memSQL

# The Database of Now ™

Dusty Everson - Account Executive

Mike Czabator - Director of Technical Sales

Mark Lochbihler - Global Director, Partner Engineering

Sarung Tripathi - Solution Engineering

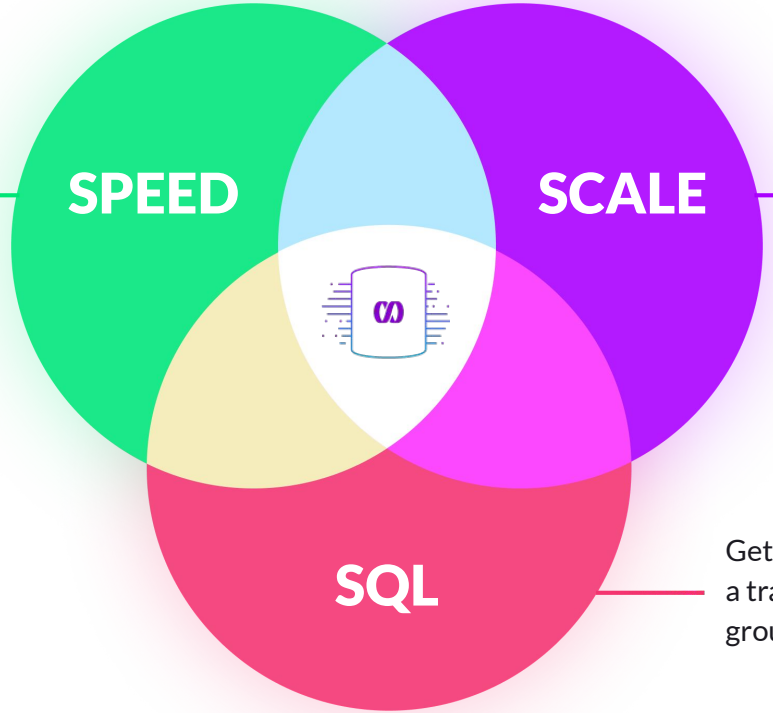
Roxanna Pourzand - Product Manager

# MemSQL Introduction

# MemSQL: The Database of Now

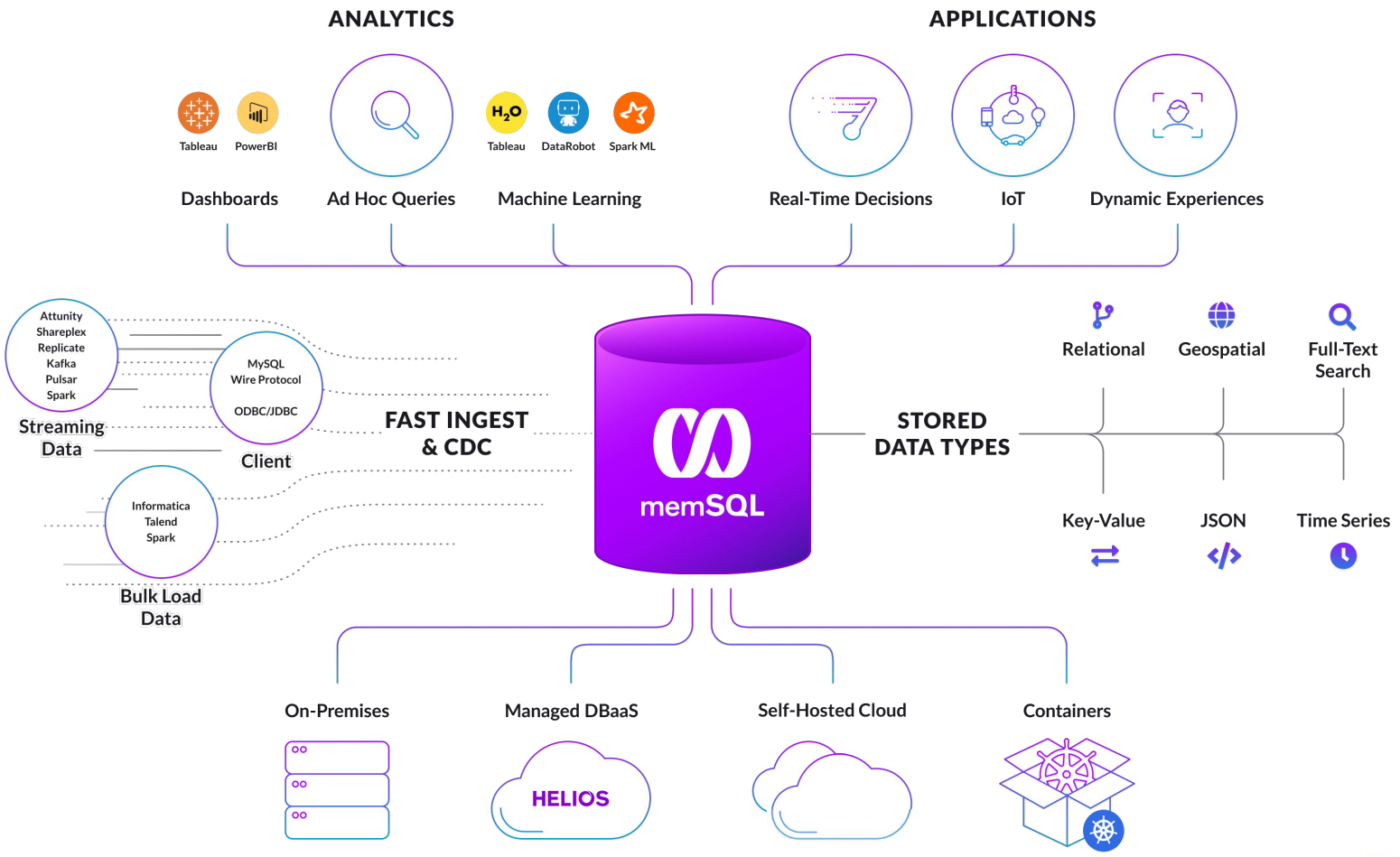
The cloud-native operational database built for speed and scale

