SOLUTION GUIDE

Self-PoC on MemSQL Helios

Nithin Krishna Reghunathan, Technical Evangelist

February 2020

memSQL

Table of Contents

1. MemSQL Overview

MemSQL is an operational database built for performing both transactions and analytics to support the demands of modern applications, analytical systems, and ML/AI at scale. MemSQL uses a cloud-native, distributed architecture to deliver maximum performance and elastic scale. Note that "cloud-native" does not mean "cloud-only"; in fact, "cloud-native" infrastructure and apps are completely flexible, being able to run on any cloud or on-premises. MemSQL does so by offering both multi-cloud and hybrid options, ranging from a database-as-a-service, to Kubernetes-based hybrid and private deployments, to traditional on-premises installations on VMs or commodity hardware.

MemSQL can ingest millions of events per second, with support for ACID transactions, while simultaneously supporting analytics, applications, machine learning model queries, and AI queries on trillions[1] of data rows. MemSQL can support running transactional and analytical workloads under high concurrency, all while supporting the standard ANSI SQL query language. You can read this technical whitepaper for a deep dive into the concepts behind the MemSQL data platform.

If you're already familiar with running MemSQL on-premises, you can now enjoy the ultra-high performance and elastic scalability of MemSQL in the cloud.

2. Introducing Helios by MemSQL

MemSQL's database-as-a-service offering is called *MemSQL Helios*. MemSQL Helios gives you the full capabilities of MemSQL without the operational overhead and complexity of managing it yourself. MemSQL Helios provides a resilient database with cloud-agnostic deployment support on AWS and Google Cloud Platform (with support for Azure and others to come). With MemSQL Helios, cluster provisioning, cluster management, deployment, upgrades, alerting, and troubleshooting are all handled by MemSQL. This

[1] MemSQL Processing Shatters Trillion Rows Per Second Barrier
https://www.memsql.com/blog/memsql-processing-shatters-trillion-rows-per-second-barrier/

greatly reduces operational expenses, by shifting the database administration (DBA) tasks needed to operate your database from your organization to MemSQL. Just as importantly, MemSQL is offered at a price point dramatically lower than traditional database vendors, while our ultra-efficient query engine means that operational costs for MemSQL also tend to be lower than the proprietary offerings from the cloud service providers.

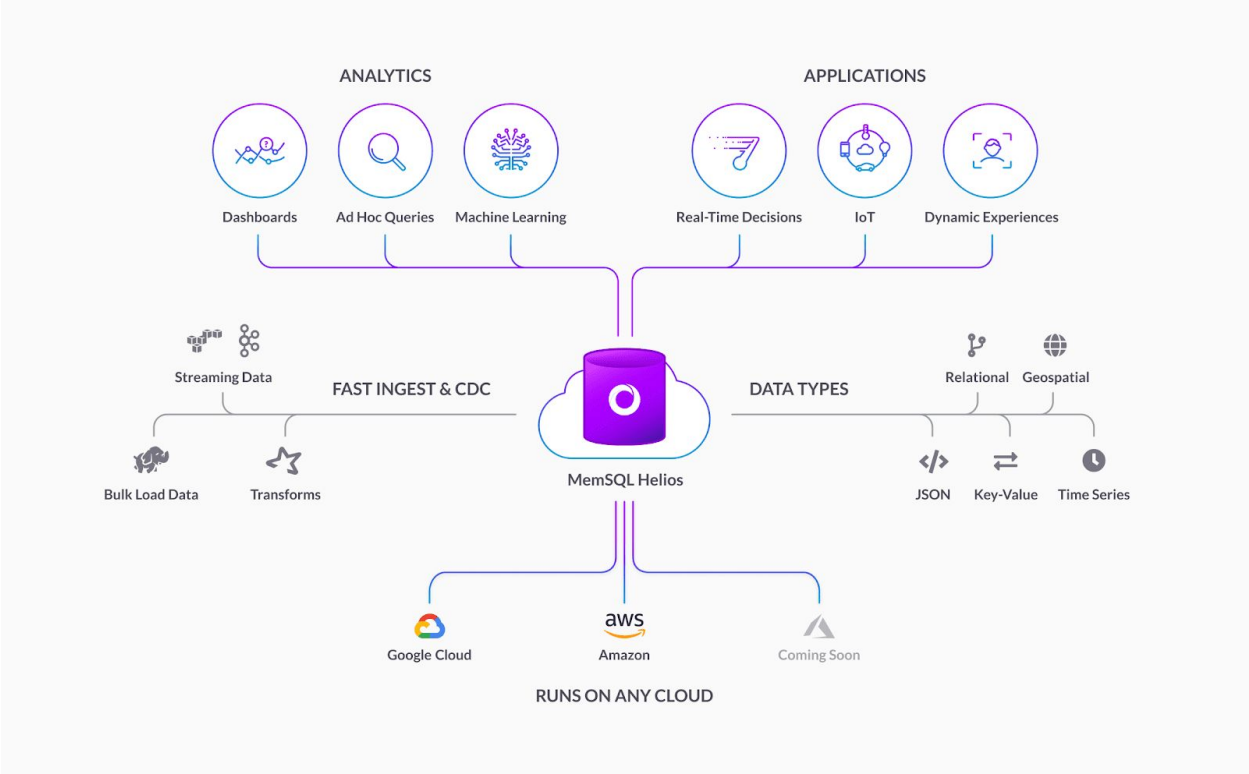

Figure 1. MemSQL Helios Managed Service

2.1 Benefits of MemSQL Helios

MemSQL Helios automatically backs up your data daily, with a retention period of seven days. MemSQL Helios runs in high availability mode, so you always have a live copy of your data, and MemSQL provides data restore services as needed. To meet regulatory compliance requirements, Helios supports client connections that are encrypted using transport layer security (TLS) and data-at-rest encryption.

