

# Broadcom CA Test Data Manager

## Azure Kubernetes Service

Continuous Testing Solution Engineering Team

DRAFT version 1.0

April, 2021



## Table of Contents

Introduction	3
TDM Architecture Diagram	3
Example Setup	4
Connecting to TDM	7
Troubleshooting/Useful commands	7
Appendix A: Creating your own Azure Container Registry (acr)	9
Appendix B: Using Azure SQL for the TDM Repository (“gtrep”)	10
Appendix C: Sample .yaml files	16

## Introduction

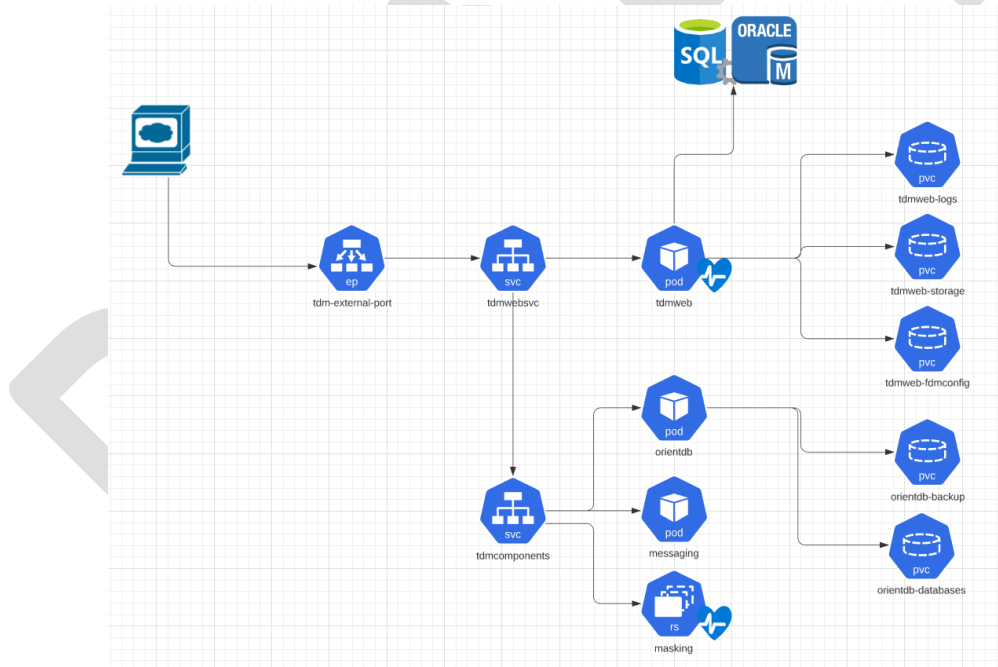
The purpose of this document is to provide information about deploying Broadcom Test Data Manager in the Azure Kubernetes Service (AKS) cloud environment. **The steps described below are for a PROOF OF CONCEPT only. It does not address scalability, security, or storage requirements beyond a simple proof. Production deployments of TDM in AKS require additional configuration per your organizational standards.**

### Note:

- This guide deals discusses AKS-specific steps and modifications. See the companion “TDM Docker for Kubernetes” document that describes the foundational information.
- ***AKS is not formally supported by Broadcom beyond the standard Kubernetes functions.***

## TDM Architecture Diagram

The below diagram shows a basic TDM K8s logical deployment architecture.



- tdmweb-fdmconfig is obsolete on the diagram above; you won't see it referenced below

## Example Setup

The steps noted here are accurate of the date of the publication of this document. Any modifications in AKS or the Broadcom docker images should be verified to determine if changes need to be verified.

These steps are based on the [AKS Quickstart](#) example.

Pre-requisites:

- Azure account
- Access to Azure Cloud Shell (Bash) - with storage account already created.
- An empty TDM “Test Data Repository” (SQL Server or Oracle) instance is available. (See [Appendix B](#) for an AzureSQL setup example).


1 Launch Azure Cloud Shell (Bash)

2 Create an Azure resource group to hold your AKS information (if not already done)

```
az group create --name myTDMAKSResourceGroup --location eastus2
```

3 Create an AKS cluster. The node count is determined by your architectural needs. Refer to the AKS Quickstart documentation regarding the pre-requisites should you enable the monitoring addon. DS3 is the minimum size VM suitable for TDM.

```
az aks create --resource-group myTDMAKSResourceGroup --name myAKSTDMCluster --node-count 1 --enable-addons monitoring --node-vm-size Standard_DS3_v2 --generate-ssh-keys [optionally, if the images registry is ACR]: --attach-acr /subscriptions/<subscriptionid>/resourceGroups/myTDMAKSResourceGroup/providers/Microsoft.ContainerRegistry/registries/<acrname>
```



```
Bash
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

scott@Azure:~$ az aks create --resource-group myTDMAKSResourceGroup --name myTDMAKSCluster --node-count 1 --generate-ssh-keys
- Running ..
```

\* Refer to the quickstart for information on how to enable monitoring.

