WEKA

Cassandra on WEKA in AKS

August 2023

WHITE PAPER

Table of Contents

Executive Summary

Cloud computing has transformed the way applications are developed and deployed, enabling developers to focus on writing code and delivering value to users, rather than dealing with infrastructure complexities, and databases such as Apache Cassandra are often used when performance and scale become a requirement of the deployment.

Apache Cassandra is an open-source NoSQL distributed database management system designed to handle very large datasets across multiple servers. It was originally developed by Facebook and later open-sourced as part of the Apache Software Foundation. It is ideal for applications demanding fast data ingestion and retrieval and is widely used in areas like social media, finance, and IoT due to its efficient handling of time series data.

Cassandra's distributed architecture makes it well-suited for Kubernetes (K8s) cloud deployments. While there are several cloud providers offering variations of Kubernetes support, this paper will discuss the use of Azure Kubernetes Service (AKS) with Cassandra. AKS is a managed Kubernetes container orchestration service provided by Microsoft Azure. It simplifies the deployment, management, and scaling of containerized applications using Kubernetes in the Azure cloud environment. AKS abstracts much of the complexity of setting up and managing a Kubernetes cluster, allowing developers to focus more on deploying and running applications.

It is important to note both Cassandra and Kubernetes share concepts of distributing workloads across multiple nodes and handling failover and scaling, which can be leveraged effectively when deploying containerized Cassandra on a Kubernetes cluster. Running Cassandra in Kubernetes involves creating Kubernetes resources like Deployments, StatefulSets, and Services to manage the Cassandra nodes and ensure they can communicate with each other and with client applications. WEKA on Azure with AKS solves the Cassandra performance and operational challenges at scale while reducing price and complexity – eliminating the need to waste time and effort on constant Cassandra rebuilds while improving Cassandra cluster agility when containerizing it on AKS.

This paper will cover the steps required to configure AKS with Cassandra.

Introduction

This paper methodically goes over the process of setting up a Cassandra cluster that is utilizing WEKA for its storage, removing the need to provision and maintain block devices per instance. We then go through several examples of creating schema and a keyspace as well as populate them with example data. Finally, we go over a scale out example of the cluster as well as nodes failures and a complete cluster destruction and instant recovery from WEKA.

Audience

This paper is for those who plan to deploy The WEKA Data Platform with Cassandra and Azure Kubernetes Service (AKS) or are considering deploying Cassandra and AKS with WEKA as the data storage component.

We have organized this document to address critical items for enabling successful design, implementation, and transition to operation.

Purpose

This document covers the following subject areas:

- Overview of WEKA.
- Prerequisites
- Deploy a Cassandra Cluster via a k8s StatefulSet
- Push a WEKA filesystem snapshot of the database to Microsoft Blob storage
- Demonstrate disaster recovery
- Benefits

Document Version History

Version Number	Published	Notes
1.0	August 2023	**Author: Alex Howard.**

Overview of WEKA

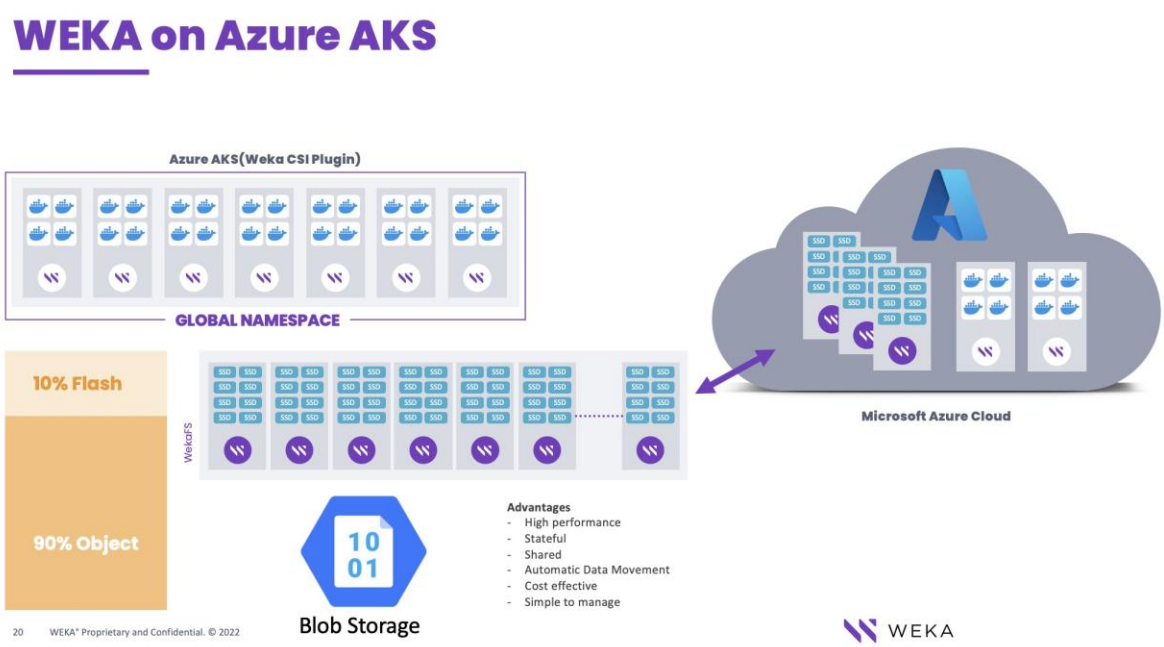
WEKA is a high-performance data platform that contains the WEKA Filesystem. WEKA solves data challenges at scale while accelerating workloads and reducing costs; it can be deployed on physical servers in the data center as well as on multiple cloud environments such as Microsoft Azure cloud and other public clouds. Cassandra nodes can utilize the WEKA filesystem instead of their locally attached SSDs or remote block devices while retaining or accelerating their Cassandra performance with additional benefits that will be demonstrated in this white paper. In addition, the ability to containerize the Cassandra nodes while still providing the same performance as non-containerized Cassandra nodes allows for an extra layer of dynamic management and simplicity without any performance compromise.

Prerequisites

1. Deploy WEKA in Azure - https://docs.weka.io/install/weka-installation-on-azure
2. Deploy an Azure AKS cluster - https://www.weka.io/resources/technical-brief/weka-with-azure-aks/
3. Configure the AKS cluster to access the WEKA filesystem via the WEKA CSI plugin - https://docs.weka.io/appendix/weka-csiplugin

This document will go over the following process:

1. Deploy a Cassandra cluster via a k8s StatefulSet
2. Populate the database with some sample data
3. Push a WEKA filesystem snapshot of the database to Microsoft Blob storage
4. Destroy the WEKA and AKS environments
5. Re-create the WEKA and AKS clusters
6. Restore the Cassandra database from the snapshot on Microsoft Blob storage



Deploying Cassandra with a StatefulSet:

Storage Class

For running Cassandra, start by creating a storage class. Save the following contents to a file, sc_wekafs_dir.yaml:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: storageclass-wekafs-dir-static
provisioner: csi.weka.io
reclaimPolicy: Retain
volumeBindingMode: Immediate
allowVolumeExpansion: true
parameters:
  volumeType: dir/v1
  filesystemName: default
  capacityEnforcement: HARD
  # optional parameters setting UID, GID and permissions on volume
  # UID of the volume owner, default 0 (root)
  #ownerUid: "1000"
  # GID of the volume owner, default 0 (root)
  #ownerGid: "1000"
  # permissions in Unix octal format, default "0750"
  #permissions: "0775"
  # name of the secret that stores API credentials for a cluster
  # change the name of secret to match secret of a particular cluster (if you have several Weka clusters)
  csi.storage.k8s.io/provisioner-secret-name: &secretName csi-wekafs-api-secret
  # change the name of the namespace in which the cluster API credentials
  csi.storage.k8s.io/provisioner-secret-namespace: &secretNamespace csi-wekafs
  # do not change anything below this line, or set to same parameters as above
  csi.storage.k8s.io/controller-publish-secret-name: *secretName
  csi.storage.k8s.io/controller-publish-secret-namespace: *secretNamespace
```

```
  csi.storage.k8s.io/controller-expand-secret-name: *secretName
  csi.storage.k8s.io/controller-expand-secret-namespace: *secretNamespace
  csi.storage.k8s.io/node-stage-secret-name: *secretName
  csi.storage.k8s.io/node-stage-secret-namespace: *secretNamespace
  csi.storage.k8s.io/node-publish-secret-name: *secretName
  csi.storage.k8s.io/node-publish-secret-namespace: *secretNamespace
```

Apply the storage class:

```
alex [ ~/cassandra ]$ kubectl apply -f sc_wekafs_dir.yaml
storageclass.storage.k8s.io/storageclass-wekafs-dir-api created
```

Service File

Next, create and apply a service file for Cassandra:

```
alex [ ~/cassandra ]$ cat cassandra-service.yaml
#cassandra-service.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: cassandra
  name: cassandra
spec:
  clusterIP: None
  ports:
  - port: 9042
  selector:
    app: cassandra
```

Apply the service file:

```
kubectl apply -f cassandra-service.yaml
```

Static, Persistent volumes

Next, create static, persistent volumes, one for each of the Cassandra nodes to be created. These persistent volumes correspond with a static directory that has been created on the WEKA filesystem in the root of the specified filesystem. These examples are using the "default" filesystem. Assuming "default" has been mounted to /weka on a client node, the directories may be created that will be the backing storage for the persistent, static volumes like so:

```
sudo mkdir /weka/cassandra-0 /weka/cassandra-1 /weka/cassandra-2
```

Now that the directories have been created, a persistent volume will be created that corresponds to each of those directories. Note that the only difference between these files is the metadata name and the volumeHandle path:

```
alex [ ~/cassandra/new ]$ cat pvc-cassandra-0.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-wekafs-dir-static-cassandra-0
spec:
  storageClassName: storageclass-wekafs-dir-static
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
  capacity:
    storage: 5Ti
  csi:
    driver: csi.weka.io
    # volumeHandle must be formatted as following:
    # dir/v1/<FILE_SYSTEM_NAME>/<INNER_PATH_IN_FILESYSTEM>
    # The path must exist, otherwise publish request will fail
    volumeHandle: dir/v1/default/cassandra-0

alex [ ~/cassandra/new ]$ kubectl apply -f pvc-cassandra-0.yaml
persistentvolume/pv-wekafs-dir-static-cassandra-0 created
```

```
alex [ ~/cassandra/new ]$ cat pvc-cassandra-1.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-wekafs-dir-static-cassandra-1
spec:
  storageClassName: storageclass-wekafs-dir-static
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
  capacity:
    storage: 5Ti
  csi:
    driver: csi.weka.io
    # volumeHandle must be formatted as following:
    # dir/v1/<FILE_SYSTEM_NAME>/<INNER_PATH_IN_FILESYSTEM>
    # The path must exist, otherwise publish request will fail
    volumeHandle: dir/v1/default/cassandra-1

alex [ ~/cassandra/new ]$ kubectl apply -f pvc-cassandra-1.yaml
persistentvolume/pv-wekafs-dir-static-cassandra-1 created
```

```
alex [ ~/cassandra/new ]$ cat pvc-cassandra-2.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-wekafs-dir-static-cassandra-2
spec:
  storageClassName: storageclass-wekafs-dir-static
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
  capacity:
```

```
    storage: 5Ti
  csi:
    driver: csi.weka.io
    # volumeHandle must be formatted as following:
    # dir/v1/<FILE_SYSTEM_NAME>/<INNER_PATH_IN_FILESYSTEM>
    # The path must exist, otherwise publish request will fail
    volumeHandle: dir/v1/default/cassandra-2

alex [ ~/cassandra/new ]$ kubectl apply -f pvc-cassandra-2.yaml
persistentvolume/pv-wekafs-dir-static-cassandra-2 created
```

StatefulSet File

Finally, it is time to create a Cassandra StatefulSet:

```
alex [ ~/cassandra/new ]$ cat cassandra-statefulset.weka.yaml
#cassandra-statefulset.yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: cassandra
  labels:
    app: cassandra
spec:
  serviceName: cassandra
  replicas: 3
  selector:
    matchLabels:
      app: cassandra
  template:
    metadata:
      labels:
        app: cassandra
    spec:
      terminationGracePeriodSeconds: 1800
      containers:
        - name: cassandra
          image: cassandra:4.0.7
          env:
          - name: CASSANDRA_SEEDS
            value: "cassandra-0.cassandra.default.svc.cluster.local"
          - name: POD_IP
            valueFrom:
              fieldRef:
                fieldPath: status.podIP
          lifecycle:
            preStop:
              exec:
               command:
               - /bin/sh
               - -c
               - nodetool drain
          readinessProbe:
            exec:
              command:
              - /bin/bash
              - -c
              - if [[ $(nodetool status | grep $POD_IP) == *"UN"* ]] ; then exit 0; else exit 1; fi
            initialDelaySeconds: 35
            timeoutSeconds: 5
          volumeMounts:
          - mountPath: "/var/lib/cassandra"
            name: pv-wekafs-dir-static
  volumeClaimTemplates:
  - metadata:
      name: pv-wekafs-dir-static
    apiVersion: v1
    kind: PersistentVolumeClaim
    spec:
```

```
      accessModes: [ "ReadWriteMany" ]
      storageClassName: storageclass-wekafs-dir-static
      volumeMode: Filesystem
      resources:
        requests:
          storage: 5Ti
```

Apply the StatefulSet file:

```
kubectl apply -f cassandra-statefulset.weka.yaml
```

At this point, a single Cassandra pod can be seen running. Note 0/1 is Ready, meaning the pod is not yet ready. Once it is ready (output of nodetool status shows the IP of the pod) it will move on to create the next pod:

```
alex [ ~/cassandra/new ]$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
cassandra-0       0/1     Running   0          13s
installer-2hnk5   1/1     Running   0          45h
installer-c89sx   1/1     Running   0          45h
installer-d5j6t   1/1     Running   0          45h
```

After giving the Cassandra processes a few minutes to start up, the status of the cluster may be viewed using 'nodetool status'. Notice the static persistent volumes have been claimed:

```
alex [ ~/cassandra/new ]$ kubectl exec --stdin --tty cassandra-0 -- nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens  Owns (effective)  Host ID                               Rack
UN  10.0.0.74  74.12 KiB  16      64.7%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1
UN  10.0.0.19  69.06 KiB  16      59.3%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1
UN  10.0.0.57  88.56 KiB  16      76.0%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1


alex [ ~/cassandra/new ]$ kubectl get pvc
NAME                                STATUS   VOLUME                                CAPACITY   ACCESS MODES   STORAGECLASS                    AG
pv-wekafs-dir-static-cassandra-0    Bound    pv-wekafs-dir-static-cassandra-2      5Ti        RWX            storageclass-wekafs-dir-static   4m
pv-wekafs-dir-static-cassandra-1    Bound    pv-wekafs-dir-static-cassandra-0      5Ti        RWX            storageclass-wekafs-dir-static   3m
pv-wekafs-dir-static-cassandra-2    Bound    pv-wekafs-dir-static-cassandra-1      5Ti        RWX            storageclass-wekafs-dir-static   11
```

Cassandra keyspace and schema

Now that there is a working Cassandra cluster, create a keyspace and table. For this scenario consider a blog application with millions of users whose admin might want to query the last X number of posts for any given user. A simple schema and keyspace for such a case might be created like this:

```
alex [ ~ ]$ kubectl exec --stdin --tty cassandra-0 -- /bin/bash
root@cassandra-0:/# cqlsh
Connected to Test Cluster at 127.0.0.1:9042
[cqlsh 6.0.0 | Cassandra 4.0.7 | CQL spec 3.4.5 | Native protocol v 5]
Use HELP for help.
cqlsh>
cqlsh> CREATE KEYSPACE blog WITH REPLICATION = { 'class' : 'NetworkTopologyStrategy', 'datacenter1' : 3 };
cqlsh> CREATE TABLE blog.posts (
   ...   user_id text,
   ...   post_id timeuuid,
   ...   content text,
   ...   PRIMARY KEY (user_id, post_id) )
   ... WITH CLUSTERING ORDER BY (post_id DESC);
cqlsh>
```

Now that there is a 'posts' table, it may be populated with data.

Scale the Cassandra StatefulSet from 3 to 6:

Prior to populating the table with data, expand the Cassandra cluster to 6 nodes. Once complete, there will be 6 Cassandra nodes running on 3 k8s worker nodes.

Additional Persistent Volumes

Create an additional 3 directories to be used by Cassandra:

```
azureuser@alexh-client-1:/weka$ sudo mkdir cassandra-3 cassandra-4 cassandra-5
```

For each of these directories, create an additional persistent volume claim to match:

```
alex [ ~/cassandra/new ]$ cat pvc-cassandra-3.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-wekafs-dir-static-cassandra-3
spec:
  storageClassName: storageclass-wekafs-dir-static
```

```
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
  capacity:
    storage: 5Ti
  csi:
    driver: csi.weka.io
    # volumeHandle must be formatted as following:
    # dir/v1/<FILE_SYSTEM_NAME>/<INNER_PATH_IN_FILESYSTEM>
    # The path must exist, otherwise publish request will fail
    volumeHandle: dir/v1/default/cassandra-3
alex [ ~/cassandra/new ]$ kubectl apply -f pvc-cassandra-3.yaml
persistentvolume/pv-wekafs-dir-static-cassandra-3 created
```

```
alex [ ~/cassandra/new ]$ cat pvc-cassandra-4.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-wekafs-dir-static-cassandra-4
spec:
  storageClassName: storageclass-wekafs-dir-static
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
  capacity:
    storage: 5Ti
  csi:
    driver: csi.weka.io
    # volumeHandle must be formatted as following:
    # dir/v1/<FILE_SYSTEM_NAME>/<INNER_PATH_IN_FILESYSTEM>
    # The path must exist, otherwise publish request will fail
    volumeHandle: dir/v1/default/cassandra-4
alex [ ~/cassandra/new ]$ kubectl apply -f pvc-cassandra-4.yaml
persistentvolume/pv-wekafs-dir-static-cassandra-4 created
```

```
alex [ ~/cassandra/new ]$ cat pvc-cassandra-5.yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-wekafs-dir-static-cassandra-5
spec:
  storageClassName: storageclass-wekafs-dir-static
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  volumeMode: Filesystem
  capacity:
    storage: 5Ti
  csi:
    driver: csi.weka.io
    # volumeHandle must be formatted as following:
    # dir/v1/<FILE_SYSTEM_NAME>/<INNER_PATH_IN_FILESYSTEM>
    # The path must exist, otherwise publish request will fail
    volumeHandle: dir/v1/default/cassandra-5
alex [ ~/cassandra/new ]$ kubectl apply -f pvc-cassandra-5.yaml
persistentvolume/pv-wekafs-dir-static-cassandra-5 created
```

Now confirm there is an additional 3 volume claims available:

```
alex [ ~/cassandra/new ]$ kubectl get pv


NAME                                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGE pv-wekafs-dir-static-cassandra-0   5Ti        RWX            Retain           Bound       default/pv-wekafs-dir-
static-cassandra-1   storage pv-wekafs-dir-static-cassandra-1   5Ti        RWX            Retain           Bound
default/pv-wekafs-dir-static-cassandra-2   storage pv-wekafs-dir-static-cassandra-2   5Ti        RWX            Retain
Bound       default/pv-wekafs-dir-static-cassandra-0   storage pv-wekafs-dir-static-cassandra-3   5Ti        RWX
Retain           Available                                          storage pv-wekafs-dir-static-cassandra-4   5Ti
RWX            Retain           Available                                   storage pv-wekafs-dir-static-
cassandra-5   5Ti        RWX            Retain           Available                                           storage
```

Updating the StatefulSet file

The updated file will change the replicas value in the Cassandra StatefulSet from 3 to 6 once it is re-applied:

```
alex [ ~/cassandra/new ]$ cat cassandra-statefulset.weka.yaml
#cassandra-statefulset.yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: cassandra
  labels:
    app: cassandra
spec:
  serviceName: cassandra
  replicas: 6
  selector:
    matchLabels:
      app: cassandra
  template:
    metadata:
      labels:
        app: cassandra
    spec:
      terminationGracePeriodSeconds: 1800
      containers:
        - name: cassandra
          image: cassandra:4.0.7
          env:
          - name: CASSANDRA_SEEDS
            value: "cassandra-0.cassandra.default.svc.cluster.local"
          - name: POD_IP
            valueFrom:
              fieldRef:
                fieldPath: status.podIP
          lifecycle:
            preStop:
              exec:
               command:
               - /bin/sh
               - -c
               - nodetool drain
          readinessProbe:
            exec:
              command:
              - /bin/bash
              - -c
              - if [[ $(nodetool status | grep $POD_IP) == *"UN"* ]] ; then exit 0; else exit 1; fi
            initialDelaySeconds: 35
            timeoutSeconds: 5
          volumeMounts:
          - mountPath: "/var/lib/cassandra"
            name: pv-wekafs-dir-static
  volumeClaimTemplates:
  - metadata:
      name: pv-wekafs-dir-static
    apiVersion: v1
    kind: PersistentVolumeClaim
    spec:
      accessModes: [ "ReadWriteMany" ]
      storageClassName: storageclass-wekafs-dir-static
      volumeMode: Filesystem
      resources:
        requests:
          storage: 5Ti

alex [ ~/cassandra/new ]$ kubectl apply -f cassandra-statefulset.weka.yaml
statefulset.apps/cassandra configured
```

After a few minutes, there should now be a 6-node Cassandra cluster running across 3 k8s worker nodes:

```
alex [ ~/cassandra/new ]$ kubectl exec --stdin --tty cassandra-0 -- nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens  Owns (effective)  Host ID                               Rack
UN  10.0.0.74  96.53 KiB  16      49.0%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1
UN  10.0.0.69  88.83 KiB  16      50.5%             7a811d25-aec1-4ee4-ba3f-925d583e9c27  rack1
UN  10.0.0.19  91.48 KiB  16      48.8%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1
UN  10.0.0.13  74.3 KiB   16      50.7%             ea512878-82de-4a96-b0e5-e34c7c2fb94a  rack1
UN  10.0.0.57  96.46 KiB  16      51.1%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1
UN  10.0.0.84  69.33 KiB  16      49.9%             350fe554-a971-487f-8cc7-bec9bbbf0eb6  rack1
```

The pods are evenly distributed across the 3 nodes, with two pods per node:

```
alex [ ~/cassandra/new ]$ kubectl get pods -o wide


NAME              READY   STATUS    RESTARTS   AGE     IP           NODE                                NOMINATED NODE
 READINESS GATES cassandra-0       1/1     Running   0          32m     10.0.0.74   aks-nodepool1-21820740-vmss000000   <none>
<none> cassandra-1       1/1     Running   0          31m     10.0.0.19   aks-nodepool1-21820740-vmss000001   <none>
<none> cassandra-2       1/1     Running   0          30m     10.0.0.57   aks-nodepool1-21820740-vmss000002   <none>
<none> cassandra-3       1/1     Running   0          5m10s   10.0.0.84   aks-nodepool1-21820740-vmss000000   <none>
<none> cassandra-4       1/1     Running   0          3m40s   10.0.0.13   aks-nodepool1-21820740-vmss000001   <none>
<none> cassandra-5       1/1     Running   0          2m11s   10.0.0.69   aks-nodepool1-21820740-vmss000002   <none>
<none> installer-dhr54   1/1     Running   0          66m     10.0.0.12   aks-nodepool1-21820740-vmss000001   <none>
<none> installer-mwvsw   1/1     Running   0          66m     10.0.0.41   aks-nodepool1-21820740-vmss000002   <none>
<none> installer-qnttp   1/1     Running   0          66m     10.0.0.70   aks-nodepool1-21820740-vmss000000   <none>
<none>
```

Additionally, you can see that all the persistent volumes have been claimed:

```
alex [ ~/cassandra/new ]$ kubectl get pvc
NAME                              STATUS   VOLUME                            CAPACITY   ACCESS MODES   STORAGECLASS                    AG
pv-wekafs-dir-static-cassandra-0   Bound    pv-wekafs-dir-static-cassandra-2   5Ti        RWX            storageclass-wekafs-dir-static   34
pv-wekafs-dir-static-cassandra-1   Bound    pv-wekafs-dir-static-cassandra-0   5Ti        RWX            storageclass-wekafs-dir-static   33
pv-wekafs-dir-static-cassandra-2   Bound    pv-wekafs-dir-static-cassandra-1   5Ti        RWX            storageclass-wekafs-dir-static   31
pv-wekafs-dir-static-cassandra-3   Bound    pv-wekafs-dir-static-cassandra-3   5Ti        RWX            storageclass-wekafs-dir-static   6m
pv-wekafs-dir-static-cassandra-4   Bound    pv-wekafs-dir-static-cassandra-4   5Ti        RWX            storageclass-wekafs-dir-static   5m
pv-wekafs-dir-static-cassandra-5   Bound    pv-wekafs-dir-static-cassandra-5   5Ti        RWX            storageclass-wekafs-dir-static   3m
```

And the directories have been populated on the WEKA filesystem:

```
azureuser@alexh-client-1:~$ ls /weka/cassandra-*
/weka/cassandra-0:
commitlog  data  hints  saved_caches

/weka/cassandra-1:
commitlog  data  hints  saved_caches

/weka/cassandra-2:
commitlog  data  hints  saved_caches

/weka/cassandra-3:
commitlog  data  hints  saved_caches

/weka/cassandra-4:
commitlog  data  hints  saved_caches

/weka/cassandra-5:
commitlog  data  hints  saved_caches
```

Generating sample data

For generating the sample data, a Standard_D32s_v3 client with a mount to the WEKA filesystem will be used:

```
azureuser@alexh-client-1:~$ df -h /weka
Filesystem      Size  Used Avail Use% Mounted on
default          51T   12M   51T   1% /weka
```

GoLang Installation

To generate the data a GoLang application will be used. To install GoLang on this client, follow these simple steps.

```
azureuser@alexh-client-1:~$ wget https://go.dev/dl/go1.20.3.linux-amd64.tar.gz
--2023-04-17 21:08:06--  https://go.dev/dl/go1.20.3.linux-amd64.tar.gz
Resolving go.dev (go.dev)... 216.239.34.21, 216.239.32.21, 216.239.38.21, ...
Connecting to go.dev (go.dev)|216.239.34.21|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://dl.google.com/go/go1.20.3.linux-amd64.tar.gz [following]
--2023-04-17 21:08:06--  https://dl.google.com/go/go1.20.3.linux-amd64.tar.gz
Resolving dl.google.com (dl.google.com)... 172.253.122.190, 172.253.122.91, 172.253.122.93, ...
Connecting to dl.google.com (dl.google.com)|172.253.122.190|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 100142274 (96M) [application/x-gzip]
Saving to: ‘go1.20.3.linux-amd64.tar.gz’

go1.20.3.linux-amd64.tar.gz                     100%[=============================================================================================

2023-04-17 21:08:06 (362 MB/s) - ‘go1.20.3.linux-amd64.tar.gz’ saved [100142274/100142274]

azureuser@alexh-client-1:~$ tar -zxf go1.20.3.linux-amd64.tar.gz
azureuser@alexh-client-1:~$ sudo mv go /usr/local/
azureuser@alexh-client-1:~$ export PATH=/usr/local/go/bin:$PATH
azureuser@alexh-client-1:~$ go version
go version go1.20.3 linux/amd64
```

generate_csv_data app

Create a directory, ~/src/myapp/generate_csv_data:

```
azureuser@alexh-client-1:~$ mkdir -p ~/src/myapp/generate_csv_data
```

Within the generate_csv_data directory, create a file ‘main.go’ with the following contents:

```
package main

import (
  "bufio"
  "fmt"
  "github.com/brianvoe/gofakeit/v6"
  "github.com/google/uuid"
  "log"
  "os"
)

func check(err error) {
  if err != nil {
    log.Fatal(err)
  }
}

func main() {
  gofakeit.Name()
  id, _ := uuid.NewUUID()

  file, err := os.Create(os.Args[1])
  defer file.Close()
  check(err)

  //4MB buffer
  writer := bufio.NewWriterSize(file, 4000000)
  defer writer.Flush()

  // CSV header
  line := fmt.Sprintf("user_id,post_id,content\n")
  _, err = writer.WriteString(line)
  check(err)

  // Create 1 million rows
  for i := 0; i < 1000000; i += 1 {
    id, _ = uuid.NewUUID()
    line = fmt.Sprintf("%s,%s,%s\n", gofakeit.Username(), id, gofakeit.Paragraph(1, 20, 20, "."))
    _, err = writer.WriteString(line)
    check(err)
  }
}
```

Compilation

To compile the application, run the following:

```
azureuser@alexh-client-1:~$ export GOPATH=/home/azureuser/src/
azureuser@alexh-client-1:~$ cd ~/src/myapp/
azureuser@alexh-client-1:~/src/myapp$ go mod init generate_csv_data
go: creating new go.mod: module generate_csv_data
go: to add module requirements and sums:
  go mod tidy
azureuser@alexh-client-1:~/src/myapp$ go mod tidy
go: finding module for package github.com/google/uuid
go: finding module for package github.com/brianvoe/gofakeit/v6
go: downloading github.com/google/uuid v1.3.0
go: downloading github.com/brianvoe/gofakeit/v6 v6.21.0
go: downloading github.com/brianvoe/gofakeit v3.18.0+incompatible
go: found github.com/brianvoe/gofakeit/v6 in github.com/brianvoe/gofakeit/v6 v6.21.0
go: found github.com/google/uuid in github.com/google/uuid v1.3.0
azureuser@alexh-client-1:~/src/myapp$ cd generate_csv_data/
azureuser@alexh-client-1:~/src/myapp/generate_csv_data$ go build
azureuser@alexh-client-1:~/src/myapp/generate_csv_data$ ls -l
total 4768
-rwxrwxr-x 1 azureuser azureuser 4876687 Apr 17 21:21 generate_csv_data
-rw-rw-r-- 1 azureuser azureuser     738 Apr 17 21:12 main.go
```

Running generate_csv_data app

Create a new directory on the WEKA filesystem and generate a single CSV file with 1 million lines:

```
azureuser@alexh-client-1:~/src/myapp/generate_csv_data$ sudo mkdir /weka/csv_files
azureuser@alexh-client-1:~/src/myapp/generate_csv_data$ sudo chown azureuser.azureuser /weka/csv_files
azureuser@alexh-client-1:~/src/myapp/generate_csv_data$ cd /weka/csv_files
azureuser@alexh-client-1:/weka/csv_files$ time ~/src/myapp/generate_csv_data/generate_csv_data file1.csv

real  1m44.066s
user  1m45.779s
sys 0m2.122s

azureuser@alexh-client-1:/weka/csv_files$ ls -lh file1.csv
-rw-rw-r-- 1 azureuser azureuser 2.5G Apr 17 21:24 file1.csv
```

The following will provide an idea of what this data will look like from the first two lines of the CSV:

```
user_id: Generated by the golang faker module post_id: A V1 UUID which includes the time (Cassandra timeuuid) content: The
content of the supposed blog post (A single paragraph with 20 sentences, each sentence with 20 words)
```

```
azureuser@alexh-client-1:/weka/csv_files$ head -n2 file1.csv
user_id,post_id,content
Brown8395,fd7289e5-dd65-11ed-85e7-000d3a99c2f6,Caravan those upstairs up earlier here whichever now ring what host school notice where few
```

Now observe how long it will take if 24 jobs are run concurrently:

```
azureuser@alexh-client-1:/weka/csv_files$ time for i in {1..24}; do echo file$i.csv; done | xargs -I {} -P 24 ~/src/myapp/generate_csv_data/

real    3m1.386s
user    60m29.601s
sys     2m0.626s
azureuser@alexh-client-1:/weka/csv_files$ du -kcs *.csv
2528488 file1.csv
2528516 file10.csv
2528596 file11.csv
2528540 file12.csv
2528460 file13.csv
2528576 file14.csv
2528716 file15.csv
2528480 file16.csv
2528576 file17.csv
2528488 file18.csv
2528512 file19.csv
2528524 file2.csv
2528492 file20.csv
2528496 file21.csv
2528448 file22.csv
2528528 file23.csv
2528592 file24.csv
2528532 file3.csv
2528516 file4.csv
2528544 file5.csv
2528456 file6.csv
2528504 file7.csv
2528592 file8.csv
2528592 file9.csv
60684764        total
```

Notice some write traffic may be seen on WEKA while the generate_csv_data application is running:

```
azureuser@alexh-client-1:/weka/csv_files$ weka status
WekaIO v4.1.0.71 (CLI build 4.1.0.71)

       cluster: weka (d9997c22-d30b-467f-98a7-eeeaeb2c4e1e)
        status: OK (12 backend containers UP, 12 drives UP)
    protection: 4+2
     hot spare: 1 failure domains (2.09 TiB)
 drive storage: 10.47 TiB total
         cloud: connected
       license: Unlicensed

     io status: STARTED 14 days ago (36 io-nodes UP, 264 Buckets UP)
    link layer: Ethernet
       clients: 10 connected
         reads: 0 B/s (0 IO/s)
        writes: 556.92 MiB/s (616 IO/s)
    operations: 616 ops/s
        alerts: 2 active alerts, use `weka alerts` to list them
```

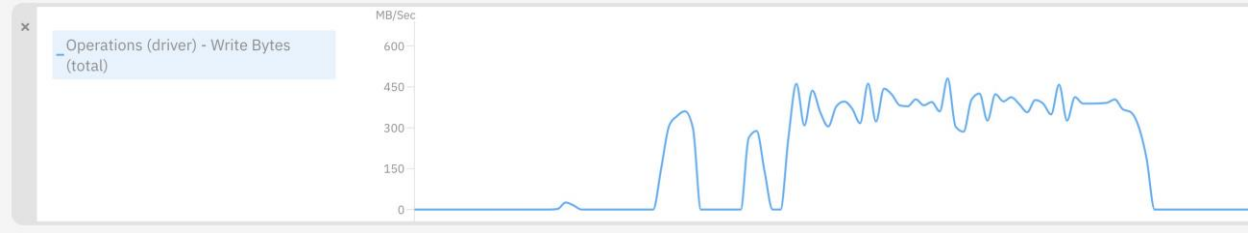
Create 1TB of CSV files

So, about 1TB worth of CSV files were generated in 45 minutes:

```
azureuser@alexh-client-1:/weka/csv_files$ time for i in {1..400}; do echo file$i.csv; done | xargs -I {} -P 24 ~/src/myapp/generate_csv_data
```

```
real    45m25.522s
user    1059m25.694s
sys     31m51.212s

azureuser@alexh-client-1:/weka/csv_files$ du -kcs *.csv | tail -1
1011415716      total
```

The writes can be seen on the WEKA cluster during this time:



Importing the data with dsbulk

To import the data to Cassandra, the datastax's dsbulk tool will be used:

https://docs.datastax.com/en/dsbulk/docs/install/dsbulkInstall.html

Installation

Install the tool on the client:

```
azureuser@alexh-client-1:/weka/csv_files$ curl -OL https://downloads.datastax.com/dsbulk/dsbulk-1.10.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   242  100   242    0     0    720      0 --:--:-- --:--:-- --:--:--    720
100 32.6M  100 32.6M    0     0  18.1M      0  0:00:01  0:00:01 --:--:-- 28.5M
azureuser@alexh-client-1:/weka/csv_files$ tar -zxf dsbulk-1.10.tar.gz
azureuser@alexh-client-1:/weka/csv_files$ ls dsbulk-1.10.0/bin/dsbulk
dsbulk-1.10.0/bin/dsbulk
```

Importing a single CSV file

Now it can be tested how long it will take to load a single, 2.5GB CSV file into the database, but first an IP address will be needed:

```
alex [ ~ ]$ kubectl get pods -o wide

NAME              READY   STATUS    RESTARTS   AGE     IP           NODE                                NOMINATED NODE
READINESS GATES cassandra-0       1/1     Running   0          177m    10.0.0.74   aks-nodepool1-21820740-vmss000000   <none>
<none> cassandra-1       1/1     Running   0          175m    10.0.0.19   aks-nodepool1-21820740-vmss000001   <none>
<none> cassandra-2       1/1     Running   0          174m    10.0.0.57   aks-nodepool1-21820740-vmss000002   <none>
<none> cassandra-3       1/1     Running   0          149m    10.0.0.84   aks-nodepool1-21820740-vmss000000   <none>
<none> cassandra-4       1/1     Running   0          147m    10.0.0.13   aks-nodepool1-21820740-vmss000001   <none>
<none> cassandra-5       1/1     Running   0          146m    10.0.0.69   aks-nodepool1-21820740-vmss000002   <none>
<none>
```

For this example, 10.0.0.74, will be used:

```
azureuser@alexh-client-1:/weka/csv_files$ dsbulk-1.10.0/bin/dsbulk load -cl LOCAL_QUORUM -k blog -t posts -h 10.0.0.74 --connector.csv.url
Operation directory: /weka/csv_files/logs/LOAD_20230417-232056-736146
    total | failed | rows/s | p50ms | p99ms | p999ms | batches
1,000,000 |      0 | 37,373 |  2.83 | 13.30 | 146.80 |     1.00
Operation LOAD_20230417-232056-736146 completed successfully in 26 seconds.
Checkpoints for the current operation were written to checkpoint.csv.
To resume the current operation, re-run it with the same settings, and add the following command line flag:
--dsbulk.log.checkpoint.file=/weka/csv_files/logs/LOAD_20230417-232056-736146/checkpoint.csv
```

After importing a single file, the additional data can be seen in nodetool status:

```
alex [ ~ ]$ kubectl exec --stdin --tty cassandra-0 -- nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  10.0.0.74   748.5 MiB   16      49.0%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1
UN  10.0.0.69   747.91 MiB  16      50.5%             7a811d25-aec1-4ee4-ba3f-925d583e9c27  rack1
UN  10.0.0.19   748.15 MiB  16      48.8%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1
UN  10.0.0.13   748.16 MiB  16      50.7%             ea512878-82de-4a96-b0e5-e34c7c2fb94a  rack1
UN  10.0.0.57   748.19 MiB  16      51.1%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1
UN  10.0.0.84   747.87 MiB  16      49.9%             350fe554-a971-487f-8cc7-bec9bbbf0eb6  rack1
```

Importing 100GB of CSV files

The following will load in approximately 100GB of CSV:

```
for i in file{12..52}; do echo $i.csv; done | xargs -I {} -P 1 time dsbulk-1.10.0/bin/dsbulk load -cl LOCAL_QUORUM -k blog -t posts -h 10.0

....
....
....
Operation directory: /weka/csv_files/logs/LOAD_20230418-001831-878381
    total | failed | rows/s | p50ms | p99ms | p999ms | batches
1,000,000 |      0 | 29,024 |  3.81 | 16.58 | 392.17 |     1.00
Operation LOAD_20230418-001831-878381 completed successfully in 34 seconds.
Checkpoints for the current operation were written to checkpoint.csv.
To resume the current operation, re-run it with the same settings, and add the following command line flag:
--dsbulk.log.checkpoint.file=/weka/csv_files/logs/LOAD_20230418-001831-878381/checkpoint.csv
91.77user 33.01system 0:39.98elapsed 312%CPU (0avgtext+0avgdata 3872020maxresident)k
```

```
5057056inputs+600outputs (0major+870746minor)pagefaults 0swaps Operation directory: /weka/csv_files/logs/LOAD_20230418-001911-
874638     total | failed | rows/s | p50ms | p99ms | p999ms | batches 1,000,000 |      0 | 19,634 |  5.97 | 19.40 | 413.14 |
1.00


Operation LOAD_20230418-001911-874638 completed successfully in 50 seconds.


Checkpoints for the current operation were written to checkpoint.csv.


To resume the current operation, re-run it with the same settings, and add the following command line flag:


--dsbulk.log.checkpoint.file=/weka/csv_files/logs/LOAD_20230418-001911-874638/checkpoint.csv 92.94user 31.76system
0:56.38elapsed 221%CPU (0avgtext+0avgdata 3714568maxresident)k


5057072inputs+600outputs (0major+829899minor)pagefaults 0swaps Operation directory: /weka/csv_files/logs/LOAD_20230418-002008-
296246     total | failed | rows/s | p50ms | p99ms | p999ms | batches 1,000,000 |      0 | 20,815 |  5.66 | 21.23 | 633.34 |
1.00


Operation LOAD_20230418-002008-296246 completed successfully in 47 seconds.


Checkpoints for the current operation were written to checkpoint.csv.


To resume the current operation, re-run it with the same settings, and add the following command line flag:


--dsbulk.log.checkpoint.file=/weka/csv_files/logs/LOAD_20230418-002008-296246/checkpoint.csv 89.23user 32.26system
0:53.59elapsed 226%CPU (0avgtext+0avgdata 3561244maxresident)k


5057072inputs+600outputs (0major+781974minor)pagefaults 0swaps Operation directory: /weka/csv_files/logs/LOAD_20230418-002101-
855394     total | failed | rows/s | p50ms | p99ms | p999ms | batches 1,000,000 |      0 | 19,060 |  6.18 | 21.36 | 446.69 |
1.00


Operation LOAD_20230418-002101-855394 completed successfully in 52 seconds.


Checkpoints for the current operation were written to checkpoint.csv.


To resume the current operation, re-run it with the same settings, and add the following command line flag:


--dsbulk.log.checkpoint.file=/weka/csv_files/logs/LOAD_20230418-002101-855394/checkpoint.csv 90.16user 32.62system
0:57.99elapsed 211%CPU (0avgtext+0avgdata 3651744maxresident)k


5057184inputs+600outputs (0major+794525minor)pagefaults 0swaps
```

Again, nodetool status will reflect the data added so far:

```
root@cassandra-0:/# nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens  Owns (effective)  Host ID                               Rack
UN  10.0.0.74  40.11 GiB  16      49.0%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1
UN  10.0.0.69  41.52 GiB  16      50.5%             7a811d25-aec1-4ee4-ba3f-925d583e9c27  rack1
UN  10.0.0.19  40.12 GiB  16      48.8%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1
UN  10.0.0.13  41.58 GiB  16      50.7%             ea512878-82de-4a96-b0e5-e34c7c2fb94a  rack1
UN  10.0.0.57  41.91 GiB  16      51.1%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1
UN  10.0.0.84  40.83 GiB  16      49.9%             350fe554-a971-487f-8cc7-bec9bbbf0eb6  rack1
```

Query the Data

With a simple Python script, the data can be viewed:

```
#!/usr/bin/python3
from cassandra.cluster import Cluster, Session

cluster = Cluster(['10.0.0.19'], port=9042)
session = cluster.connect()

query = session.execute("SELECT * from blog.posts where user_id = 'Douglas4214' limit 5").all()

for row in query:
    print(row[0],row[1],row[2][0:100])
```

```
azureuser@alexh-client-1:~$ ./c_query.py


Douglas4214 b5840af2-dd6a-11ed-b26c-000d3a99c2f6 He vivaciously you as on orange anxious over ours how ahead formerly do being
monthly where


Douglas4214 ae87ba0f-dd6a-11ed-9811-000d3a99c2f6 In we her way from include sunshine yours crowded whatever i.e. then hail
walk who next sta


Douglas4214 ac81a541-dd6a-11ed-9fde-000d3a99c2f6 Other throw therefore could both despite before formerly yesterday i.e. when
alternatively


Douglas4214 99f9a976-dd6a-11ed-98a9-000d3a99c2f6 Place boy that fact it which somebody this though where acknowledge nobody
itself indeed w


Douglas4214 9025d1fb-dd6a-11ed-b08f-000d3a99c2f6 His no significant their with her production one yours one cackle constantly
example plate
```

Scaling the AKS cluster

Using the az aks scale command

As you may recall, currently the Cassandra cluster is running with 6-nodes or 6 pods on 3 k8s nodes. In this next example the AKS cluster will be expanded from 3 k8s nodes to 6 k8s nodes. This can easily be done from the Azure portal console:

```
alex [ ~ ]$ az aks scale -n example-k8s-cluster --resource-group weka-example-rg --node-count 6

...
...

alex [ ~ ]$ kubectl get nodes
NAME                                STATUS   ROLES   AGE     VERSION
aks-nodepool1-21820740-vmss000000   Ready    agent   5h45m   v1.24.9
aks-nodepool1-21820740-vmss000001   Ready    agent   5h46m   v1.24.9
aks-nodepool1-21820740-vmss000002   Ready    agent   5h45m   v1.24.9
aks-nodepool1-21820740-vmss000003   Ready    agent   4m45s   v1.24.9
aks-nodepool1-21820740-vmss000004   Ready    agent   4m28s   v1.24.9
aks-nodepool1-21820740-vmss000005   Ready    agent   4m45s   v1.24.9
```

Moving Pods

Now the pods cassandra-3, cassandra-4 and cassandra-5 may be moved to these new worker nodes, deleting them one by one, ensuring the re-created pod appears on one of the new nodes and is in a ready status before moving to the next one:

```
alex [ ~ ]$ kubectl delete pod cassandra-3 pod "cassandra-3" deleted alex [ ~ ]$ kubectl get pods -o wide


NAME              READY   STATUS    RESTARTS   AGE     IP           NODE                                NOMINATED NODE
READINESS GATES cassandra-0       1/1     Running   0          4h33m   10.0.0.74    aks-nodepool1-21820740-vmss000000
<none>          <none> cassandra-1       1/1     Running   0          4h32m   10.0.0.19    aks-nodepool1-21820740-vmss000001
<none>          <none> cassandra-2       1/1     Running   0          4h30m   10.0.0.57    aks-nodepool1-21820740-vmss000002
<none>          <none> cassandra-3       0/1     Running   0          15s     10.0.0.184   aks-nodepool1-21820740-vmss000003
<none>          <none> cassandra-4       1/1     Running   0          4h4m    10.0.0.13    aks-nodepool1-21820740-vmss000001
<none>          <none> cassandra-5       1/1     Running   0          4h2m    10.0.0.69    aks-nodepool1-21820740-vmss000002
<none>          <none>
```