Customers using MemSQL Helios are responsible for the logical management of their data, including schema design and implementation (DDL), index and query tuning,

assigning proper security permissions, requesting a database restore if needed, and requesting an increase or decrease in cluster capacity.

Depending on the needs of their application, customers have several options with MemSQL Helios. They can either opt to use dedicated, reserved resources if they need strong isolation guarantees, or they can choose to go with an on-demand model (running side-by-side with other tenants), where cluster usage is calculated hourly and billed monthly.

Key benefits of MemSQL Helios include:

- **Effortless deployment and management**

  As we have all come to expect from cloud services, deployment and upgrades are built in. With MemSQL Helios you get the full benefits and capabilities of the MemSQL data platform without having to worry about deployment, management, or maintenance. There's no need to rack servers, script deployments, or manage VMs.

- **Avoid cloud lock-in through multi-cloud flexibility**

  Helios is available today on Amazon Web Services and Google Cloud Platform, and availability on Microsoft Azure is next on the development roadmap. MemSQL operates exactly the same whether deployed on-premises on bare metal, on-premises on cloud infrastructure, using the MemSQL Kubernetes Operator, or within the Helios service. You can use MemSQL to support a broad set of operational and analytical use cases, allowing for a simple, single platform across applications, analytical systems, and cloud deployments.

- **Superior TCO**

  Compared to either legacy databases, or proprietary databases from cloud service providers, MemSQL Helios offers superior total cost of ownership (TCO). MemSQL offers high performance, scalability, ANSI SQL support, and the ability to replace

traditional databases, at a fraction of the cost. When compared to the proprietary databases offered on Amazon Web Services and Google Cloud Platform, MemSQL's unique architecture and high-performance query engine mean that many operational analytics workloads run with far less resource consumption, offering significant cost savings. And costs for MemSQL Helios are very similar to costs if you use the MemSQL database in the cloud directly, and manage it yourself - but with MemSQL Helios, the costs and hassles of managing the infrastructure that supports the MemSQL database are included in what you pay for the managed service offering.

3. Cluster Prerequisites

The following settings are recommended for spinning up a cluster on MemSQL Helios.

Items	Description	Available Settings
1.	Cluster Name	<set_self_PoC_test_name_db>
2.	Cluster Type	Development (by default)
3.	Choose a Region	AWS Virginia 1 Production, GCP Virginia1 Production, AWS Oregon 1 Production, or GCP Mumbai 1 Production
4.	Master User name	admin
5.	Set Cluster Password	<set_test_strong_password>
6.	Configure Cluster Access Restriction	**Specify IP Address Range Whitelist and/or click on 'Add my current IP address'

Table 1. Prerequisite Settings for MemSQL Helios

* Each unit has 8 vCPUs and 64 GB Memory (HA is enabled by default)

** Different ranges must be split into separate lines. We strongly recommend configuring your firewall to restrict which hosts can access MemSQL.

4. MemSQL Studio

MemSQL Studio is a visual user interface (UI) that allows you to easily monitor, debug and interact with all of your MemSQL clusters. Designed to be lightweight, easy to deploy, and easy to upgrade, MemSQL Studio provides the tools you need to maintain cluster health without the overhead of complex, heavyweight, and error-prone client software.

MemSQL Studio turns user actions into standard SQL queries that are run against your MemSQL cluster. Results are then displayed back to you in the form of tables and graphics that help you understand your cluster better. Conceptually, MemSQL Studio is a UI on top of the MemSQL database engine itself, pairing the stability and security guarantees of the command line with the ease of use of a visual UI.

When the cluster is up and running, open Studio and load any sample datasets to quickly start interacting with MemSQL.

5. Key Functionalities

Learn more about some of the key in-built functionalities in MemSQL that are required for better understanding of the use cases discussed in following chapters:

- **MemSQL Pipelines**

  MemSQL Pipelines is a MemSQL database feature that natively ingests real-time data from external sources. As a built-in component of the database, Pipelines can extract, transform, and load external data without the need for third-party tools or middleware. Pipelines are robust, scalable, and highly performant, and they support fully distributed workloads.

Pipelines support Apache Kafka, Amazon S3, Azure Blob, Filesystem, and HDFS data sources. In addition, they natively support the JSON, Avro, and CSV data formats.

- **Shard Key**

  The shard key is a collection of the columns in a table that are used to control how the rows of that table are distributed. To determine the partition responsible for a given row, MemSQL computes a hash from all the columns in the shard key to the partition ID. Therefore, rows with the same shard key will reside on the same partition.

- **Reference Tables**

  Reference tables are relatively small tables that do not need to have their data distributed, and that are present, in the form of a copy, on every node in the cluster. They are both created, and written to, on the master aggregator. Reference tables are updated via master-slave replication to every node in the cluster from the master aggregator. Replication enables reference tables to be dynamic: updates that you perform to a reference table on the master aggregator are quickly reflected on every machine in the cluster.

  MemSQL aggregators can take advantage of reference tables' ubiquity by pushing joins between reference tables and a distributed table onto the leaves. Imagine you have a distributed `clicks` table storing billions of records and a smaller `customers` table with just a few million records. Since the `customers` table is relatively small, it can be replicated onto every node in the cluster. If you run a join between the `clicks` table and the `customers` table, then the bulk of the work for the join will occur on the leaves, in parallel.

6. Operational Analytics: Build a Stock Ticker Database on MemSQL Helios

This chapter describes steps to build a sample *Stock Ticker* database on MemSQL Helios. This is a great use case to leverage the operational analytics capability of MemSQL.

6.1 Use Case - Stock Ticker

Stock market data can be used to analyze and run strong predictive models that may result in a large financial payoff. The amount of financial data on the web is seemingly infinite. The dataset* used here consists of the information about historical stock prices (last five years up to Feb 2018) for the companies found on the S&P 500 index.