(Review the JSON response to verify the provisioningState is “Succeeded”)

- 4 After the deployment is complete, get the cluster credentials to save to the config file.

```
az aks get-credentials --resource-group myTDMAKSResourceGroup --name myAKSTDMCluster  
  
Merged "myAKSTDMCluster" as current context in /home/<id>/kube/config
```

- 5 Confirm that the nodes are created

```
kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
aks-nodepool1-33788036-vmss000000	Ready	agent	98s	v1.18.14

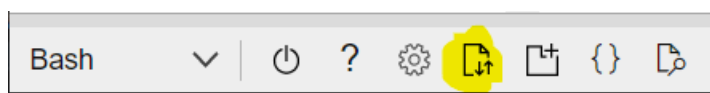
- 6 See [Appendix C](#) for example .yaml files for the deployment

Update the TDM-k8-Complete-AKS.yaml for the 6 GTREP\* variables with the gtrep database information. Examples shown below:

```
- name: GTREP_DATABASE  
  value: gtrep  
  
- name: GTREP_DB_TYPE  
  value: sqlserver  
  
- name: GTREP_HOST  
  value: <mytdmrepo>.database.windows.net  
  
- name: GTREP_PASSWORD  
  value: "<mypswd>"  
  
- name: GTREP_PORT  
  value: "1433"  
  
- name: GTREP_USER  
  value: gtrep
```

Update the TDM-k8-Complete-AKS.yaml "images:" tags to point to the private docker registry where you've pushed the TDM Docker images downloaded from the Broadcom Support site. See [Appendix A](#) for example commands to configure Azure Container Registry to host these images.

- 7 Open the Cloud Shell. Once the Bash shell appears, upload the two .yaml files



Now we are ready to create Persistent Volumes per [AKS policies](#). From the Bash shell, execute:

***kubectl apply -f TDM-k8s-Storage-AKS.yaml***

persistentvolumeclaim/tdmweb-logs created

persistentvolumeclaim/tdmweb-storage created

persistentvolumeclaim/tdmweb-fdmconfig created

persistentvolumeclaim/orientdb-backup created

persistentvolumeclaim/orientdb-config created

persistentvolumeclaim/orientdb-databases created

8 Next, run the complete yaml to create the pods & services

***kubectl apply -f TDM-k8s-Complete-AKS.yaml***

service/tdmwebsvc created

service/tdmcomponents created

service/tdm-external-port created

pod/orientdb created

pod/messaging created

pod/action-download created

pod/tdmweb created

deployment.apps/masking created