```
alex [ ~ ]$ kubectl exec --stdin --tty cassandra-3 -- nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack
UN  10.0.0.74   40.11 GiB  16      49.0%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1
UN  10.0.0.184  41.07 GiB  16      49.9%             350fe554-a971-487f-8cc7-bec9bbbf0eb6  rack1
UN  10.0.0.69   41.52 GiB  16      50.5%             7a811d25-aec1-4ee4-ba3f-925d583e9c27  rack1
UN  10.0.0.19   40.12 GiB  16      48.8%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1
UN  10.0.0.13   41.58 GiB  16      50.7%             ea512878-82de-4a96-b0e5-e34c7c2fb94a  rack1
UN  10.0.0.57   41.91 GiB  16      51.1%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1
```

```
alex [ ~ ]$ kubectl get pods -o wide


NAME              READY   STATUS    RESTARTS   AGE     IP           NODE                                NOMINATED NODE
READINESS GATES cassandra-0       1/1     Running   0          4h37m   10.0.0.74    aks-nodepool1-21820740-vmss000000
<none>           <none> cassandra-1       1/1     Running   0          4h36m   10.0.0.19    aks-nodepool1-21820740-vmss000001
<none>           <none> cassandra-2       1/1     Running   0          4h34m   10.0.0.57    aks-nodepool1-21820740-vmss000002
<none>           <none> cassandra-3       1/1     Running   0          4m10s   10.0.0.184   aks-nodepool1-21820740-vmss000003
<none>           <none> cassandra-4       1/1     Running   0          61s     10.0.0.109   aks-nodepool1-21820740-vmss000004
<none>           <none> cassandra-5       1/1     Running   0          4h6m    10.0.0.69    aks-nodepool1-21820740-vmss000002
<none>           <none> installer-7p671   1/1     Running   0          11m     10.0.0.157   aks-nodepool1-21820740-vmss000003
<none>           <none> installer-dhr54   1/1     Running   0          5h10m   10.0.0.12    aks-nodepool1-21820740-vmss000001
<none>           <none> installer-lnrxg   1/1     Running   0          11m     10.0.0.99    aks-nodepool1-21820740-vmss000004
<none>           <none> installer-mwvsw   1/1     Running   0          5h10m   10.0.0.41    aks-nodepool1-21820740-vmss000002
<none>           <none> installer-qnttp   1/1     Running   0          5h10m   10.0.0.70    aks-nodepool1-21820740-vmss000000
<none>           <none> installer-zlb79   1/1     Running   0          11m     10.0.0.128   aks-nodepool1-21820740-vmss000005
<none>           <none> alex [ ~ ]$ kubectl exec --stdin --tty cassandra-4 -- nodetool status


Datacenter: datacenter1


======================


Status=Up/Down


|/ State=Normal/Leaving/Joining/Moving


--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack


UN  10.0.0.74   40.11 GiB  16      49.0%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1


UN  10.0.0.184  41.07 GiB  16      49.9%             350fe554-a971-487f-8cc7-bec9bbbf0eb6  rack1


UN  10.0.0.109  41.7 GiB   16      50.7%             ea512878-82de-4a96-b0e5-e34c7c2fb94a  rack1


UN  10.0.0.69   41.52 GiB  16      50.5%             7a811d25-aec1-4ee4-ba3f-925d583e9c27  rack1


UN  10.0.0.19   40.12 GiB  16      48.8%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1 UN  10.0.0.57   41.91 GiB  16
51.1%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1
```

```
alex [ ~ ]$ kubectl get pods -o wide


NAME              READY   STATUS    RESTARTS   AGE     IP           NODE                                NOMINATED NODE
 READINESS GATES cassandra-0       1/1     Running   0          4h39m   10.0.0.74    aks-nodepool1-21820740-vmss000000
<none>           <none> cassandra-1       1/1     Running   0          4h38m   10.0.0.19    aks-nodepool1-21820740-vmss000001
<none>           <none> cassandra-2       1/1     Running   0          4h36m   10.0.0.57    aks-nodepool1-21820740-vmss000002
<none>           <none> cassandra-3       1/1     Running   0          6m4s    10.0.0.184   aks-nodepool1-21820740-vmss000003
<none>           <none> cassandra-4       1/1     Running   0          2m55s   10.0.0.109   aks-nodepool1-21820740-vmss000004
<none>           <none> cassandra-5       1/1     Running   0          50s     10.0.0.131   aks-nodepool1-21820740-vmss000005
<none>           <none> installer-7p671   1/1     Running   0          13m     10.0.0.157   aks-nodepool1-21820740-vmss000003
<none>           <none> installer-dhr54   1/1     Running   0          5h12m   10.0.0.12    aks-nodepool1-21820740-vmss000001
<none>           <none> installer-lnrxg   1/1     Running   0          13m     10.0.0.99    aks-nodepool1-21820740-vmss000004
<none>           <none> installer-mwvsw   1/1     Running   0          5h12m   10.0.0.41    aks-nodepool1-21820740-vmss000002
<none>           <none> installer-qnttp   1/1     Running   0          5h12m   10.0.0.70    aks-nodepool1-21820740-vmss000000
<none>           <none> installer-zlb79   1/1     Running   0          13m     10.0.0.128   aks-nodepool1-21820740-vmss000005
<none>           <none> alex [ ~ ]$ kubectl exec --stdin --tty cassandra-5 -- nodetool status


Datacenter: datacenter1


=======================


Status=Up/Down


|/ State=Normal/Leaving/Joining/Moving


--  Address     Load       Tokens  Owns (effective)  Host ID                               Rack


UN  10.0.0.74   40.11 GiB  16      49.0%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1


UN  10.0.0.184  41.07 GiB  16      49.9%             350fe554-a971-487f-8cc7-bec9bbbf0eb6  rack1


UN  10.0.0.109  41.7 GiB   16      50.7%             ea512878-82de-4a96-b0e5-e34c7c2fb94a  rack1


UN  10.0.0.19   40.12 GiB  16      48.8%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1


UN  10.0.0.131  41.55 GiB  16      50.5%             7a811d25-aec1-4ee4-ba3f-925d583e9c27  rack1


UN  10.0.0.57   41.91 GiB  16      51.1%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1
```

Note: You can see each pod spins up quickly, between approximately 50-60 seconds until re-joined to the cluster.

You can see how easily it was to re-balance the Cassandra nodes across the additional AKS worker nodes without needing to bootstrap additional Cassandra hosts. Now that the scaling is complete, you will see some improvement in the import process (47,325 rows/s vs 37,373) :

```
azureuser@alexh-client-1:/weka/csv_files$ dsbulk-1.10.0/bin/dsbulk load -cl LOCAL_QUORUM -k blog -t posts -h 10.0.0.74 --connector.csv.url
Operation directory: /weka/csv_files/logs/LOAD_20230418-010748-328770
    total | failed | rows/s | p50ms | p99ms | p999ms | batches
1,000,000 |      0 | 47,325 |  1.97 |  4.59 |  87.03 |     1.00
Operation LOAD_20230418-010748-328770 completed successfully in 20 seconds.
Checkpoints for the current operation were written to checkpoint.csv.
To resume the current operation, re-run it with the same settings, and add the following command line flag:
--dsbulk.log.checkpoint.file=/weka/csv_files/logs/LOAD_20230418-010748-328770/checkpoint.csv
```

Import the remaining files

The results of 'nodetool status' after having imported the entire 1TB worth of CSV files:

```
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  10.0.0.82   317.42 GiB  16      50.7%             ea512878-82de-4a96-b0e5-e34c7c2fb94a  rack1
UN  10.0.0.54   312.62 GiB  16      49.9%             350fe554-a971-487f-8cc7-bec9bbbf0eb6  rack1
UN  10.0.0.120  325.26 GiB  16      51.1%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1
UN  10.0.0.13   306.66 GiB  16      49.0%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1
UN  10.0.0.166  312 GiB     16      48.8%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1
UN  10.0.0.129  322.4 GiB   16      50.5%             7a811d25-aec1-4ee4-ba3f-925d583e9c27  rack1
```

Backup/Restore of Cassandra on AKS using WEKA:

First step is to issue a nodetool flush on all of the Cassandra nodes via the Azure portal console:

```
alex [ ~/cassandra ]$ for i in {0..5}; do kubectl exec --stdin --tty cassandra-$i -- nodetool flush ; done
```

Snapshot + Upload

Next, create a snapshot on WEKA:

```
weka@example-weka-backend-0:~$ weka fs snapshot create default cassandra-snap1
SNAPSHOT ID  NAME             ACCESS POINT              IS WRITABLE  CREATION TIME
8            cassandra-snap1  @GMT-2023.04.19-13.21.55  False         2023-04-19T13:21:55
```

Upload this snapshot to the blob object store:

```
weka@example-weka-backend-0:~$ weka fs snapshot upload default cassandra-snap1
Snapshot upload has started as CWTaskId: 6. The object store locator is:

    d9997c22/d/s/7/spec/d30b-467f-98a7-eeeaeb2c4e1e

You can use this locator to restore the filesystem using the "weka fs download" command.
To see the upload progress use the "weka fs snapshot" command.
```

Check the status to ensure it has been uploaded successfully:

```
weka@example-weka-backend-0:~$ weka fs snapshot
SNAPSHOT ID  FILESYSTEM  NAME             ACCESS POINT              IS WRITABLE  CREATION TIME        LOCAL OBJECT STATUS  LOCAL OBJECT PROG
8            default     cassandra-snap1  @GMT-2023.04.19-13.21.55  False        2023-04-19T13:21:55  SYNCHRONIZED         100%
```

The upload to the object store can be viewed in the WEKA statistics as well:

Destroying the environment

To simulate a catastrophic failure both the AKS and WEKA environments will be destroyed:

1. Delete the AKS cluster:

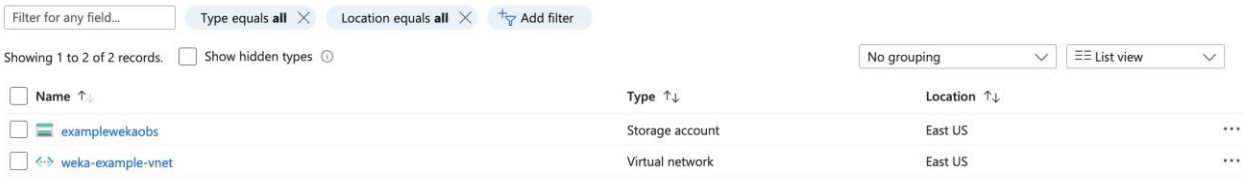
```
alex [ ~/cassandra ]$ az aks delete -g weka-example-rg -n example-k8s-cluster
The behavior of this command has been altered by the following extension: aks-preview
```

```
Are you sure you want to perform this operation? (y/n): y
```

2. Delete the WEKA cluster:

```
terraform destroy -var-file vars.auto.tfvars -var='get_weka_io_token=<get_token>' -var='subscription_id=<sub_id>'

...
...
...