The data file consists of the following columns:

- **Date** - In format: yy-mm-dd
- **Open** - Price of the stock at market open (this is NYSE data so all in USD)
- **High** - Highest price reached in the day
- **Low Close** - Lowest price reached in the day
- **Volume** - Number of shares traded
- **Name** - The stock's ticker name

Follow the steps below to test and build a sample stock ticker database on MemSQL Helios.

6.2 Reference Architecture

The diagram below shows the reference architecture overview for this use case.

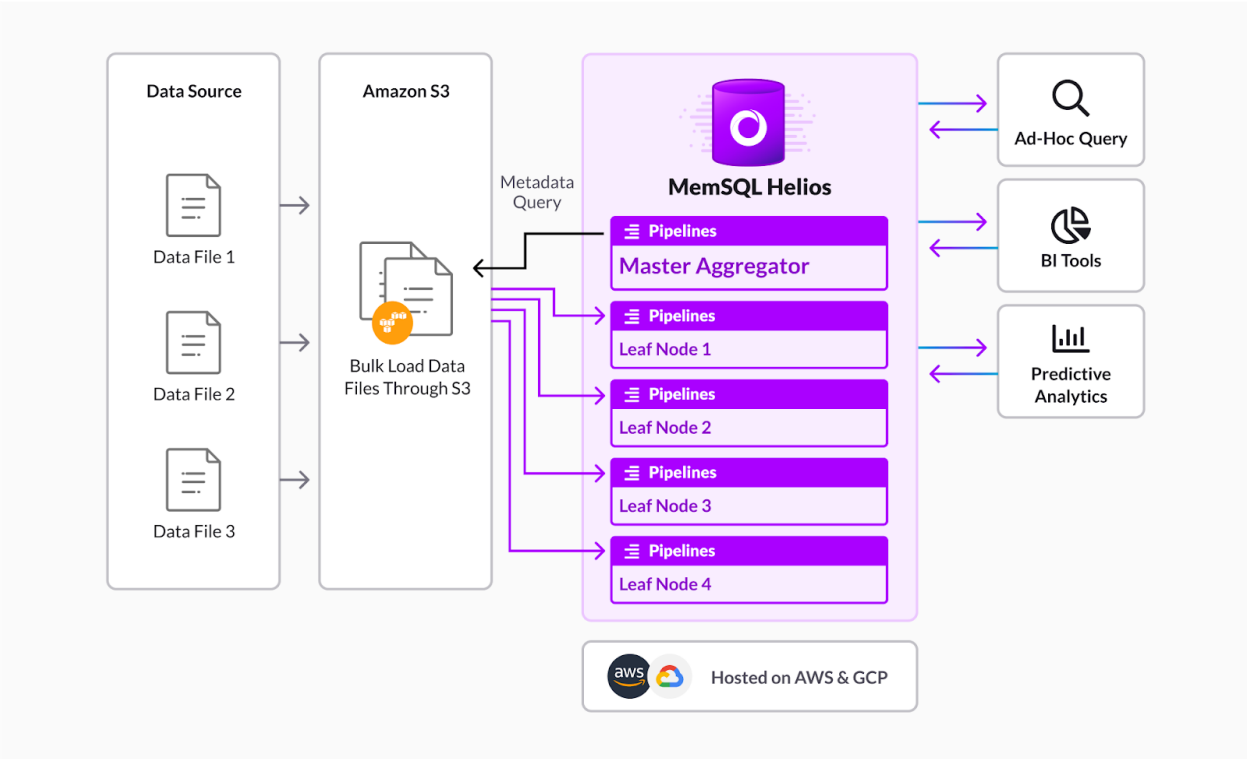

Figure 2. Stock ticker reference architecture

The raw data available from the *stock ticker* database application is initially loaded into Amazon S3 buckets. You can load the data into MemSQL Helios using the built-in pipeline functionality in the MemSQL engine. Large sets of data available from the source are bulk loaded into the database created in a MemSQL Helios cluster in a distributed fashion, using the shard key defined in Data Definition Language (DDL).

We are leveraging the memory optimized rowstore of MemSQL for this test case. The distributed, cloud-native architecture helps in managing the data efficiently by delivering ultra-fast performance and no-limits scalability to perform operational analytics, historical analysis, and predictive modelling using machine learning (ML) and artificial intelligence (AI). MemSQL Helios can be easily integrated with the leading BI tools (refer to Chapter 7) in order to visualize your results stored for various data analytics applications.

Follow the steps below to build and explore a sample database for the *Stock Ticker* application.

6.3 Spin Up a MemSQL Helios Cluster

The cluster deployment phase is a single-click approach upon providing the prerequisite platform settings. The cluster prerequisites defined in Chapter 3 show the recommended settings to spin up a free trial version of a Helios cluster.

The following link demonstrates the steps to spin up a self-managed MemSQL Helios cluster: Create MemSQL Helios Cluster.

6.4 Create Database and Table Definitions

You can connect with the Studio (visual UI) tool and take advantage of the built-in SQL editor to deploy the data definition language (DDL) commands.

- Create Database
    - Run this command to drop any existing database named 'stocks':

      ```
      DROP DATABASE IF EXISTS stocks;
      ```

    - Run this command to CREATE a new database named 'stocks':

      ```
      CREATE DATABASE stocks;
      ```

    - Run this command to switch to stocks database:

      ```
      USE stocks;
      ```

- Deploy the DDL to create table
    - The following command creates a table named *ticks*:

      ```
      CREATE TABLE ticks (
      dt DATETIME DEFAULT NULL,
      open FLOAT DEFAULT NULL,
      high FLOAT DEFAULT NULL,
      ```

```
low FLOAT DEFAULT NULL,
close FLOAT DEFAULT NULL,
volume int(11) DEFAULT NULL,
Name VARCHAR(8) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
 SHARD KEY tick (Name, dt)
);
```

6.5 Load Data into MemSQL Helios

Note. For your convenience, sample data sets have been already loaded into a Kafka cluster, which is managed by MemSQL. If you notice data feeds not functioning as described, or have any comments or suggestions, please contact MemSQL.