Accelerate time to insight with ultra fast ingest and high performance query




Build on a cloud-native data platform designed for today's most demanding applications and analytical systems

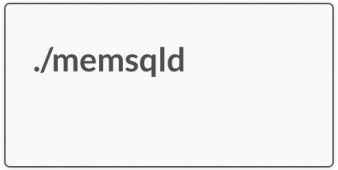
Get the familiarity & ease of integration of a traditional RDBMS and SQL, but with a groundbreaking, modern architecture


ANALYTICS
APPLICATIONS
Tableau
PowerBI
Tableau
DataRobot
Spark ML
Dashboards
Ad Hoc Queries
Machine Learning
Real-Time Decisions
IoT
Dynamic Experiences
Attunity
Shareplex
Replicate
Kafka
Pulsar
Spark
Streaming Data
MySQL Wire Protocol
ODBC/JDBC
Client
Informatica
Talend
Spark
Bulk Load Data
FAST INGEST & CDC
memSQL
STORED DATA TYPES
Relational
Geospatial
Full-Text Search
Key-Value
JSON
Time Series
On-Premises
Managed DBaaS
Self-Hosted Cloud
Containers
HELIOS
RUNS EVERYWHERE

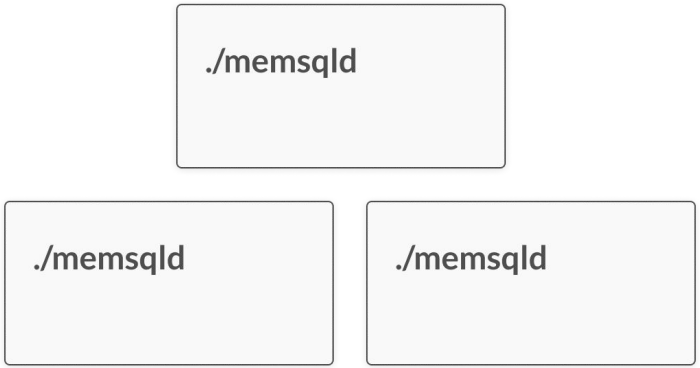
# MemSQL - Data Landscape

# Architecture

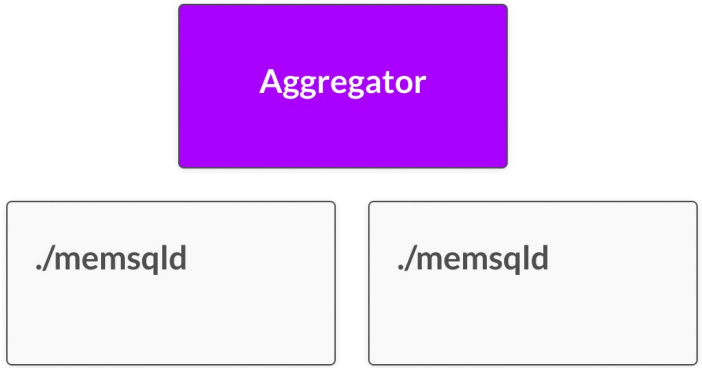
# MemSQL is a Database, a Linux Daemon

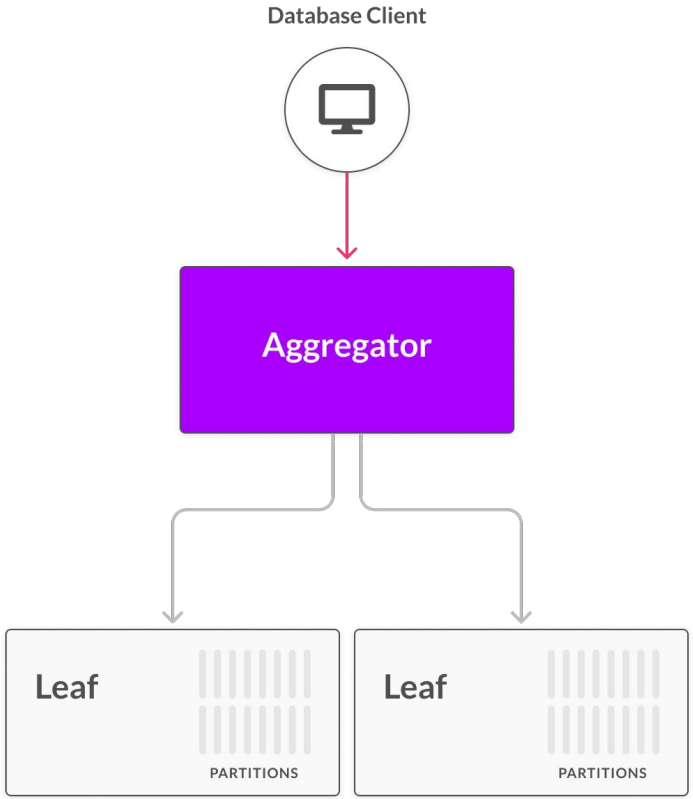


# MemSQL is a Distributed System



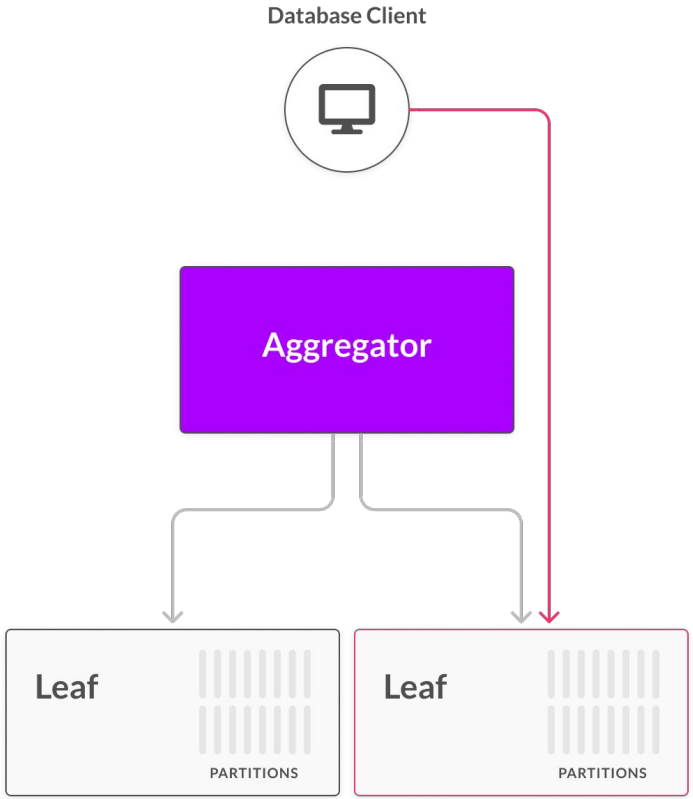
# Aggregators Aggregate





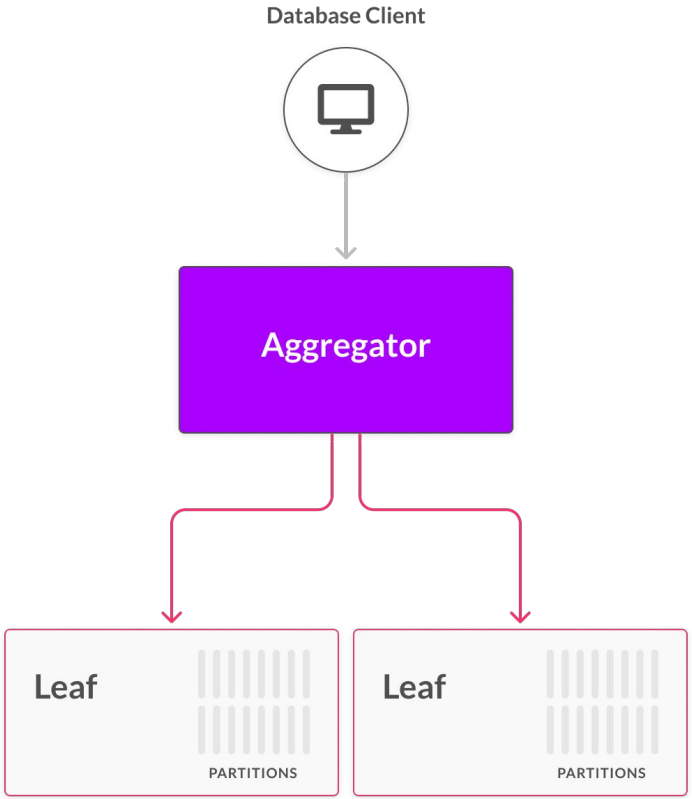

Aggregators interact with clients and leverage leaf nodes

```
aggregator-1> create database foo;
Query OK, 1 row affected (5.48 sec)
```