**NOTE:** It will take some number of minutes [10-12] for TDM to fully start – you can use the “kubectl logs tdmweb” command to monitor the progress of the services starting up. The server is up when you see

31-Mar-2021 18:40:23.400 INFO [main] org.apache.catalina.startup.Catalina.start Server startup in [301,382] milliseconds

- 9 Determine how to access TDM running on pods in the cluster

## Connecting to TDM

***kubectrl --namespace default get services***

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
kubernetes	ClusterIP	10.0.0.1	<none>	443/TCP
tdm-external-port	LoadBalancer	10.0.80.61	40.76.153.8	443:30610/TCP
tdmcomponents	ClusterIP	None	<none>	2424/TCP,5671/TCP
tdmwebsvc	NodePort	10.0.107.188	<none>	8443:31141/TCP

You'll see a tdm-external-port look for the EXTERNAL-IP address and the mapped port. Use that address/port combination with /TestDataManager to point a browser to the TDM Portal interface.

e.g. <https://40.76.153.8:443/TestDataManager>

You will use port 443 in the URL. This is mapped to the internal port 8443 in the "complete" .yaml file.

Login with the default administrator/marmite combination to verify that connectivity is correct.

A quick use-case checkout can be performed by creating another SQL Database, and selecting "sample" at the Additional Settings/Data source section – that will create an AdventureWorksLT sample db on your db server.

## Troubleshooting/Useful commands

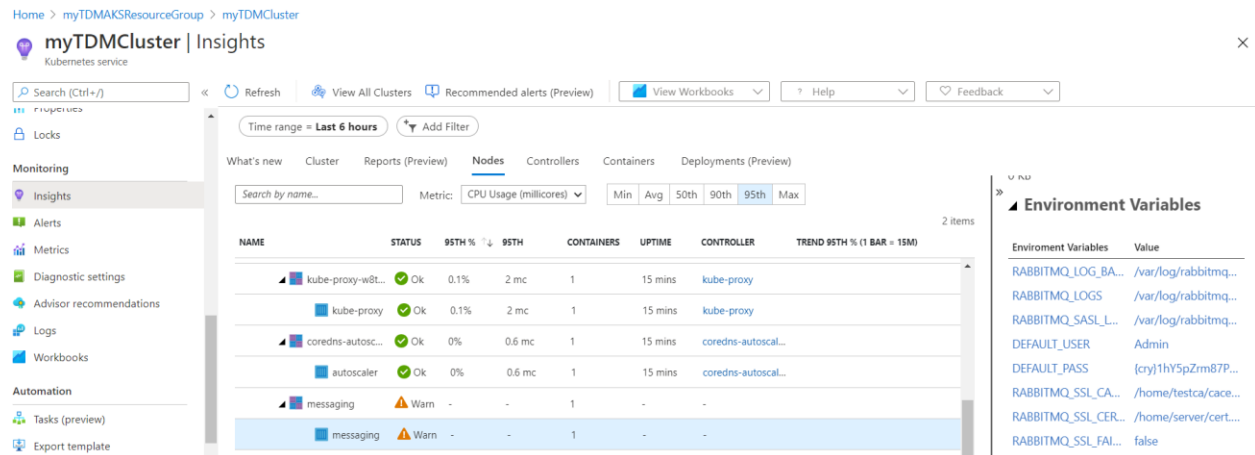
Verify that the external port is available:

***kubectrl get service tdm-external-port --watch***

Check pod status via command line:

***kubectrl get pods***

Alternately, use the Kubernetes monitoring dashboard to check status of pods, etc.



Execute the below command to verify that the grep db initialization/services startup script has completed successfully:

**kubectl logs tdmweb**

Execute the below command to verify that the tdmweb container is running properly:

**kubectl describe pod tdmweb**

Other key commands:

**kubectl describe pvc**

Shows the structure and status of the Storage.

**kubectl describe node**

Shows the current status of the K8s node hosting the pods



## Appendix A: Creating your own Azure Container Registry (acr)