- Create a new S3 Pipeline

```
CREATE OR REPLACE PIPELINE ticks_pipeline
 AS LOAD DATA S3 'helios-self-poc-stockticker/'
 CONFIG '{"region":"us-east-1"}'
 SKIP DUPLICATE KEY ERRORS
 INTO TABLE ticks
 FIELDS TERMINATED BY ','
 LINES TERMINATED BY '\n';
```

- Start Pipeline

```
ALTER PIPELINE ticks_pipeline SET OFFSETS earliest;
START PIPELINE ticks_pipeline;
```

- Stop Pipeline

```
STOP PIPELINE ticks_pipeline;
```

- Check the status of the S3 upload

```
select * from information_schema.pipelines_files;
```

Note: You can also monitor the status of pipelines by following these steps: Click on Console in MemSQL Studio tool > Click on Pipelines, or run this command directly through the SQL editor in the console.

6.6 Run Sample Queries

- Query 1- How many events have we processed?

  ```
  Select count(*) from ticks;
  ```

- Query 2- What stock data are we analyzing?

  ```
  Select * from ticks limit 100;
  ```

- Query 3 - Count the number of events with stock ticker named 'AAL.'

  ```
  SELECT count(*) FROM ticks WHERE Name LIKE 'AAL';
  ```

- Query 4 - List the top 100 stock data points for the ticker named 'AAPL.'

  ```
  SELECT dt,high,low
  FROM ticks
  WHERE Name LIKE 'AAPL'
  GROUP BY high
  ORDER BY dt limit 100;
  ```

- Query 5 - Find the top 10 average stock data for the ticker named 'ABBV.'

  ```
  SELECT dt,AVG(HIGH),AVG(low),AVG(volume)
  FROM ticks
  WHERE Name LIKE 'ABBV'
  GROUP BY high
  ORDER BY dt LIMIT 10;
  ```

- Query 6 - Find the top 50 average stock data traded on *2013-02-08*.

  ```
  SELECT Name,AVG(HIGH),AVG(low),AVG(volume)
  FROM ticks
  WHERE dt like '%2013-02-08%'
  Limit 50;
  ```

Note: We encourage you to create and run more sample queries that align with your analytical workloads.

7. Streaming Analytics on MemSQL Helios

7.1 Use Case - Clickstream Advertising

The sample dataset used consists of clickstream advertising data. Marketer researchers can use clickstream data to infer vital details such as the user's demographic information, interests, browsing history and purchasing habits, building up a much more complete picture of their customer and their online activities. They can delve backwards in time to find the very first actions that started a user on their journey, or use predictive modelling to forecast their likely future actions.

7.2 Reference Architecture

The reference architecture diagram shown below gives you an overview of the workflow for this use case.

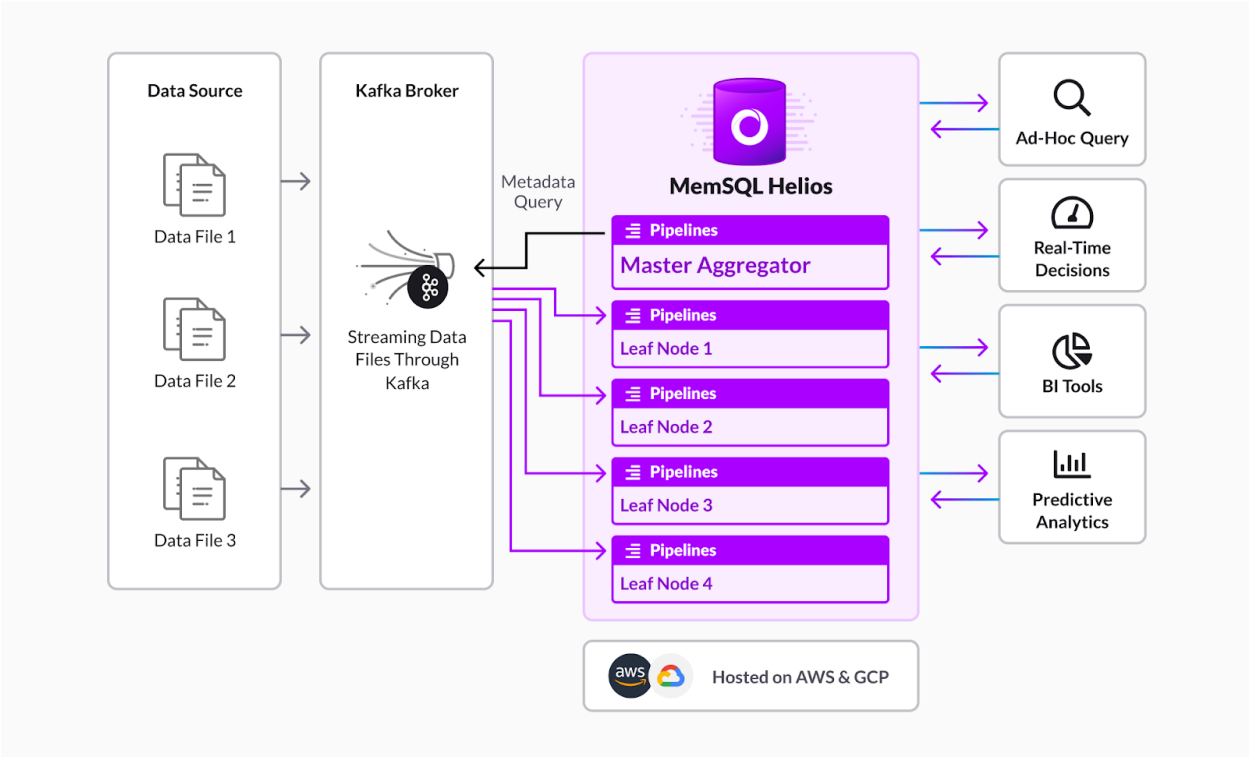

Figure 3. Clickstream reference architecture

The data files available from the *clickstream advertising* application are initially loaded into a Kafka topic to build a streaming application. You can leverage the Pipelines functionality (the built-in ETL engine of MemSQL) and stream data directly into the database created in MemSQL Helios. The lock-free ingest technology in MemSQL allows you to query while you ingest, seamlessly.