Leaves store a partition per core on the machine (by default)

```
leaf-2> show databases;
+--------------------+
| Database           |
+--------------------+
| cluster            |
| foo                |
| foo_1              |
| foo_3              |
| foo_5              |
| foo_7              |
| foo_9              |
| foo_11             |
| information_schema |
| memsql             |
+--------------------+
10 rows in set (0.01 sec)
```

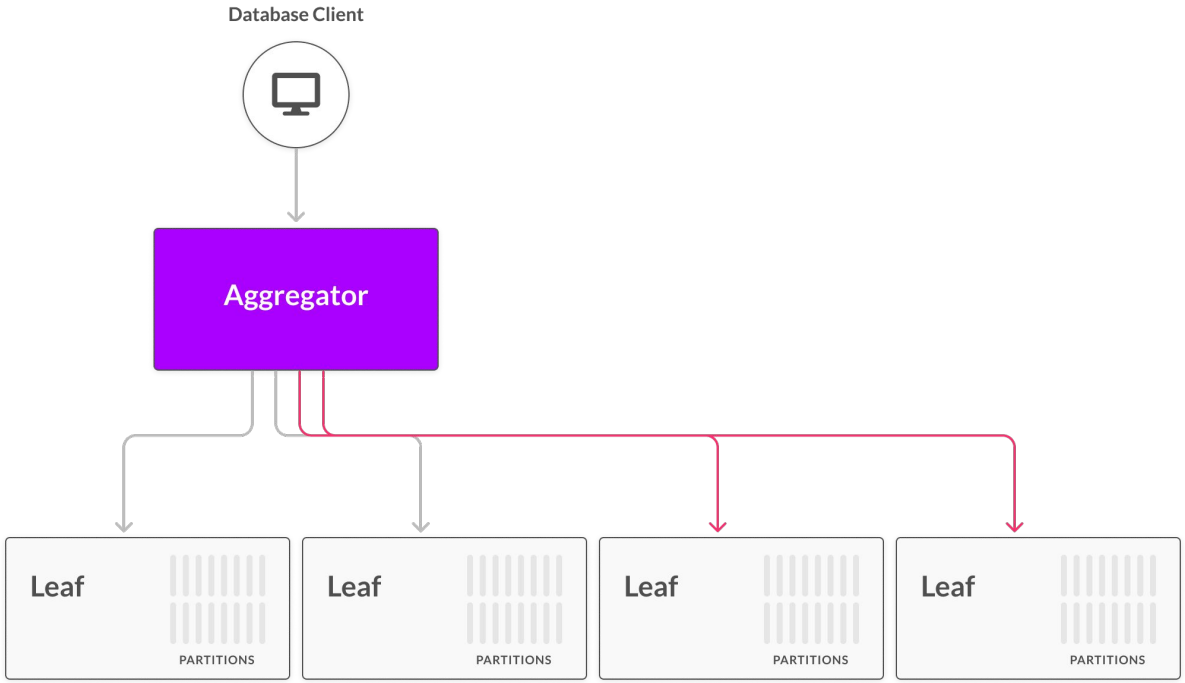



Massively parallel processing (MPP) across all the leaf nodes for query execution

```
aggregator-1> SELECT avg(price) FROM
orders;
...
```

```
leaf-1> using memsql_demo_9 SELECT
count(1), sum(price) FROM orders;
...
```