```
az acr create --resource-group <myResourceGroup> --name <acrName> --sku Basic
```

Then, from a Docker system with the images pulled & ([with azure client installed](#))

```
az acr login --name <acrName>
```

```
docker tag (your docker image/version) <acrLoginServer>/(your front-end docker image/version)
```

```
docker images (verify the tag)
```

```
docker push <acrLoginServer>/(your docker image/version)
```

(repeat for all images, resulting in) :

```
az acr repository list --name <acrName>
```

```
[  
  "tdm/action-service",  
  "tdm/masking",  
  "tdm/messaging",  
  "tdm/orientdb",  
  "tdm/tdmtools",  
  "tdm/tdmweb"  
]
```

Use attach-acr switch when creating cluster.

```
az aks create \  
  --resource-group myResourceGroup \  
  --name myAKSTDMCluster \  
  --node-count 1 \  
  --generate-ssh-keys \  
  --attach-acr <acrName>
```

Use full path to acr if not in the same subscription - see [here](#).

## Appendix B: Using Azure SQL for the TDM Repository (“gtrep”)

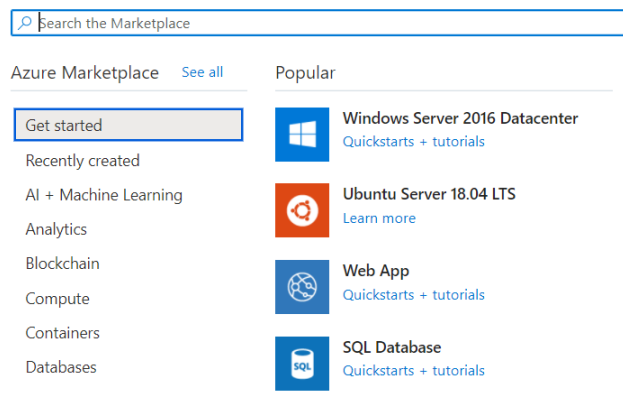
Pre-requisites: Logged in with your Azure ID, with an Azure Resource Group created.

SQL Server Management Studio installed on your local system.

While in the Resource Group dashboard, Click the Add button and select SQL Database

[Home](#) > [myTDMAKSResourceGroup](#) >

### New



Define the database name (**MUST BE “gtrep”**), and create a new server. Configure the storage to your organization’s needs.

### Create SQL Database

Microsoft

manage all your resources.

Subscription \* ⓘ Visual Studio Professional Subscription

Resource group \* ⓘ myTDMAKSResourceGroup [Create new](#)

#### Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name \* gtrep ✓

Server \* ⓘ (new) sschmitz-tdmrepo (East US 2) [Create new](#)

Want to use SQL elastic pool? \* ⓘ ☐ Yes ☒ No

Compute + storage \* ⓘ **General Purpose**  
Gen5, 2 vCores, 32 GB storage, zone redundant disabled  
[Configure database](#)

**\*\*\*\* IT IS CRITICAL TO DEFINE A DATABASE CONFIGURATION THAT IS PERFORMANT. IF THE TDM REPOSITORY INITIALIZATION CANNOT COMPLETE WITHIN 300 SECONDS (5 MINUTES), THEN THE TDMWEB POD WILL NEVER STABILIZE & TDM WON'T WORK \*\*\***

Specify a public endpoint and allow Azure services and resources to access the server. Add current client IP address.

[Home](#) > [myTDMASResourceGroup](#) > [New](#) >

## Create SQL Database


Microsoft

Basics **Networking** Additional settings Tags Review + create

Configure network access and connectivity for your server. The configuration selected below will apply to the selected server 'sschmitz-tdmrepo' and all databases it manages. [Learn more](#)

### Network connectivity

Choose an option for configuring connectivity to your server via public endpoint or private endpoint. Choosing no access creates with defaults and you can configure connection method after server creation. [Learn more](#)