Destroy complete! Resources: 93 destroyed.
```

After that, the original resource group contains just a few items:



Creating a New Environment

In this section, new WEKA and AKS environments will be created using the same Cassandra data that was captured in the snapshot.

Using Terraform, create a new WEKA cluster:

```
$ terraform apply -var-file vars.auto.tfvars -var='get_weka_io_token=<get_token>' -var='subscription_id=<subscription_id>'
...
...
...
DOWNLOAD-SSH-KEYS-COMMAND = <<EOT
######################################## download ssh keys command from blob ############################################################
  CLUSTER: weka
  az keyvault secret download --file private.pem --encoding utf-8 --vault-name  example-weka-key-vault --name private-key --query "value"
  az keyvault secret download --file public.pub --encoding utf-8 --vault-name  example-weka-key-vault --name public-key --query "value"

EOT
IPS = [
  "example-weka-backend-0: 104.45.142.3",
  "example-weka-backend-1: 104.45.139.164",
  "example-weka-backend-2: 104.45.139.181",
  "example-weka-backend-3: 104.45.139.228",
  "example-weka-backend-4: 104.45.139.219",
  "example-weka-backend-5: 104.45.142.132",
]
SSH-KEY-PATH = "/tmp/example-weka-public-key.pub, /tmp/example-weka-private-key.pem"
get-cluster-status = <<EOT
######################################## get cluster status ##############################################################################
function_key=$(az functionapp keys list --name example-weka-function-app --resource-group weka-example-rg --subscription <subscription_id>
curl https://example-weka-function-app.azurewebsites.net/api/status?code=$function_key

EO
```

After having created a new WEKA cluster, the new system is up and running with an empty, default filesystem:

```
weka@example-weka-backend-0:~$ weka fs
FILESYSTEM ID  FILESYSTEM NAME  USED SSD  AVAILABLE SSD  USED TOTAL  AVAILABLE TOTAL  THIN PROVISIONED  THIN PROVISIONED MINIMUM SSD  THIN P
0              default          4.09 KB   11.51 TB       4.09 KB     57.58 TB         False
```

Restore from snapshot

Now that the system is up and running the first step in the recovery process is to delete the “default” filesystem:

```
weka@example-weka-backend-0:~$ weka fs delete default
You are about to delete a filesystem. This action DELETES ALL DATA in the filesystem and cannot be undone.
Are you sure you want to continue (yes/no)? yes
```

Next, create a new filesystem called default and attach the snapshot that was uploaded earlier:

```
weka@example-weka-backend-0:~$ weka fs download default default 60TB 1TB azure-obs d9997c22/d/s/7/spec/d30b-467f-98a7-eeeaeb2c4e1e
FSId: 1

weka@example-weka-backend-0:~$ weka fs
FILESYSTEM ID  FILESYSTEM NAME  USED SSD  AVAILABLE SSD  USED TOTAL  AVAILABLE TOTAL  THIN PROVISIONED  THIN PROVISIONED MINIMUM SSD  THIN P
1              default          12.35 GB  11 TB          3.16 TB     60 TB            False
```

Mounting the WEKA filesystem to a client

From the client, install the WEKA client:

```
azureuser@alexh-client-1:~$ curl http://10.0.0.9:14000/dist/v1/install | sudo sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1413  100  1413    0     0   689k      0 --:--:-- --:--:-- --:--:--  689k
Downloading WekaIO CLI 4.1.0.71
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 92.4M  100 92.4M    0     0   840M      0 --:--:-- --:--:-- --:--:--  840M
Installing...
Installing agent of version 4.1.0.71
Waiting for agent service to be ready
```

```
Installation finished successfully
WekaIO CLI 4.1.0.71 is now installed
```

After mounting the WEKA filesystem, the contents from the snapshot taken earlier can be seen:

```
azureuser@alexh-client-1:~$ sudo mount -t wekafs -onet=udp,num_cores=4 10.0.0.9/default /weka

Mounting 10.0.0.9/default on /weka

Basing mount on container client

Downloading [1/19] http://10.0.0.9:14000/dist/v1/image/weka-node-7dd65bb3b9240a695217fe1e98f1c96d.squashfs

Downloading [2/19] http://10.0.0.9:14000/dist/v1/image/container-samba-weka-4.7.12.3-95d7e0704b141138b43d76b226f55eda.squashfs

Downloading [3/19] http://10.0.0.9:14000/dist/v1/image/dependencies-1.0.0-81dd3682be55880a5ef2b337bce9ae4d.squashfs

Downloading [4/19] http://10.0.0.9:14000/dist/v1/image/weka-ganesha-18ca10b28151817a7c8bb7267e6f5e9d.squashfs

Downloading [5/19] http://10.0.0.9:14000/dist/v1/image/weka-samba-fa23b1ed5b86200b1d682f072bc9af61.squashfs

Downloading [6/19] http://10.0.0.9:14000/dist/v1/image/driver-uio-pci-generic-1.0.0-d644841c998c88e4fc66529e4484dbb6.squashfs

Downloading [7/19] http://10.0.0.9:14000/dist/v1/image/api-52da22b2cd0ed1b50978e57d509a47c6.squashfs

Downloading [8/19] http://10.0.0.9:14000/dist/v1/image/weka-driver-1.0.0-595420f10959c344dc93b1eff50bb016.squashfs

Downloading [9/19] http://10.0.0.9:14000/dist/v1/image/weka-smbw-b038d6b57ae8f593e98282b2709b69b1.squashfs

Downloading [10/19] http://10.0.0.9:14000/dist/v1/image/container-s3-weka-release-459ba907ea2ca2fc6c3940deba2dc631.squashfs
```

```
Downloading [11/19] http://10.0.0.9:14000/dist/v1/image/weka-container-2.3.0-9c60adc0f77d96f577211d30cbf9ef3c.squashfs
Downloading [12/19] http://10.0.0.9:14000/dist/v1/image/container-ganesha-dev-weka-2-
2a9043bc5a3669e6d3498a134575e134.squashfs


Downloading [13/19] http://10.0.0.9:14000/dist/v1/image/ofed-1b295470b56ec067af7340f2cca7e27a.squashfs


Downloading [14/19] http://10.0.0.9:14000/dist/v1/image/weka-s3-152101ca875b1b3e68598a8436da01ba.squashfs


Downloading [15/19] http://10.0.0.9:14000/dist/v1/image/weka-hostside-be7022b2fb8d07e029e3ae414452a3e5.squashfs


Downloading [16/19] http://10.0.0.9:14000/dist/v1/image/dashboard-8dbfa0f8a99260cd06825181ea0eb6d6.squashfs


Downloading [17/19] http://10.0.0.9:14000/dist/v1/image/ui-1.0.0-5bc747765d326e6e1c3488285822f459.squashfs


Downloading [18/19] http://10.0.0.9:14000/dist/v1/image/weka-driver-igb-uio-4.0.0-b8dc002ff96443389fdef3f08462b238.squashfs


Downloading [19/19] http://10.0.0.9:14000/dist/v1/image/container-smbw-weka-4.7.12.3-2f44067d8868530d8f7e86d035fbb7a8.squashfs


Finished getting version 4.1.0.71


Creating Weka container 'client' in version 4.1.0.71


Applying resources


Starting container 'client' Waiting for container 'client' to join cluster client: Allocated core 4 to slot 2 on "alexh-
client-1":"client" (1/4) client: Allocated core 2 to slot 1 on "alexh-client-1":"client" (2/4) client: Allocated core 8 to
slot 4 on "alexh-client-1":"client" (3/4) client: Allocated core 6 to slot 3 on "alexh-client-1":"client" (4/4) client:
Starting hugepages allocation for "alexh-client-1":"client" client: Container "alexh-client-1":"client" allocated 2812 out of
2812 required hugepages after 1 retries client: Allocated 5624MB hugepages memory from 1 NUMA nodes for "alexh-client-
1":"client" client: Bandwidth of "alexh-client-1":"client" set to unlimited


client: WekaFS driver attached by "NodeId<65534>" on "alexh-client-1":"client" Container "client" is ready (pid = 11974)


Calling the mount command Mount completed successfully azureuser@alexh-client-1:~$ ls /weka/


cassandra-0  cassandra-1  cassandra-2  cassandra-3  cassandra-4  cassandra-5  csv_files
```

Re-create the AKS Cluster

Now that the snapshot has been restored, the k8s may be re-created. This will require 6 worker nodes to correspond to the 6 Cassandra pods that will be running:

```
az aks create -g weka-example-rg -n example-k8s-cluster \
 --kubernetes-version 1.24.9 --location eastus -p example-dns -k 1.24.9 \
--network-plugin azure --auto-upgrade-channel none --node-resource-group \
MC_weka-example-rg_weka_example_eastus --enable-node-public-ip \
--vnet-subnet-id /subscriptions/<subscription_id>/resourceGroups/weka-example-rg/providers/Microsoft.Network/virtualNetworks/weka-example-v
--service-cidr 10.10.0.0/16 --dns-service-ip 10.10.0.10 \
--docker-bridge-address 172.17.0.1/16 --node-vm-size Standard_D16s_v3 \
 --node-count 6
...
```

```
...
"systemData": null,
  "tags": null,
  "type": "Microsoft.ContainerService/ManagedClusters",
  "upgradeSettings": null,
  "windowsProfile": {
    "adminPassword": null,
    "adminUsername": "azureuser",
    "enableCsiProxy": true,
    "gmsaProfile": null,
    "licenseType": null
  },
  "workloadAutoScalerProfile": {
    "keda": null,
    "verticalPodAutoscaler": null
  }
}
```

The AKS credentials will be needed and the old ones overwritten:

```
alex [ ~/cassandra ]$ az aks get-credentials --resource-group weka-example-rg --name example-k8s-cluster
The behavior of this command has been altered by the following extension: aks-preview
A different object named example-k8s-cluster already exists in your kubeconfig file.
Overwrite? (y/n): y
A different object named clusterUser_weka-example-rg_example-k8s-cluster already exists in your kubeconfig file.
Overwrite? (y/n): y
Merged "example-k8s-cluster" as current context in /home/alex/.kube/config
```

Next, the WEKA CSI plugin will be installed using helm:

```
alex [ ~/cassandra/csi ]$ helm install csi-wekafs csi-wekafs/csi-wekafsplugin --namespace csi-wekafs --create-namespace
NAME: csi-wekafs
LAST DEPLOYED: Fri Mar 31 20:37:33 2023
NAMESPACE: csi-wekafs
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Thank you for installing csi-wekafsplugin.