The data gets sharded based on the shard key defined in the DDL and stored in a distributed fashion. We are leveraging the compressed columnstore on disk to store the test data. You should be able to stream tens of millions of records in a second and simultaneously issue queries to perform ad hoc and real-time analytics. MemSQL Helios can be easily integrated with the leading BI tools (refer to Chapter 7) in order to visualize your data for various analytics applications.

Follow these steps to build and explore a sample database for the *clickstream advertising* application.

7.3 Spin Up a MemSQL Helios Cluster

Note: The prerequisite settings recommended in Chapter 2 have to be met before spinning up the cluster.

The following link demonstrates the steps to spin up a self managed MemSQL Helios cluster: Create MemSQL Helios Cluster.

7.4 Create Database and Table Definitions

You can connect with the Studio (visual UI) tool and leverage its SQL editor to create the database, then implement Data Definition Language (DDL) commands.

- Create database
    - Run this command to drop any existing database named *adtech*:
      `DROP DATABASE IF EXISTS adtech;`

- Run this command to CREATE a new database named *adtech*:

```
CREATE DATABASE adtech;
```

- Run this command to switch to the *adtech* database:

```
USE adtech;
```

- Create a table named *events*:

```
CREATE TABLE events (
    user_id int,
    event_name varchar(128),
    advertiser varchar(128),
    campaign int(11),
    gender varchar(128),
    income varchar(128),
    page_url varchar(128),
    region varchar(128),
    country varchar(128),
    KEY adtmidx (user_id,event_name,advertiser,campaign)
    USING CLUSTERED COLUMNSTORE, SHARD KEY user_id (user_id));
```

- Create a reference table named *campaigns*:

```
CREATE REFERENCE TABLE campaigns (
campaign_id smallint(6) NOT NULL DEFAULT '0',
campaign_name varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci
DEFAULT NULL, PRIMARY KEY (campaign_id));
```

- Insert sample data into the reference table:

```
INSERT INTO `campaigns` VALUES (1,'demand great'),(2,'blackout'),(3,'flame
broiled'),(4,'take it from a fish'),(5,'thank you'),(6,'designed by you'),(7,'virtual
launch'),(8,'ultra light'),(9,'warmth'),(10,'run healthy'),(11,'virtual city'),(12,'online
lifestyle'),(13,'dream burger'),(14,'super bowl tweet');
```

7.5 Load Data into MemSQL Helios

Note: For your convenience, sample data sets have been already loaded to a Kafka cluster, managed by MemSQL. If you notice data feeds not functioning as described, or have any comments or suggestions, please contact MemSQL.

Kafka topic: public-kafka.memcompute.com/ad_events

- **Create the Pipeline named *events_pipelines*:**

```
CREATE  or REPLACE PIPELINE events_pipeline
AS LOAD DATA KAFKA 'public-kafka.memcompute.com:9092/ad_events'
BATCH_INTERVAL 2500
INTO TABLE events
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
LINES TERMINATED BY '\n' STARTING BY ''
(user_id,event_name,advertiser,campaign,gender,income,page_url,region,country);
```

- **Start Pipeline**

```
/* Add ALTER for latest offsets */
ALTER PIPELINE events_pipeline SET OFFSETS earliest;
START PIPELINE events_pipeline;
```

7.6 Run Sample Queries while Streaming Data

- Query 1 - How many events have we processed?

```
SELECT COUNT(*) FROM events;
```

- Query 2 - How many users stands in a specific income range of '100K+.'

```
SELECT user_id,advertiser, event_name,gender,country
FROM events
WHERE income = "100k+"
group by campaign
```

```
ORDER BY advertiser desc;
```

- Query 3 - Find the traditional funnel campaigning information for the advertiser named *Walgreens*.

```
SELECT
Campaign,
Campaign_name,
 impression_count,
click_count,
downstream_conversion_count,
click_count / impression_count AS conv_1,
downstream_conversion_count / click_count AS conv_2,
downstream_conversion_count / impression_count AS all_conv
FROM (
SELECT campaign,
 SUM(CASE WHEN (event_name="Impression") THEN 1 ELSE null END) AS
impression_count,
SUM(CASE WHEN (event_name="Click") THEN 1 ELSE null END) AS click_count,
SUM(CASE WHEN (event_name="Downstream Conversion") THEN 1 ELSE null
END) AS downstream_conversion_count
FROM events
WHERE advertiser = "Walgreens"
group by campaign) tab
LEFT JOIN campaigns ON campaigns.campaign_id = campaign
ORDER BY all_conv desc;
```

- Query 4 - Find the conversion metrics information for the advertiser named *Walgreens*.

```
SELECT campaign, advertiser, country, SUM(CASE WHEN
(event_name="Impression") THEN 1 ELSE null END) AS impression_count,
```

```
SUM(CASE WHEN (event_name="Click") THEN 1 ELSE null END) AS click_count,
SUM(CASE WHEN (event_name="Downstream Conversion") THEN 1 ELSE null
END) AS downstream_conversion_count
FROM events
WHERE advertiser = "Walgreens";
```

- Query 5 - Targeted campaign information for advertiser named *McDonalds*.

```
SELECT user_id, page_url,region, country,
SUM(CASE WHEN (event_name="Click") THEN 1 ELSE null END) AS click_count,
 SUM(CASE WHEN (event_name="Downstream Conversion") THEN 1 ELSE null
END) AS downstream_conversion_count
FROM events
WHERE advertiser = "McDonalds"
group by campaign
ORDER BY user_id desc;
```

Note: We encourage you to try out more sample queries that align with your regular workloads.