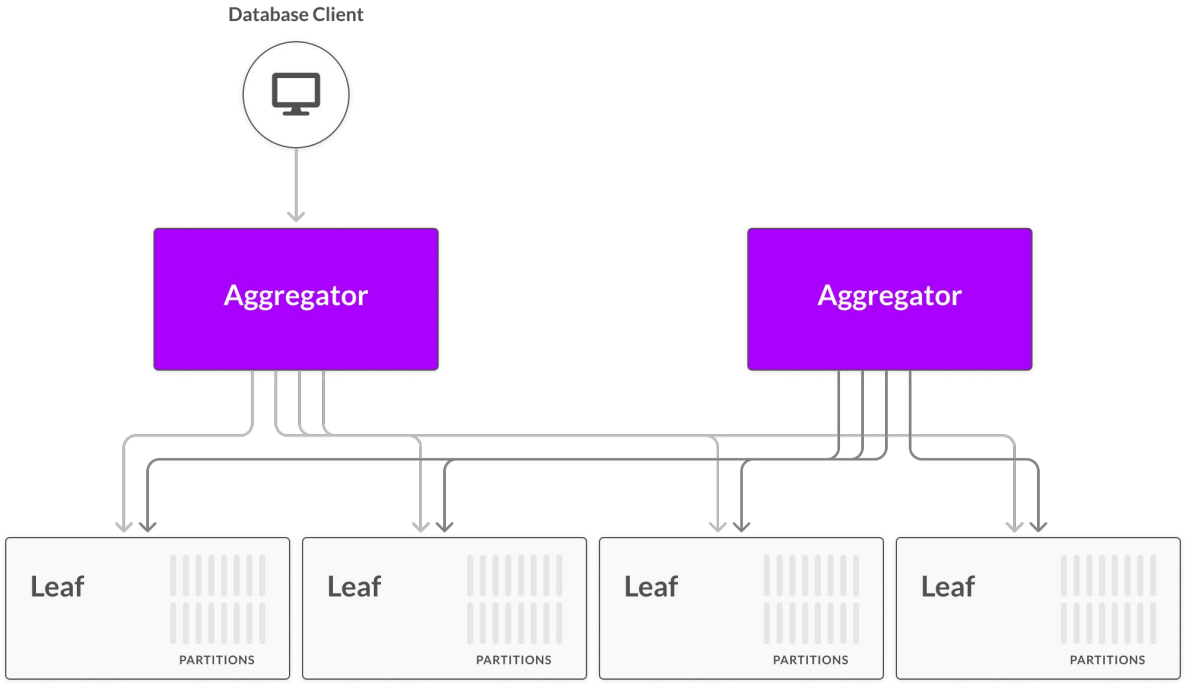
```
leaf-2> using memsql_demo_17 SELECT
count(1), sum(price) FROM orders;
...
```




Scale up and down on the fly

```
1  aggregator-1> ADD LEAF leaf-3…
2  aggregator-1> REBALANCE PARTITIONS;
3
4
```

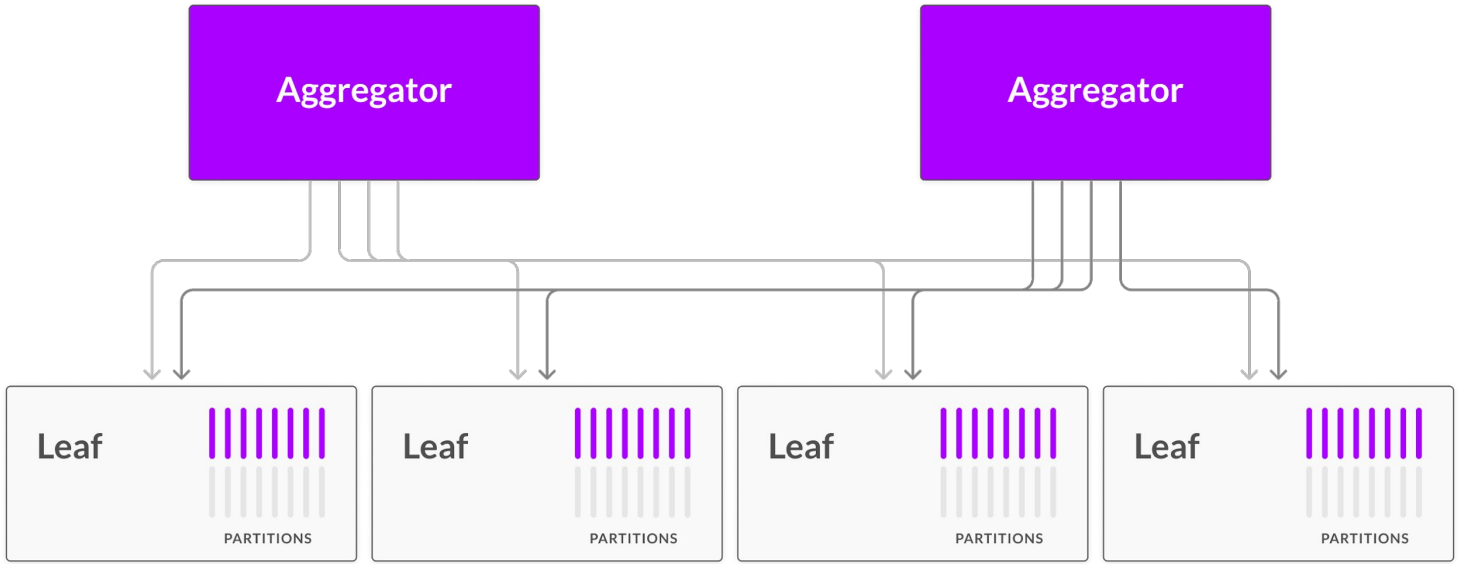
```
1  aggregator-1> ADD LEAF leaf-4…
2  aggregator-1> REBALANCE PARTITIONS;
3
4
```




