin Client UI Quick Start Guide.

(Optional) Add a Static IP Address to the Service

When creating a Kubernetes load balancer service in Azure, the service is assigned a public IP address that lasts for the lifespan of the resource. This IP address will change if the service is deleted and recreated. Reference Microsoft documentation for further information about static IPs in Kubernetes. To assign a static IP that persists between service redeployments, do the following:

1. Delete any running load balancer resource for the dxemssql cluster.
   - `kubectl delete -f lb.yaml`

2. Create a Public IP address resource with a static allocation method and standard SKU using the Azure CLI. The public IP address should be assigned to the *additional resource group* Azure created during the initial Kubernetes cluster deployment. The name of this auto-generated resource group follows the pattern "MC_<resource-group>_<cluster-name>_<region>". To list the name, run the command `az group list | grep "MC_"`. For example:
    - `az network public-ip create -g MC_examplegroup_examplecluster_westus2 -n exampleStaticIPName --sku Standard --allocation-method static`
3. Copy the `ipAddress` value from the JSON output.
4. Use the sed command below to add the static IP to the service YAML file. Replace the `<Public_IP>` value with the IP address copied in step 2. Note in the command that there are two spaces after the backslash.
    - `sed -i '/.*type:.*/a \  loadBalancerIP: <Public_IP>' lb.yaml`
5. Reapply the load balancer.
    - `kubectl apply -f lb.yaml`
6. Verify the load balancer is using the new static IP.
    - `kubectl get service`

References

- Azure CLI Reference
- Azure - Create a Static IP for Kubernetes
- Docker CLI Reference
- DxEnterprise DxCLI Reference
- Kubernetes CLI Reference
- Kubernetes StatefulSet Documentation
- Quorum Considerations for SQL Server Availability Groups