Connectivity method \*   
☐ No access  
☒ Public endpoint  
☐ Private endpoint

### Firewall rules

Setting 'Allow Azure services and resources to access this server' to Yes allows communications from all resources inside the Azure boundary, that may or may not be part of your subscription. [Learn more](#)  
Setting 'Add current client IP address' to Yes will add an entry for your client IP address to the server firewall.

Allow Azure services and resources to access this server \*    
Add current client IP address \*

[Review + create](#) [< Previous](#) [Next : Additional settings >](#)

## Use the defaults for additional settings

[Home](#) > [myTDMASResourceGroup](#) > [New](#) >

## Create SQL Database

Microsoft

Basics Networking **Additional settings** Tags Review + create

Customize additional configuration parameters including collation & sample data.

### Data source

Start with a blank database, restore from a backup or select sample data to populate your new database.

Use existing data \*

### Database collation

Database collation defines the rules that sort and compare data, and cannot be changed after database creation. The default database collation is SQL\_Latin1\_General\_CP1\_CI\_AS. [Learn more](#)

Collation \*    
[Find a collation](#)

### Azure Defender for SQL

Protect your data using Azure Defender for SQL, a unified security package including vulnerability assessment and advanced threat protection for your server. [Learn more](#)

Get started with a 30 day free trial period, and then 15 USD/server/month.

Enable Azure Defender for SQL \* 

[Review + create](#) [< Previous](#) [Next : Tags >](#)

## Create the database/database server

[Home](#) > [myTDMASResourceGroup](#) > [New](#) >

### Create SQL Database

Microsoft

frequency as my Azure subscription; and (c) agree that Microsoft may share my contact, usage and transactional information third-party offerings. For additional details see [Azure Marketplace Terms](#).

#### Basics

Subscription	Visual Studio Professional Subscription
Resource group	myTDMASResourceGroup
Region	East US
Database name	gtrep
Server	(new) sschmitz-tdmrepo
Compute + storage	General Purpose: Gen5, 2 vCores, 32 GB storage, zone redundant disabled

#### Networking

Allow Azure services and resources to access this server	Yes
Add current client IP address 35.227.4.138	Yes
Private endpoint	None

#### Additional settings

Use existing data	Blank
Collation	SQL_Latin1_General_CP1_CI_AS

Create

< Previous

[Download a template for automation](#)

[Home](#) >

Microsoft.SQLDatabase.newDatabaseNewServer\_4209dc0b91ff45638c8b7 | Overview

Deployment

Search (Ctrl+F) Delete Cancel Redeploy Refresh

Overview

Inputs

Outputs

Template

We'd love your feedback! →