Scale up and down on the fly

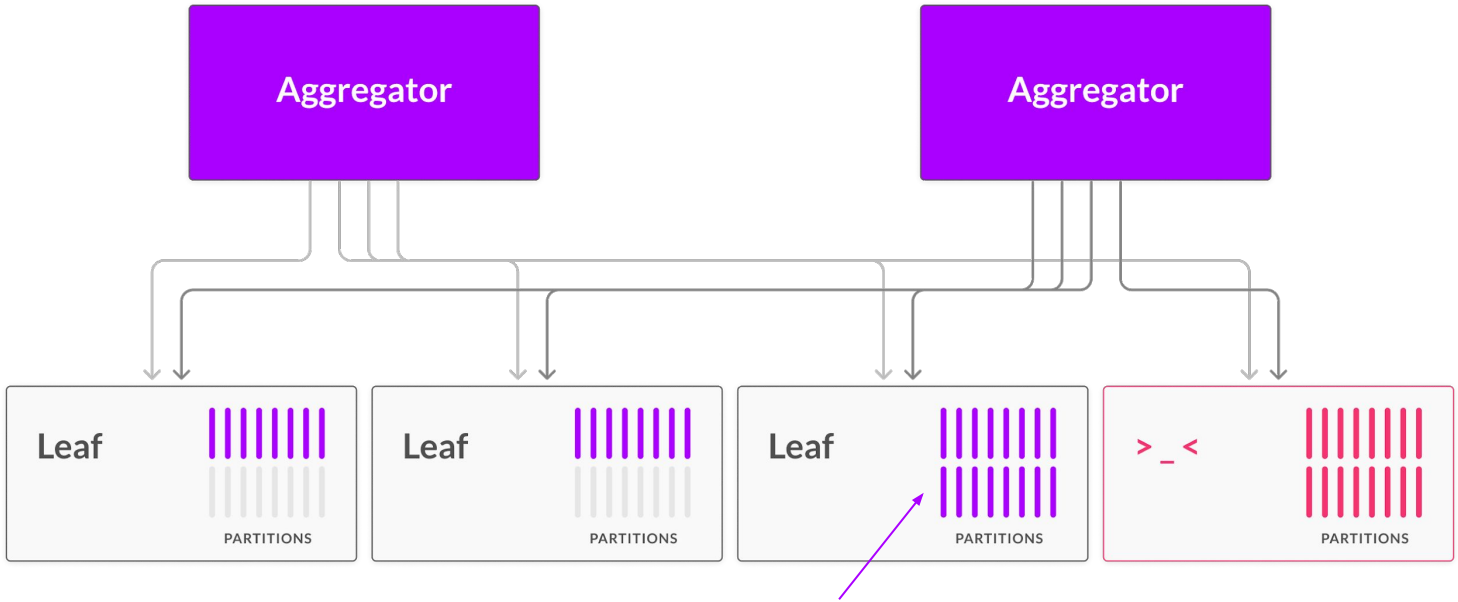
```
1  [memsql.cnf]
2  master-agg=agg-1
3
4
```