Your release is named csi-wekafs.
The release is installed in namespace csi-wekafs

To learn more about the release, try:

  $ helm status -n csi-wekafs csi-wekafs
  $ helm get all -n csi-wekafs csi-wekafs

Official Weka CSI Plugin documentation can be found here: https://docs.weka.io/appendix/weka-csi-plugin

Examples on how to configure a storage class and start using the driver are here:
https://github.com/weka/csi-wekafs/tree/master/examples
```

At this point the AKS cluster should be up and running with the CSI plugin:

```
alex [ ~/cassandra ]$ kubectl get nodes
NAME                                STATUS   ROLES   AGE     VERSION
aks-nodepool1-57956347-vmss000000   Ready    agent   7m3s    v1.24.9
aks-nodepool1-57956347-vmss000001   Ready    agent   6m46s   v1.24.9
aks-nodepool1-57956347-vmss000002   Ready    agent   7m3s    v1.24.9
aks-nodepool1-57956347-vmss000003   Ready    agent   7m1s    v1.24.9
aks-nodepool1-57956347-vmss000004   Ready    agent   6m58s   v1.24.9
aks-nodepool1-57956347-vmss000005   Ready    agent   6m52s   v1.24.9

alex [ ~/cassandra ]$ kubectl get pods -n csi-wekafs
NAME                         READY   STATUS    RESTARTS   AGE
csi-wekafs-controller-0   5/5     Running   0          110s
csi-wekafs-node-5n4w5     3/3     Running   0          110s
csi-wekafs-node-d8hzh     3/3     Running   0          110s
csi-wekafs-node-sw9kz     3/3     Running   0          110s
csi-wekafs-node-trfvd     3/3     Running   0          110s
csi-wekafs-node-vxl28     3/3     Running   0          110s
csi-wekafs-node-zlbv6     3/3     Running   0          110s
```

As before, the secrets.yaml file will need to be updated. Be sure and change the default password on WEKA (you can easily do this by browsing to port 14000 on any of the cluster backend nodes. In this case, the WEKA password was changed to be the same one so the only thing that needs to change are the endpoints:

```
alex [ ~/cassandra/csi ]$ echo -n 10.0.0.7:14000,10.0.0.8:14000,10.0.0.9:14000 | base64
MTAuMC4wLjc6MTQwMDAsMTAuMC4wLjg6MTQwMDAsMTAuMC4wLjk6MTQwMDA=
alex [ ~/cassandra/csi ]$ cat secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: csi-wekafs-api-secret
  namespace: csi-wekafs
type: Opaque
data:
  username: YWRtaW4=
  password: V2VrYS5pbzEyMw==
  organization: Um9vdA==
  endpoints: MTAuMC4wLjc6MTQwMDAsMTAuMC4wLjg6MTQwMDAsMTAuMC4wLjk6MTQwMDA=
  scheme: aHR0cA==
```

Going back to the daemonset configuration, the configMap will need to be updated with the new addresses of the new cluster. In this case:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: sample-installer-config
  namespace: default
data:
  install.sh: |
    #!/bin/bash
    # Update and install packages
    mkdir /weka
    ssh-keygen -t rsa -N '' -f ~/.ssh/id_rsa <<< y
    cat ~/.ssh/id_rsa.pub >> ~azureuser/.ssh/authorized_keys
    ip addr
    ssh -i ~/.ssh/id_rsa -o "StrictHostKeyChecking no" azureuser@localhost "curl http://10.0.0.9:14000/dist/v1/install | sudo sh"
    ssh -i ~/.ssh/id_rsa -o "StrictHostKeyChecking no" azureuser@localhost "sudo mount -t wekafs -o net=udp,num_cores=1 10.0.0.7,10.0.0.8,10
```

After making those changes, simply apply the daemonset as previously done:

```
alex [ ~ ]$ kubectl apply -f k8s
configmap/sample-installer-config created
daemonset.apps/installer created
alex [ ~ ]$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
installer-4m775   1/1     Running   0          5s
installer-cbzmc   1/1     Running   0          5s
installer-dp9cj   1/1     Running   0          5s
installer-hhkhp   1/1     Running   0          5s
installer-hpjrr   1/1     Running   0          5s
installer-lxxhq   1/1     Running   0          5s
```

While those are coming online (mounting the WEKA filesystem to the worker nodes) we can check on the status of one of them to verify it is working properly:

```
alex [ ~ ]$ kubectl logs installer-4m775 | tail
```

```
Downloading [10/19] http://10.0.0.7:14000/dist/v1/image/container-s3-weka-release-459ba907ea2ca2fc6c3940deba2dc631.squashfs

Downloading [11/19] http://10.0.0.7:14000/dist/v1/image/weka-container-2.3.0-9c60adc0f77d96f577211d30cbf9ef3c.squashfs
Downloading [12/19] http://10.0.0.7:14000/dist/v1/image/container-ganesha-dev-weka-2-
2a9043bc5a3669e6d3498a134575e134.squashfs

Downloading [13/19] http://10.0.0.7:14000/dist/v1/image/ofed-1b295470b56ec067af7340f2cca7e27a.squashfs

Downloading [14/19] http://10.0.0.7:14000/dist/v1/image/weka-s3-152101ca875b1b3e68598a8436da01ba.squashfs

Downloading [15/19] http://10.0.0.7:14000/dist/v1/image/weka-hostside-be7022b2fb8d07e029e3ae414452a3e5.squashfs

Downloading [16/19] http://10.0.0.7:14000/dist/v1/image/dashboard-8dbfa0f8a99260cd06825181ea0eb6d6.squashfs

Downloading [17/19] http://10.0.0.7:14000/dist/v1/image/ui-1.0.0-5bc747765d326e6e1c3488285822f459.squashfs

Downloading [18/19] http://10.0.0.7:14000/dist/v1/image/weka-driver-igb-uio-4.0.0-b8dc002ff96443389fdef3f08462b238.squashfs

Downloading [19/19] http://10.0.0.7:14000/dist/v1/image/container-smbw-weka-4.7.12.3-2f44067d8868530d8f7e86d035fbb7a8.squashfs
```

After a few moments the client should be able to mount the filesystem successfully on each of the daemonset pods (each k8s worker node):

```
alex [ ~ ]$ for i in `kubectl get pods | grep installer | awk '{print $1;}'`; do kubectl logs $i | tail -1; done
Mount completed successfully
Mount completed successfully
Mount completed successfully
Mount completed successfully
Mount completed successfully
Mount completed successfully
```

Now, once again create a storage class:

```
alex [ ~/cassandra/new ]$ cat sc.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: storageclass-wekafs-dir-static
provisioner: csi.weka.io
reclaimPolicy: Retain
volumeBindingMode: Immediate
allowVolumeExpansion: true
parameters:
  volumeType: dir/v1
  filesystemName: default
```

```
capacityEnforcement: HARD


# optional parameters setting UID, GID and permissions on volume


# UID of the volume owner, default 0 (root)


#ownerUid: "1000"


# GID of the volume owner, default 0 (root)


#ownerGid: "1000"


# permissions in Unix octal format, default "0750"


#permissions: "0775"


  # name of the secret that stores API credentials for a cluster   # change the name of secret to match secret of a particular
cluster (if you have several Weka clusters)   csi.storage.k8s.io/provisioner-secret-name: &secretName csi-wekafs-api-secret
# change the name of the namespace in which the cluster API credentials   csi.storage.k8s.io/provisioner-secret-namespace:
&secretNamespace csi-wekafs   # do not change anything below this line, or set to same parameters as above
csi.storage.k8s.io/controller-publish-secret-name: *secretName   csi.storage.k8s.io/controller-publish-secret-namespace:
*secretNamespace   csi.storage.k8s.io/controller-expand-secret-name: *secretName   csi.storage.k8s.io/controller-expand-
secret-namespace: *secretNamespace   csi.storage.k8s.io/node-stage-secret-name: *secretName   csi.storage.k8s.io/node-stage-
secret-namespace: *secretNamespace   csi.storage.k8s.io/node-publish-secret-name: *secretName   csi.storage.k8s.io/node-
publish-secret-namespace: *secretNamespace alex [ ~/cassandra/new ]$ cat sc.yaml ^C alex [ ~/cassandra/new ]$ kubectl apply -
f sc.yaml  storageclass.storage.k8s.io/storageclass-wekafs-dir-static created alex [ ~/cassandra/new ]$ kubectl get sc


NAME                            PROVISIONER          RECLAIMPOLICY   VOLUMEBINDINGMODE      ALLOWVOLUMEEXPANSION   AGE
azurefile                       file.csi.azure.com   Delete          Immediate              true                   17m
azurefile-csi                   file.csi.azure.com   Delete          Immediate              true                   17m
azurefile-csi-premium           file.csi.azure.com   Delete          Immediate              true                   17m
azurefile-premium               file.csi.azure.com   Delete          Immediate              true                   17m
default (default)               disk.csi.azure.com   Delete          WaitForFirstConsumer   true                   17m
managed                         disk.csi.azure.com   Delete          WaitForFirstConsumer   true                   17m
managed-csi                     disk.csi.azure.com   Delete          WaitForFirstConsumer   true                   17m
managed-csi-premium             disk.csi.azure.com   Delete          WaitForFirstConsumer   true                   17m
managed-premium                 disk.csi.azure.com   Delete          WaitForFirstConsumer   true                   17m
storageclass-wekafs-dir-static  csi.weka.io          Retain          Immediate              true                   4s
```

Likewise, create the 6 persistent volumes that map to the 6 directories on WEKA. These are the same pvc files were created before:

```
alex [ ~/cassandra/new ]$ for i in {0..5}; do kubectl apply -f pvc-cassandra-$i.yaml; done
persistentvolume/pv-wekafs-dir-static-cassandra-0 created
persistentvolume/pv-wekafs-dir-static-cassandra-1 created
persistentvolume/pv-wekafs-dir-static-cassandra-2 created
persistentvolume/pv-wekafs-dir-static-cassandra-3 created
persistentvolume/pv-wekafs-dir-static-cassandra-4 created
persistentvolume/pv-wekafs-dir-static-cassandra-5 created
```

The 6 persistent volumes are now ready to be claimed:

```
alex [ ~/cassandra/new ]$ kubectl get pv


NAME                                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS
REASON   pv-wekafs-dir-static-cassandra-0   5Ti        RWX            Retain           Available           storageclass-
wekafs-dir-static            pv-wekafs-dir-static-cassandra-1   5Ti        RWX            Retain           Available
storageclass-wekafs-dir-static           pv-wekafs-dir-static-cassandra-2   5Ti        RWX            Retain
Available           storageclass-wekafs-dir-static           pv-wekafs-dir-static-cassandra-3   5Ti        RWX
Retain           Available           storageclass-wekafs-dir-static           pv-wekafs-dir-static-cassandra-4   5Ti
RWX            Retain           Available           storageclass-wekafs-dir-static           pv-wekafs-dir-static-cassandra-
5   5Ti        RWX            Retain           Available           storageclass-wekafs-dir-static
```

And now create the Cassandra service:

```
alex [ ~/cassandra/new ]$ cat cassandra-service.yaml
#cassandra-service.yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: cassandra
  name: cassandra
spec:
  clusterIP: None
  ports:
  - port: 9042
  selector:
    app: cassandra
alex [ ~/cassandra/new ]$ kubectl apply -f cassandra-service.yaml
service/cassandra created
```

And finally, the Cassandra StatefulSet file will be created:

```
alex [ ~/cassandra/new ]$ cat cassandra-statefulset.weka.yaml
#cassandra-statefulset.yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: cassandra
  labels:
    app: cassandra
spec:
  serviceName: cassandra
  replicas: 6
  selector:
    matchLabels:
      app: cassandra
  template:
    metadata:
      labels:
        app: cassandra
    spec:
      terminationGracePeriodSeconds: 1800
      containers:
        - name: cassandra
          image: cassandra:4.0.7
          env:
          - name: CASSANDRA_SEEDS
            value: "cassandra-0.cassandra.default.svc.cluster.local"
          - name: POD_IP
            valueFrom:
              fieldRef:
                fieldPath: status.podIP
          lifecycle:
            preStop:
              exec:
               command:
               - /bin/sh
               - -c
               - nodetool drain
          readinessProbe:
            exec:
              command:
              - /bin/bash
              - -c
              - if [[ $(nodetool status | grep $POD_IP) == *"UN"* ]] ; then exit 0; else exit 1; fi
            initialDelaySeconds: 35
            timeoutSeconds: 5
          volumeMounts:
          - mountPath: "/var/lib/cassandra"
            name: pv-wekafs-dir-static
  volumeClaimTemplates:
  - metadata:
      name: pv-wekafs-dir-static
```

```
    apiVersion: v1
    kind: PersistentVolumeClaim
    spec:
      accessModes: [ "ReadWriteMany" ]
      storageClassName: storageclass-wekafs-dir-static
      volumeMode: Filesystem
      resources:
        requests:
          storage: 10Gi
alex [ ~/cassandra/new ]$ kubectl apply -f cassandra-statefulset.weka.yaml
statefulset.apps/cassandra created
```