✓ Your deployment is complete

Deployment name: Microsoft.SQLDatabase.newDatabaseNewServer\_4209dc0b91ff45638c8b7 Start time: 1/5/2021, 9:52:13 AM  
Subscription: Visual Studio Professional Subscription Correlation ID: 3ba0f59d-7681-4a34-870d-e9c59e29890f  
Resource group: myTDMASResourceGroup

Deployment details (Download)

Next steps

[Go to resource](#)

[Home](#) > [Microsoft.SQLDatabase.newDatabaseNewServer\\_4209dc0b91ff45638c8b7](#) >

gtrep (sschmitz-tdmrepo/gtrep)

SQL database

Search (Ctrl+F) Copy Restore Export Set server firewall Delete Connect with... Feedback

Overview

Activity log

Tags

Diagnose and solve problems

Quick start

Query editor (preview)

Power Platform

Power BI (preview)

Essentials

Resource group (change) : myTDMASResourceGroup

Status : Online

Location : East US

Subscription (change) : Visual Studio Professional Subscription

Subscription ID : 157fa2db-74b1-4131-b80b-aba8b958cc9f

Tags (change) : [Click here to add tags](#)

Server name : sschmitz-tdmrepo.database.windows.net

Elastic pool : No elastic pool

Connection strings : [Show database connection strings](#)

Pricing tier : General Purpose: Gen5, 2 vCores

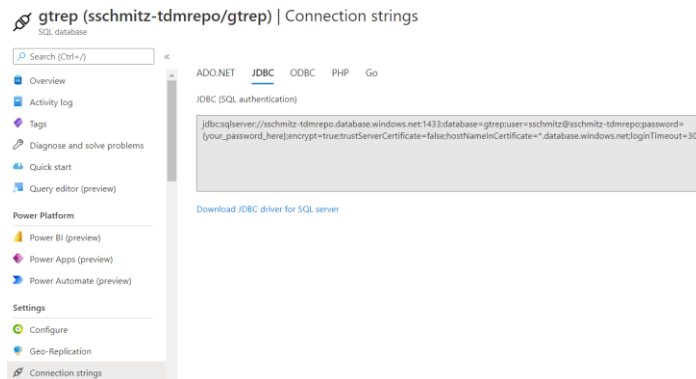
Earliest restore point : No restore point available

Show data for last: 1 hour 24 hours 7 days

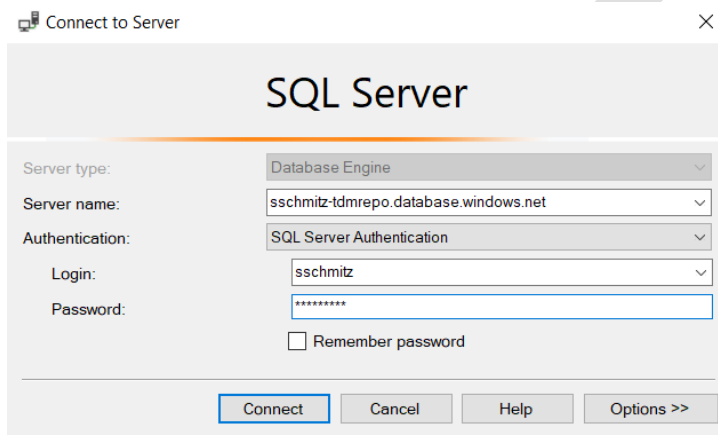
Aggregation type: Max

Compute utilization

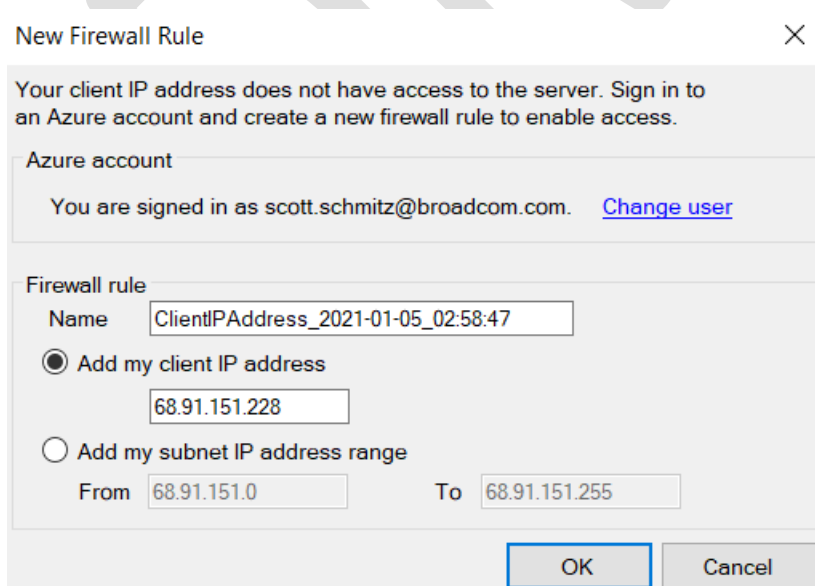
Confirm connection information by selecting the “Connection strings” hyperlink or left menu item



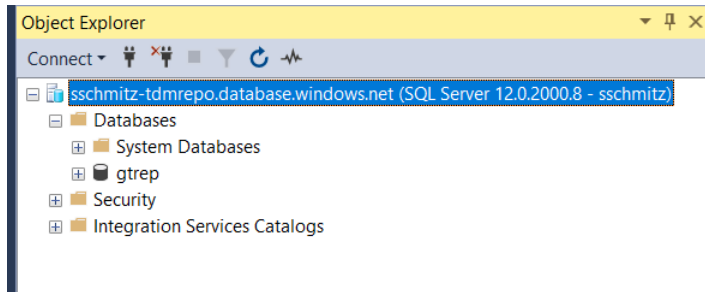
Launch SQL Server Management Studio from your local box and connect to the Server Name as listed above using the credentials specified in the setup.



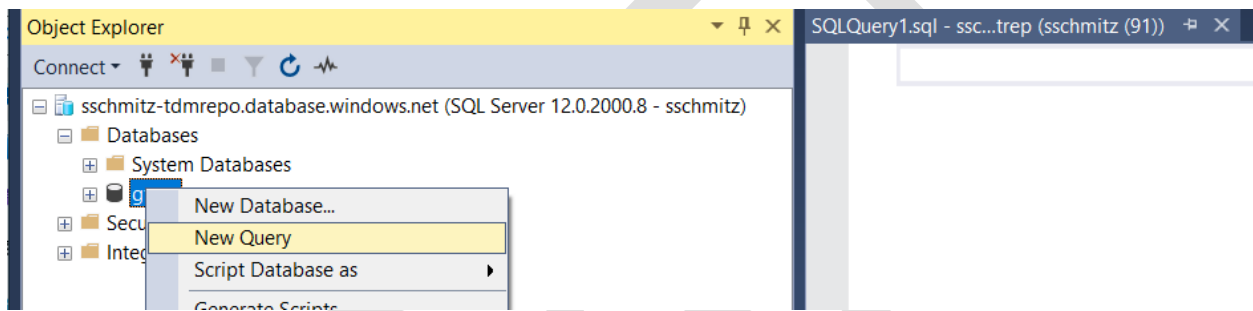
You MAY be prompted to sign-in again to Azure and create a new firewall rule.



Once connected, the SSMS Object Explorer will show the gtrep database



Next, we need to configure it per the TDM RepoKit's "gtrep-schema.sql". Right-mouse on the gtrep db and select New Query to open a query window.



**Due to the different configuration of Azure SQL (versus SQL Server), the OOTB database script needs to be slimmed down to the following.** Paste into the SSMS query window and select "Execute".

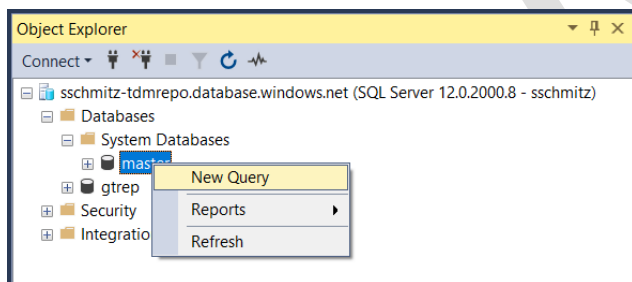
```
ALTER DATABASE [gtrep] SET COMPATIBILITY_LEVEL = 100
GO
ALTER DATABASE [gtrep] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [gtrep] SET ANSI_NULLS OFF
GO
ALTER DATABASE [gtrep] SET ANSI_PADDING OFF
GO
ALTER DATABASE [gtrep] SET ANSI_WARNINGS OFF
GO
ALTER DATABASE [gtrep] SET ARITHABORT OFF
GO
ALTER DATABASE [gtrep] SET AUTO_CREATE_STATISTICS ON
GO
ALTER DATABASE [gtrep] SET AUTO_SHRINK OFF
GO
ALTER DATABASE [gtrep] SET AUTO_UPDATE_STATISTICS ON
GO
ALTER DATABASE [gtrep] SET CURSOR_CLOSE_ON_COMMIT OFF
GO
```