8. Real-time Geospatial Analysis on Ride-sharing Data

8.1 Use Case: Real-time Insights on Ride-sharing Data

This use case simulates taxi or ride-sharing data collected from New York City, that can be ingested and analyzed in the MemSQL engine. Three pipelines are used to stream data from their corresponding topics in a public Kafka cluster. You will be running analytics while ingesting highly concurrent streaming data. The queries used are capable of running

cross-joins across multiple tables, executing geospatial functions etc. against streaming data, and helping end users make decisions based on real-time insights.

8.2 Reference Architecture

The reference architecture diagram shown below gives you an overview about the workflow for this use case.

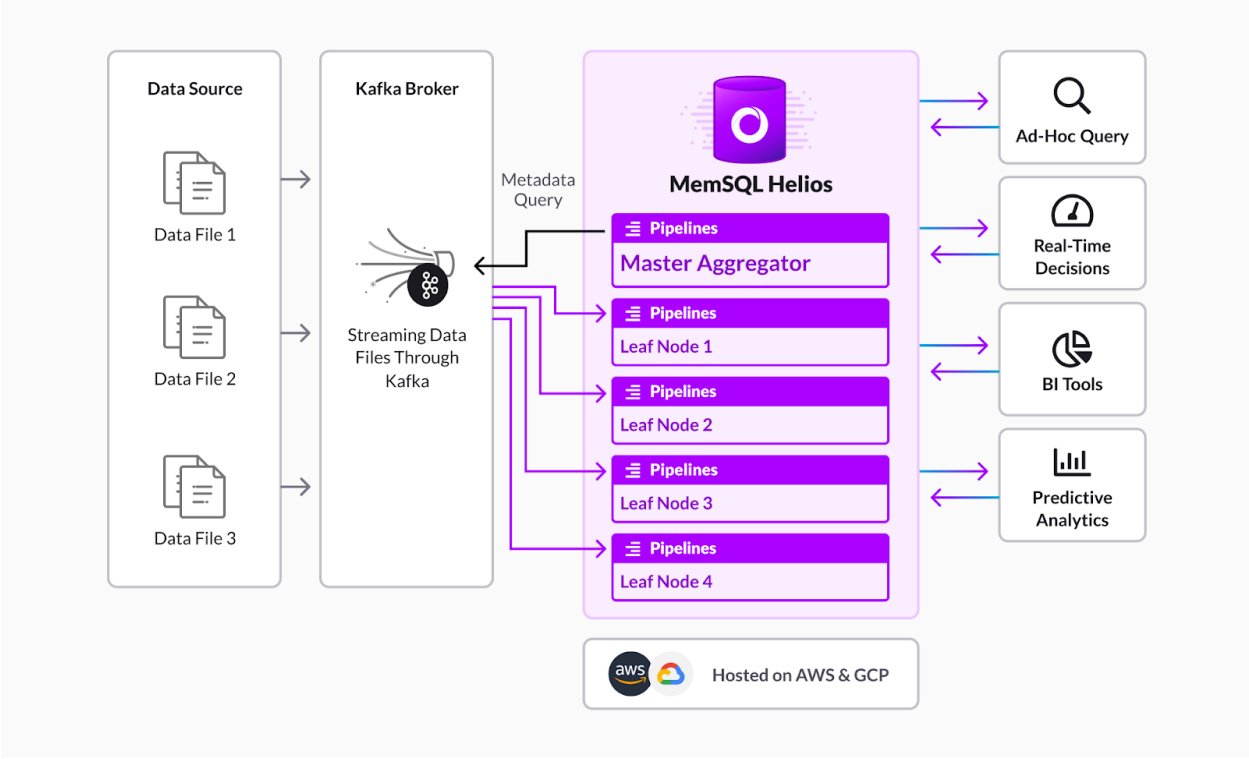

Figure 4. Reference architecture overview

The data files are streamed into multiple Kafka topics to build a streaming application. The Pipelines functionality in MemSQL can be leveraged to perform real-time streaming of data into MemSQL Helios database. We take advantage of the optimized rowstore technology in MemSQL (sharded by a predefined primary key) to uniformly distribute the data among the leaf nodes. The end users should be able to receive up-to-date information for their queries, as this unique architecture can seamlessly deliver real-time insights from the rapidly changing data, streamed through the Kafka cluster. MemSQL

Helios can be easily integrated with the leading BI tools (refer to Chapter 7) in order to visualize your data for various analytics applications.

8.3 Spin Up a MemSQL Helios Cluster

Note: The prerequisite settings recommended in Chapter 2 have to be met before spinning up the cluster.

The following link demonstrates the steps to spin up a self managed MemSQL Helios cluster: Create MemSQL Helios Cluster.

8.4 Create Database and Table Definitions

You can connect with the Studio (visual UI) tool and leverage its SQL editor to create the database followed by the implementation of Data Definition Language (DDL) commands.

Before creating a new pipeline, a database and table with the appropriate schema must exist. The following SQL commands will create a new database named *nyc_taxi*, and then create 3 tables in the database: drivers, neighborhoods, and trips.