As the StatefulSet is coming up, one of the persistent volumes has already been claimed:

```
alex [ ~/cassandra/new ]$ kubectl get pv


NAME                                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM
STORAGE pv-wekafs-dir-static-cassandra-0   5Ti        RWX            Retain           Bound       default/pv-wekafs-dir-
static-cassandra-0   storage pv-wekafs-dir-static-cassandra-1   5Ti        RWX            Retain           Available
storage pv-wekafs-dir-static-cassandra-2   5Ti        RWX            Retain           Available
storage pv-wekafs-dir-static-cassandra-3   5Ti        RWX            Retain           Available
storage pv-wekafs-dir-static-cassandra-4   5Ti        RWX            Retain           Available
storage pv-wekafs-dir-static-cassandra-5   5Ti        RWX            Retain           Available
storage
```

After a few minutes all the pods are up:

```
alex [ ~/cassandra/new ]$ kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
cassandra-0       1/1     Running   0          5m21s
cassandra-1       1/1     Running   0          4m17s
cassandra-2       1/1     Running   0          3m16s
cassandra-3       1/1     Running   0          2m29s
cassandra-4       1/1     Running   0          102s
cassandra-5       1/1     Running   0          53s
installer-81c85   1/1     Running   0          10m
installer-9nvws   1/1     Running   0          10m
installer-bx9cp   1/1     Running   0          10m
installer-dmvrp   1/1     Running   0          10m
installer-tqwhq   1/1     Running   0          10m
installer-tvb2z   1/1     Running   0          10m
```

All the persistent volumes are bound:

```
alex [ ~/cassandra/new ]$ kubectl get pv


NAME                                CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM
STORAGECLA pv-wekafs-dir-static-cassandra-0   5Ti        RWX            Retain           Bound    default/pv-wekafs-dir-
static-cassandra-0   storagecla pv-wekafs-dir-static-cassandra-1   5Ti        RWX            Retain           Bound
default/pv-wekafs-dir-static-cassandra-3   storagecla pv-wekafs-dir-static-cassandra-2   5Ti        RWX            Retain
Bound    default/pv-wekafs-dir-static-cassandra-4   storagecla pv-wekafs-dir-static-cassandra-3   5Ti        RWX
Retain           Bound    default/pv-wekafs-dir-static-cassandra-5   storagecla pv-wekafs-dir-static-cassandra-4   5Ti
RWX            Retain           Bound    default/pv-wekafs-dir-static-cassandra-2   storagecla pv-wekafs-dir-static-
cassandra-5   5Ti        RWX            Retain           Bound    default/pv-wekafs-dir-static-cassandra-1   storagecla
```

As noted, nodetool reports the Cassandra cluster is up:

```
alex [ ~/cassandra/new ]$ kubectl exec --stdin --tty cassandra-0 -- nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address     Load        Tokens  Owns (effective)  Host ID                               Rack
UN  10.0.0.46   306.66 GiB  16      49.0%             4882df7f-26ca-4692-b842-e712c4f6a3f2  rack1
UN  10.0.0.78   317.35 GiB  16      50.7%             ea512878-82de-4a96-b0e5-e34c7c2fb94a  rack1
UN  10.0.0.40   325.34 GiB  16      51.1%             603c92cd-72e6-4247-b8f4-5f1cdd1243dd  rack1
UN  10.0.0.162  311.34 GiB  16      48.8%             a775e1ae-2ff1-4427-bbc2-b77f24252c33  rack1
UN  10.0.0.151  312.63 GiB  16      49.9%             350fe554-a971-487f-8cc7-bec9bbbf0eb6  rack1
UN  10.0.0.119  321.2 GiB   16      50.5%             7a811d25-aec1-4ee4-ba3f-925d583e9c27  rack1
```

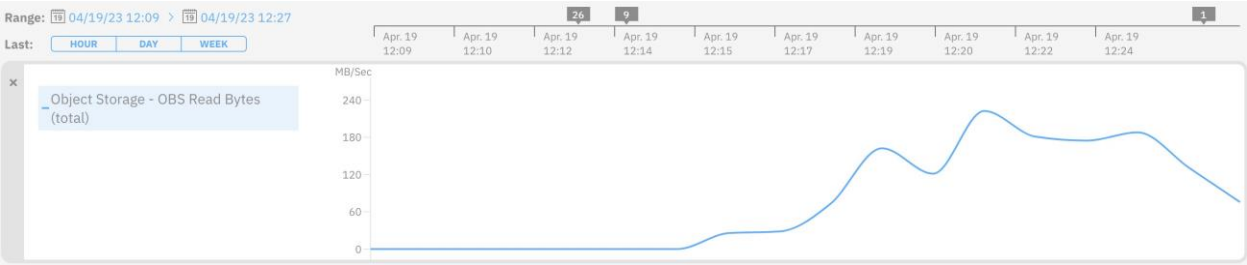
Notice as soon as the Cassandra cluster comes up some compaction jobs begin to run:

```
alex [ ~/cassandra/new ]$ for i in {0..5}; do echo cassandra-$i; done | xargs -I {} -P 0 kubectl exec --stdin --tty {} -- nodetool compactio

pending tasks: 0

pending tasks: 0

pending tasks: 1
- blog.posts: 1

pending tasks: 0

id                                   compaction type keyspace table completed   total       unit  progress
89fa18f0-ded6-11ed-9a44-899a365bf7f1 Compaction      blog     posts 18475806702 96133285673 bytes 19.22%
Active compaction remaining time :   0h19m17s
pending tasks: 1
- blog.posts: 1

id                                   compaction type keyspace table completed   total       unit  progress
7a804e30-ded6-11ed-b769-f906ea2d02ef Compaction      blog     posts 19416477588 33014294378 bytes 58.81%
Active compaction remaining time :   0h03m22s
pending tasks: 1
- blog.posts: 1

id                                   compaction type keyspace table completed   total       unit  progress
890b7330-ded6-11ed-b139-69eb943ed4d8 Compaction      blog     posts 18670059282 24588959554 bytes 75.93%
Active compaction remaining time :   0h01m28s
```

In just a few minutes of the cluster being up, it has already pulled around ~140GB of data from the object store back onto the SSD tier:

```
azureuser@alexh-client-1:~$ weka fs
FILESYSTEM ID  FILESYSTEM NAME  USED SSD   AVAILABLE SSD  USED TOTAL  AVAILABLE TOTAL  THIN PROVISIONED  THIN PROVISIONED MINIMUM SSD  THIN
1              default          144.07 GB  11 TB          3.23 TB     60 TB            False
```

The current download activity from the object store can be viewed on the WEKA console:



Now the query from earlier can be re-run (top 5 posts for user Douglas4214):

```
azureuser@alexh-client-1:~$ ./c_query.py


Douglas4214 39a1b9c6-dd70-11ed-9f46-000d3a99c2f6 Yourself they fortnightly this hail ugly shall us these to whom exaltation
play though pove


Douglas4214 2ed43086-dd70-11ed-b8bb-000d3a99c2f6 Their regularly murder wad you first to so those run that dream part
previously factory som


Douglas4214 07e61ca1-dd70-11ed-8e2c-000d3a99c2f6 House join besides entirely hand out destroy where which hers he under hers
snore tomorrow


Douglas4214 fffc899c-dd6f-11ed-8d82-000d3a99c2f6 What their anyone most very stand might crawl whose rarely which sometimes
who weekly her i


Douglas4214 f84e292c-dd6f-11ed-8382-000d3a99c2f6 Care must a dunk owing any recently these additionally anybody result infancy
catalog pree
```

If required, even much larger queries can be made:

```
azureuser@alexh-client-1:~/azureuser$ time ./simple_query.py | dd of=/dev/null
14642546+355 records in
14642720+1 records out
7497072828 bytes (7.5 GB, 7.0 GiB) copied, 131.297 s, 57.1 MB/s

real  2m11.300s
user  1m23.527s
sys 0m26.525s
```

Benefits

Many of the reasons Cassandra is selected for modern application development and deployment are based on its optimized architecture and distributed nature. When coupled with the WEKA Data Platform on Azure, customers can overcome the scale and performance challenges of Cassandra and take advantage of the high-performance data pipelines from the WEKA Filesystem as well as lower cloud costs and the complexities of management of the Cassandra cluster. Additionally, when an organization relies on applications built on cloud instances of Cassandra, disaster recovery and business continuity become critical to their success, and WEKA's Snap-to-Object feature provides the protection required and the recovery time objective necessary to meet or exceed customer expectations from a data protection perspective.

Additional benefits of containerized Cassandra on Azure with WEKA

- Easily manage and orchestrate using Azure Kubernetes Service (AKS) – WEKA is fully integrated with AKS which allows simple creation of the Kubernetes orchestrated Cassandra compute instances.
- All the Cassandra instance nodes can operate from a single WEKA filesystem that provides as much performance as the Cassandra nodes require. This significantly simplifies the design and operation of the Cassandra cluster as well as reduces cost due to eliminating the need to purchase instances with local instance capacity and/or purchase and maintain remote drives.
- Eliminate Cassandra rebuilds – upon a Cassandra instance node failure a new instance can be provisioned and mapped to the same filesystem/directory that the previous Cassandra instance used thereby removing the need for a Cassandra cross instance rebuild (since the new Cassandra instance already has the relevant data) this leaves only the short Cassandra Hints sync to complete. The implications are that the Cassandra cluster returns to a fully resilient state almost immediately without loading the compute, network, and storage of the Cassandra instances. This can be performed automatically by the WEKA CSI driver with the Kubernetes environment.
- On the fly Scale-out and scale-in the Cassandra cluster with no data movement required – Multiple containerized Cassandra nodes pods can run on a single Kubernetes worker instance, thereby allowing the creation of additional Kubernetes workers that will later contain some of the existing containerized Cassandra nodes pods migrated to them without requiring any Cassandra level data movement.
- Scale up and down Azure instance sizes on the fly without any data movement – in case a Kubernetes worker node instance is burdened with multiple Cassandra nodes pods workload, that worker node can be scaled up to a bigger instance type that can then increase performance to the existing pods. Since the data resides on the WEKA filesystem this is done instantly with no burden of Cassandra data movement. The same applies to

scaling down the worker node instances when less performance per worker node instance is required. This allows for significantly better cost and performance management.

- Fully elastic Cassandra storage capacity – By scaling out the WEKA cluster and/or by expanding it to an Azure BLOB environment it is easy to dynamically change the available capacity of the Cassandra Database without any need to add/remove Cassandra nodes and/or perform expensive data redistribution.
- Create test/dev database instances – by using the WEKA capability to create writable snapshots it is easy to snapshot the database filesystem and provision it to a different test/dev Cassandra database instance.
- Instant integrated Backups for the Cassandra environment – utilizing the WEKA instant snapshots at scale capability the WEKA system(s) that contain the Cassandra environment can be snapshotted at any required interval to provide immediate point-in-time backup of the Cassandra instances and the entire cluster. No need for external Cassandra backup software.
- Integrated Cassandra DR – With the WEKA capability of sending snapshots to remote locations and recovering the data to different provisioned WEKA systems, any Cassandra data can be sent to different Azure AZs and Regions to provide Cross AZ/Region backup and DR with immediate recovery allowing aggressive RPOs and RTOs. No need for external Cassandra backup/DR software.

About WEKA

WEKA offers a modern subscription software-based data platform delivering 10x+ performance and scale demanded by today's cloud and AI workloads. With the simplicity of NAS, the performance of SAN or DAS, and the scale of object storage, no more compromises between Simplicity, Speed, or Scale. Learn more at www.weka.io or follow us on Twitter @WekaIO.

WKA367–01 08/2023