```

ALTER DATABASE [gtrep] SET CONCAT_NULL_YIELDS_NULL OFF
GO
ALTER DATABASE [gtrep] SET NUMERIC_ROUNDABORT OFF
GO
ALTER DATABASE [gtrep] SET QUOTED_IDENTIFIER OFF
GO
ALTER DATABASE [gtrep] SET RECURSIVE_TRIGGERS OFF
GO
ALTER DATABASE [gtrep] SET AUTO_UPDATE_STATISTICS_ASYNC OFF
GO
ALTER DATABASE [gtrep] SET ALLOW_SNAPSHOT_ISOLATION OFF
GO
ALTER DATABASE [gtrep] SET PARAMETERIZATION SIMPLE
GO
ALTER DATABASE [gtrep] SET MULTI_USER
GO
USE [gtrep]
GO
ALTER DATABASE [gtrep] SET READ_WRITE
GO

```

Expand “System Databases” to show “master”. Right-mouse and select “New Query”



In the Query window, execute the following commands: (modify the password to your org. standards)

```

CREATE LOGIN gtrep WITH password='Gridt00ls'
GO

```



Switch back to the gtrep db Query window, and execute the following command:

```

CREATE USER [gtrep] FOR LOGIN [gtrep] WITH DEFAULT_SCHEMA=[dbo]
GO
EXEC sp_addrolemember 'db_owner', 'gtrep';
GO

```

Congratulations – you’ve setup the empty gtrep database that will be populated by TDM Portal once you’ve logged into the TDM Docker container. The server and database will show on your Azure dashboard:

<input type="checkbox"/>	 gtrep (sschmitz-tdmrepo/gtrep)	SQL database	East US 2
<input type="checkbox"/>	 sschmitz-tdmrepo	SQL server	East US 2

## Appendix C: Sample .yaml files

Create these files locally, then upload using the Bash shell (see instructions above)

### TDM-k8s-Storage-AKS.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: tdmweb-logs
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: managed-premium
resources:
  requests:
    storage: 1Gi
```

---

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: tdmweb-storage
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: managed-premium
resources:
  requests:
    storage: 1Gi
```

---

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: orientdb-backup
```



```
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: managed-premium
  resources:
    requests:
      storage: 100Mi
```

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: orientdb-config
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: managed-premium
  resources:
    requests:
      storage: 100Mi
```

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: orientdb-databases
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: managed-premium
  resources:
    requests:
      storage: 2Gi
```

**TDM-k8s-Complete-AKS.yaml**

```
---
apiVersion: v1
kind: Service
metadata:
  labels:
    app: tdmhost
    name: tdmwebsvc
    name: tdmwebsvc
spec:
  type: NodePort
  ports:
    - name: "8443"
      port: 8443
      targetPort: 8443
  selector:
    name: tdmwebsvc
---

apiVersion: v1
kind: Service
metadata:
  labels:
    name: tdmsvc
    name: tdmcomponents
spec:
  clusterIP: None
  ports:
    - name: "2424"
      port: 2424
      targetPort: 2424
    - name: "5671"
      port: 5671
      targetPort: 5671
```

```
selector:
```

```
  name: tdmcomponents
```

```
---
```

```
apiVersion: v1
```

```
kind: Service
```

```
metadata:
```

```
  name: tdm-external-port
```

```
spec:
```

```
  selector:
```

```
    name: tdmwebsvc
```

```
  ports:
```

```
    - port: 443
```

```
      targetPort: 8443
```

```
  type: LoadBalancer
```

```
---
```

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: orientdb
```

```
  labels:
```

```
    name: tdmcomponents
```

```
spec:
```

```
  hostname: orientdb
```

```
  subdomain: tdmcomponents
```

```
  containers:
```

```
    - name: orientdb
```

```
      image: tdmacr.azurecr.io/tdm/orientdb:latest
```

```
  resources:
```

```
    limits:
```

```
      memory: "8Gi"
```

```
    requests:
```

```
      memory: "3Gi"
```

```
env:
- name: ORIENTDB_ROOT_PASSWORD
  value: '{cry}tHpzgrvNhtVu6uHGnd9EdlAuwMR30OL0sAXhBWdgM3Md'
volumeMounts:
- mountPath: /orientdb/backup
  name: orientdb-backup
- mountPath: /orientdb/databases
  name: orientdb-databases
ports:
- containerPort: 2424
volumes:
- name: orientdb-backup
  persistentVolumeClaim:
    claimName: orientdb-backup
- name: orientdb-databases
  persistentVolumeClaim:
    claimName: orientdb-databases
---
apiVersion: v1
kind: Pod
metadata:
  name: messaging
  labels:
    name: tdmcomponents
spec:
  hostname: messaging
  subdomain: tdmcomponents
  containers:
  - name: messaging
    image: tdmacr.azurecr.io/tdm/messaging:latest
    env:
    - name: RABBITMQ_LOG_BASE
      value: /var/log/rabbitmq/log
    - name: RABBITMQ_LOGS
```