- Create Database
    - Drop any existing database named *nyc_taxi*:

      ```
      DROP DATABASE IF EXISTS nyc_taxi;
      ```
    - Create a new database named *nyc_taxi*:

      ```
      CREATE DATABASE nyc_taxi;
      ```
    - Switch to the *nyc_taxi* database:

      ```
      USE nyc_taxi;
      ```
- Create a table named *drivers*:

  ```
  CREATE TABLE drivers (
    id bigint(20) NOT NULL,
    first_name varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci
  DEFAULT NULL,
  ```

```
last_name varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
location geographypoint NOT NULL,
goal_location geographypoint DEFAULT NULL,
status varchar(20) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
trip_id bigint(20) DEFAULT -1,
PRIMARY KEY (id),
KEY location (location));
```

- Create a table named *neighborhoods*:

```
CREATE TABLE neighborhoods (
id bigint(20) NOT NULL,
name varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
borough varchar(128) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
polygon geography DEFAULT NULL,
PRIMARY KEY (id));
```

- Create a table named *trips*:

```
CREATE TABLE trips (  id bigint(20) NOT NULL,
status varchar(16) CHARACTER SET utf8 COLLATE utf8_general_ci DEFAULT NULL,
pickup_location geographypoint NOT NULL DEFAULT 'Point(0 0)',
dropoff_location geographypoint NOT NULL DEFAULT 'Point(0 0)',
request_time int(11) DEFAULT NULL,
request_attempts int(11) NOT NULL DEFAULT 1,
accept_time int(11) DEFAULT NULL,
pickup_time int(11) DEFAULT NULL,
dropoff_time int(11) DEFAULT NULL,
num_riders int(11) DEFAULT NULL,
```

```
 price int(11) DEFAULT NULL,
 driver_id bigint(20) NOT NULL DEFAULT 0,
 PRIMARY KEY (id));
```

8.5 Load Data into MemSQL Helios

Note: For your convenience, sample data sets have been already loaded to a Kafka cluster, managed by MemSQL. If you notice data feeds not functioning as described, or have any comments or suggestions, please contact MemSQL.

For the sample stream, we will be publishing the status of 1000 drivers every second, and publishing 50 trips every thirty seconds. The "neighborhoods" stream is a static data set, and will not be updated after the initial load.

The following topics in a public Kafka cluster are used as pipeline data sources. Since a Kafka pipeline is paired with a single Kafka topic, in the following sections you will create one pipeline for each of these topics:

```
public-kafka.memcompute.com/drivers
public-kafka.memcompute.com/trips
public-kafka.memcompute.com/neighborhoods
```

- Create a Pipeline named *drivers*:

```
CREATE or REPLACE PIPELINE drivers
AS LOAD DATA KAFKA 'public-kafka.memcompute.com:9092/drivers'
BATCH_INTERVAL 2500
INTO TABLE drivers
FIELDS TERMINATED BY ',' ENCLOSED BY '' ESCAPED BY '\\'
LINES TERMINATED BY '\n' STARTING BY '';
```

- Create a Pipeline named *trips*:

```
CREATE or REPLACE  PIPELINE trips
AS LOAD DATA KAFKA 'public-kafka.memcompute.com:9092/trips'
```

```
BATCH_INTERVAL 2500
INTO TABLE trips
FIELDS TERMINATED BY ',' ENCLOSED BY '' ESCAPED BY '\\'
LINES TERMINATED BY '\n' STARTING BY '';
```

- Create a Pipeline named *neighborhoods*:

```
CREATE or REPLACE PIPELINE neighborhoods
AS LOAD DATA S3 'memsql-datafeeds/nyc-taxi-data/'
CONFIG '{"region":"us-east-1"}'
SKIP DUPLICATE KEY ERRORS
INTO TABLE neighborhoods
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
LINES TERMINATED BY '\n' STARTING BY '';
```

- Start Pipeline:

```
ALTER PIPELINE drivers SET OFFSETS EARLIEST;
ALTER PIPELINE trips SET OFFSETS EARLIEST;
START PIPELINE drivers;
START PIPELINE trips;
START PIPELINE neighborhoods;
```

- Commands to stop the Pipelines:

```
STOP PIPELINE drivers;
STOP  PIPELINE trips;
STOP  PIPELINE neighborhoods;
STOP ALL PIPELINES;
```

- Run this command to check the status of the upload into S3:

```
select * from information_schema.pipelines_files;
```

8.6 Run Sample Queries while Streaming Real-time Data

Query 1 - Total number of trips for each neighborhood.

Note: This query joins across multiple tables and executes geospatial functions while ingesting streaming data.

```
SELECT COUNT(*) num_rides, n.name
FROM trips t, neighborhoods n
WHERE
  n.id IN (
 SELECT id FROM neighborhoods
   ) AND
   GEOGRAPHY_INTERSECTS(t.pickup_location, n.polygon)
GROUP BY n.name
ORDER BY num_rides DESC;
```

Query 2 - The average amount of time between someone requesting a ride and that person being picked up.

```
SELECT ROUND(AVG(pickup_time - request_time) / 60,2) val
FROM trips t, neighborhoods n
WHERE
 n.id IN (
SELECT id FROM neighborhoods) AND
GEOGRAPHY_INTERSECTS(t.pickup_location, n.polygon) AND
pickup_time != 0 AND
request_time != 0;
```

Query 3 - The average distance of a trip.

```
SELECT ROUND(AVG(geography_distance(pickup_location, dropoff_location) /
1000), 2) val
FROM trips t, neighborhoods n
WHERE
```

```
n.id IN (
SELECT id FROM neighborhoods) AND
GEOGRAPHY_INTERSECTS(t.pickup_location, n.polygon);
```

Query 4 - The average amount of time between someone being picked up and that person being dropped off.

```
SELECT ROUND(AVG(dropoff_time - pickup_time) / 60, 2) val
FROM trips t, neighborhoods n
WHERE
status = "completed" AND
 n.id IN (
SELECT id FROM neighborhoods) AND
GEOGRAPHY_INTERSECTS(t.pickup_location, n.polygon);
```