# Data is Copied for High Availability



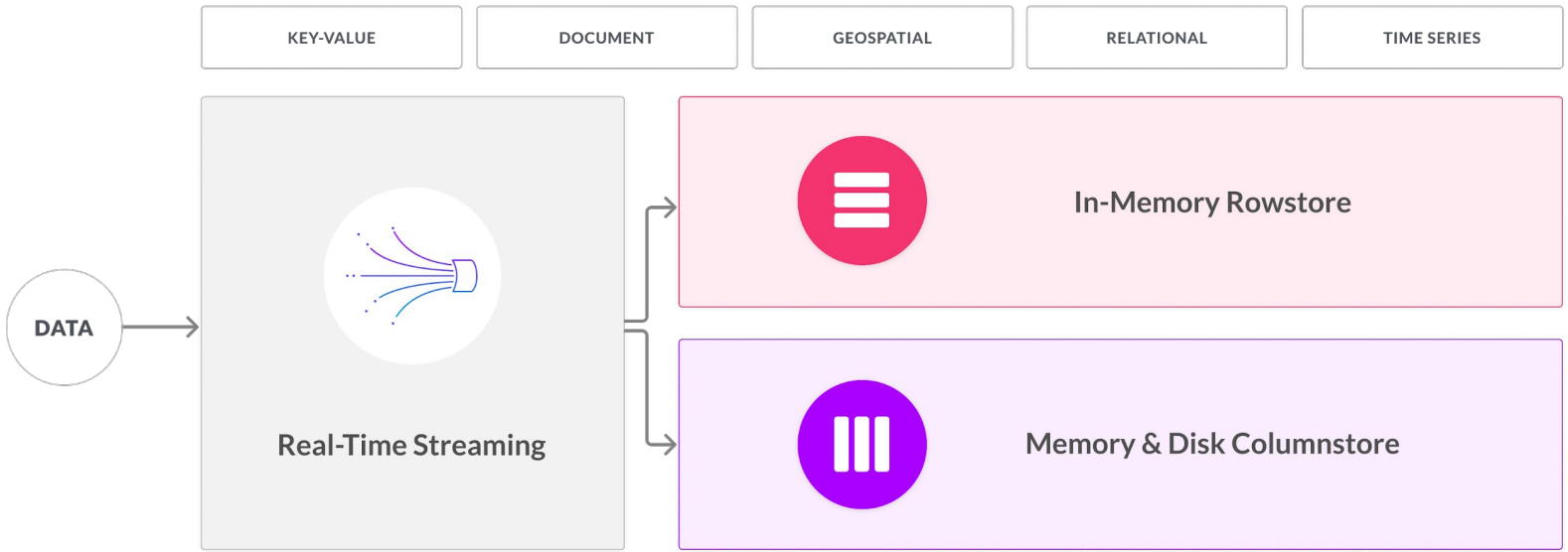

Each leaf has a pair which replicates its data

# Stays Online in Case of Node Failure




Pair's replicas are promoted to master partitions

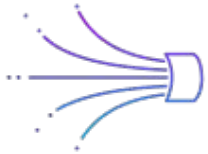
# Two Storage Engines



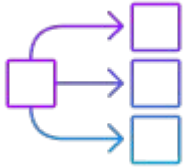
# MemSQL capabilities

# What Makes MemSQL Unique?

Distributed Cloud Native Architecture



Streaming Ingest

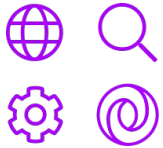


Lock-free

Query Compilation


ANSI SQL


Compatibility

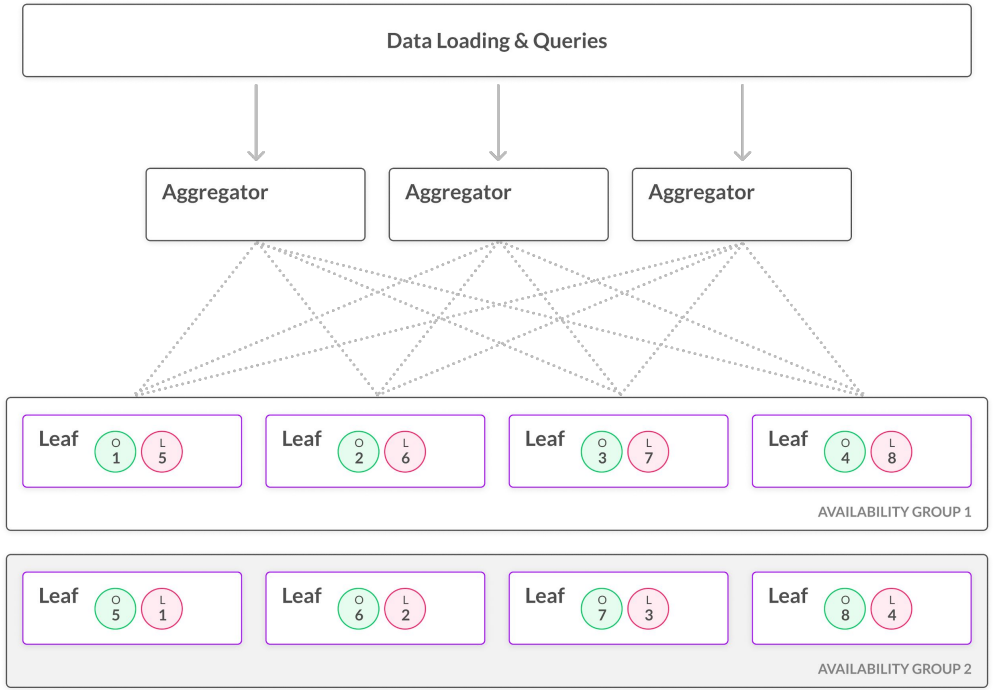


Data Flexibility

# MemSQL Data Architecture

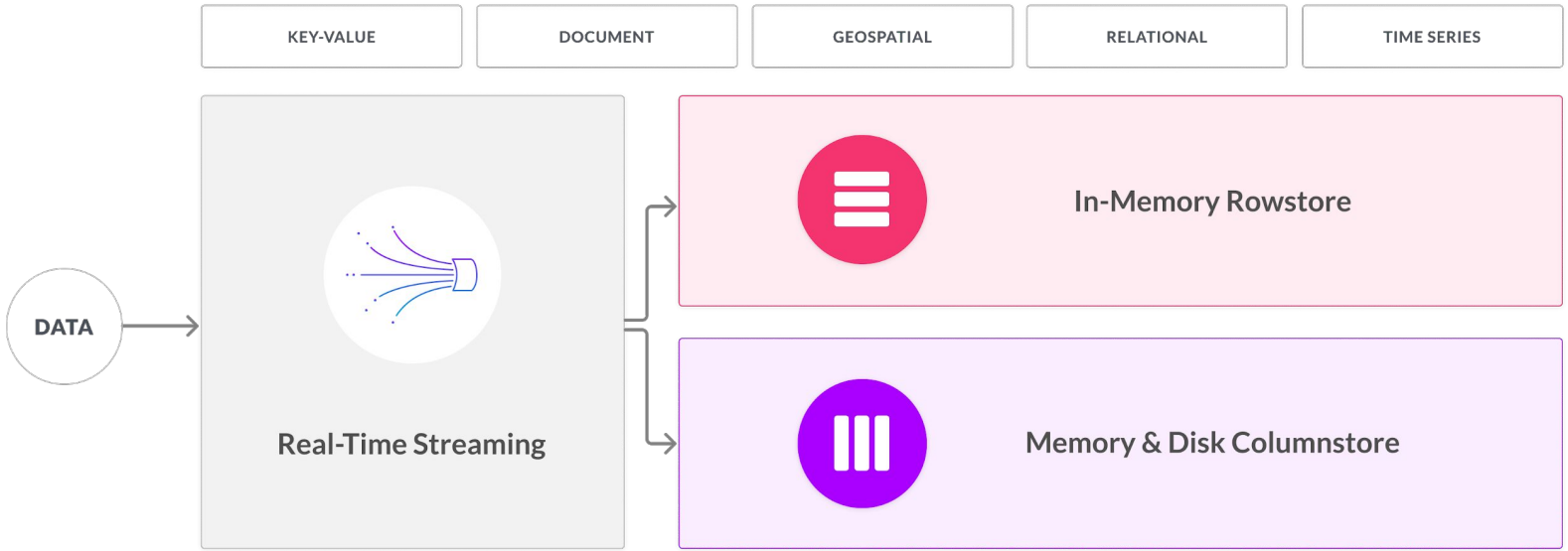