```
value: /var/log/rabbitmq/log/rabbitmq.log
- name: RABBITMQ_SASL_LOGS
value: /var/log/rabbitmq/log/rabbitmq_sasl.log
- name: DEFAULT_USER
value: Admin
- name: DEFAULT_PASS
value: '{cry}1hY5pZrm87PWjgPdmypDbVZnL4a108Ixy8YLuUVRMCr8'
- name: RABBITMQ_SSL_CACERTFILE
value: /home/testca/cacert.pem
- name: RABBITMQ_SSL_CERTFILE
value: /home/server/cert.pem
- name: RABBITMQ_SSL_FAIL_IF_NO_PEER_CERT
value: "false"
- name: RABBITMQ_SSL_VERIFY
value: verify_none
- name: RABBITMQ_SSL_KEYFILE
value: /home/server/key.pem
ports:
- containerPort: 5671
- containerPort: 15672
- containerPort: 15671
...

apiVersion: v1
kind: Pod
metadata:
name: action-download
labels:
name: tdmcomponents
spec:
hostname: action-download
subdomain: tdmcomponents
containers:
- name: action-download
image: tdmacr.azurecr.io/tdm/action-service:latest
ports:
```

```
- containerPort: 9443

env:
- name: ACTION_SECRET
  value: "123"
- name: PUBLISH_ACTION
  value: "/opt/download.sh"
---

apiVersion: v1
kind: Pod
metadata:
  name: tdmweb
  labels:
    name: tdmwebsvc
spec:
  hostname: tdmweb
  subdomain: tdmwebsvc
  containers:
  - name: tdmweb
    image: tdmacr.azurecr.io/tdm/tdmweb:latest
    resources:
      limits:
        memory: "8Gi"
      requests:
        memory: "4Gi"
    volumeMounts:
    - name: tdmweb-logs
      mountPath: /mnt/logs
    - name: tdmweb-storage
      mountPath: mnt/storage
  ports:
  - containerPort: 8443
  env:
  - name: APPLICATION_PROP
    value: tdmweb.TDMMaskingService.taskTimeout=30|tdmweb.profiling.uncommitted.reads=true|tdmweb.profiling.query.timeout=300
  - name: GTREP_DATABASE
```

```
value: gtrep
- name: GTREP_DB_TYPE
value: sqlserver
- name: GTREP_HOST
value: <yourdb>.database.windows.net
- name: GTREP_PASSWORD
value: "<yourpswd>"
- name: GTREP_PORT
value: "1433"
- name: GTREP_USER
value: gtrep
- name: MESSAGING_PASS
value: '{cry}1hY5pZrm87PWjgPdmypDbVZnL4a108lxy8YLuUVRMCr8'
- name: MESSAGING_PORT
value: "5671"
- name: MESSAGING_SERVER
value: messaging.tdmcomponents
- name: MESSAGING_USER
value: Admin
- name: ORIENTDB_HOST
value: orientdb.tdmcomponents
- name: ORIENTDB_PASSWORD
value: '{cry)tHpzgrvNhtVu6uHGnd9EdIAuwMR30OL0sAXhBWdgM3Md'
livenessProbe:
  httpGet:
    path: /TestDataManager
    port: 8443
    scheme: HTTPS
  initialDelaySeconds: 300
  periodSeconds: 30
volumes:
- name: tdmweb-logs
  persistentVolumeClaim:
    claimName: tdmweb-logs
- name: tdmweb-storage
  persistentVolumeClaim:
```

```
claimName: tdmweb-storage
```

```
---
```

```
apiVersion: apps/v1
```

```
kind: Deployment
```

```
metadata:
```

```
  name: masking
```

```
  labels:
```

```
    name: tdmcomponents
```

```
spec:
```

```
  replicas: 1
```

```
  selector:
```

```
    matchLabels:
```

```
      name: tdmcomponents
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        name: tdmcomponents
```

```
    spec:
```

```
      hostname: masking
```

```
      subdomain: tdmcomponents
```

```
      containers:
```

```
        - name: masking
```

```
          image: tdmacr.azurecr.io/tdm/masking:latest
```

```
          env:
```

```
            - name: MESSAGING_SERVER
```

```
              value: messaging.tdmcomponents
```

```
            - name: MESSAGING_PORT
```

```
              value: "5671"
```

```
            - name: MESSAGING_USER
```

```
              value: Admin
```

```
            - name: MESSAGING_PASS
```

```
              value: '{cry}1hY5pZrm87PWjgPdmyPDbVZnL4a108lxy8YLuUVRMCr8'
```

```
          livenessProbe:
```



```
exec:
```

```
command:
```

```
- /bin/sh
```

```
- -c
```

```
- 'cat /opt/tdm/logs/TDMMaskingService*.log | grep -vz "Connection refused"'
```

```
initialDelaySeconds: 10
```

```
periodSeconds: 15
```

```
---
```

DRAFT