Query 5 - The average cost of a trip.

```
SELECT ROUND(AVG(price), 2) val
FROM trips t, neighborhoods n
WHERE
status = "completed" AND
n.id IN (
SELECT id FROM neighborhoods
) AND
GEOGRAPHY_INTERSECTS(t.pickup_location, n.polygon);
```

Query 6 - The average amount of time it takes from the time a driver accepts a ride to the time they pick up the passenger.

```
SELECT ROUND(AVG(pickup_time - accept_time) / 60, 2) val
FROM trips t, neighborhoods n
WHERE
  pickup_time != 0 AND
```

```
  accept_time != 0 AND
  n.id IN (
    SELECT id FROM neighborhoods
  ) AND
  GEOGRAPHY_INTERSECTS(t.pickup_location, n.polygon);
```

Query 7 - The average number of riders per trip.

```
SELECT ROUND(AVG(num_riders), 2) val
FROM trips t, neighborhoods n
WHERE
 status = "completed" AND
 n.id IN (
SELECT id FROM neighborhoods) AND
GEOGRAPHY_INTERSECTS(t.pickup_location, n.polygon);
```

9. Visualize Data with Business Intelligence (BI) Tools

A sample dashboard built for a *ClickStream Advertising* application by integrating MemSQL Helios with Looker will look like the diagram below (Figure 4).

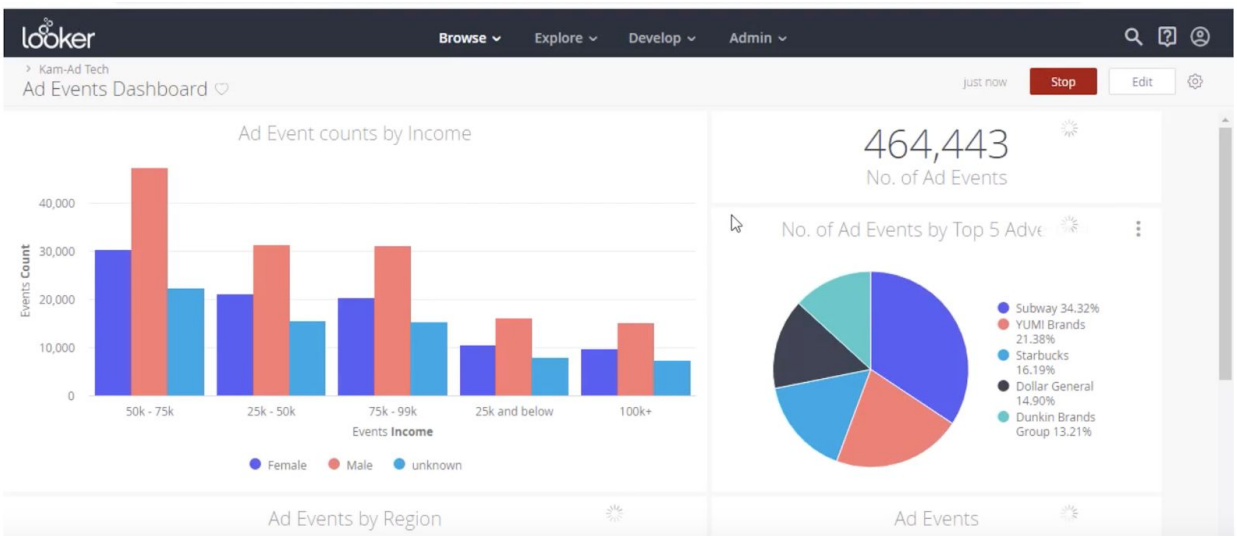

Figure 4. Sample dashboard for clickstream advertising

MemSQL partners with the leading third-party BI tool providers to perform data visualization and real-time analytics using the data stored in your MemSQL Helios cluster. Most of the leading BI tools have been successfully integrated, tested, and certified by MemSQL through our partnership program.

The following how-to guides show you the step-by-step approach involved in integrating MemSQL Helios with BI tools. Listed below are our trusted partners:

- Tableau
- Looker
- Data Virtuality Pipes
- Informatica Cloud

- Power BI
- Talend with Open Studio
- Dremio
- Tibco Spotfire
- MicroStrategy
- Streamsets
- Spark
- Sisense
- Collibra
- Zoomdata

10. Conclusion

As the amount of data across modern data-driven organizations grows, the challenges involved in managing the data become more complicated. You need a highly performing, self-managed database in the cloud, that is robust and scalable enough to meet the demanding requirements of your operational and analytical workloads.

MemSQL Helios, a fully-managed service offering of MemSQL, stands apart, with its unique excellence in distributed cloud-native architecture, by delivering ultra-fast performance and scalability for cloud workloads. The different use cases described in this paper demonstrate: (i) how easily you can spin up a cluster, (ii) bulk load or real-time streaming of sample data sets, (iii) running ad-hoc analytics on real-time data, and (iv) easy integration with BI tools to visualize the data stored in a MemSQL Helios cluster.

With the MemSQL Helios database, you don't have to worry about complex database infrastructure management tasks such as server provisioning, cluster setup, patching, backup or recovery, as they are all fully automated. This allows developers to stay focused on the application development lifecycle.

Many enterprises are leveraging MemSQL Helios for building smarter applications that can deliver data-driven decision-making capability, which the company can run with dramatic cost savings. Being able to ingest millions of events per second from S3, Kafka, Hadoop, Spark, Azure blob and more, with query response time in milliseconds, MemSQL Helios is an ideal solution for use cases such as real-time and near-real-time reporting, historical analysis, and operational analytics. MemSQL Helios can be considered as a great database solution for solving some of the complex big data problems in our data-driven world.

With MemSQL Helios now publicly available, we hope that you can experience it for yourself, and share your success story.

Test drive MemSQL Helios at memsql.com/helios.