- **Aggregator Linear Scalability**
  - High concurrent OLTP and OLAP
  - High throughput data ingestion
- **Leaf Linear Scalability**
  - High data volume
  - High throughput ELT

**Data is replicated between Leaf Nodes for HA**



# MemSQL table types

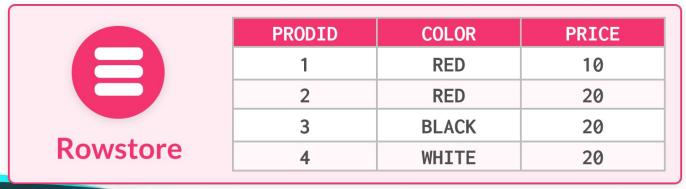
# Two Storage Engines



# Row and Column Orientations

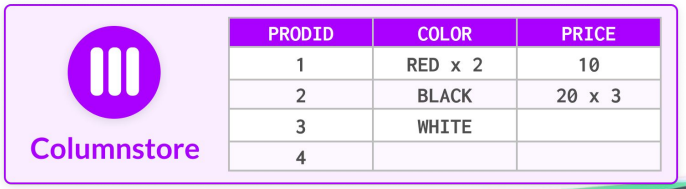
## In Memory Rowstore

- All data in memory, but also persisted to disk
- Scan ~40M rows / second / core
- Streaming or Batch Insert
- Select / Update Individual Rows Efficiently
- Row lock on update and delete
- Sparse Compression




## Memory Optimized Columnstore

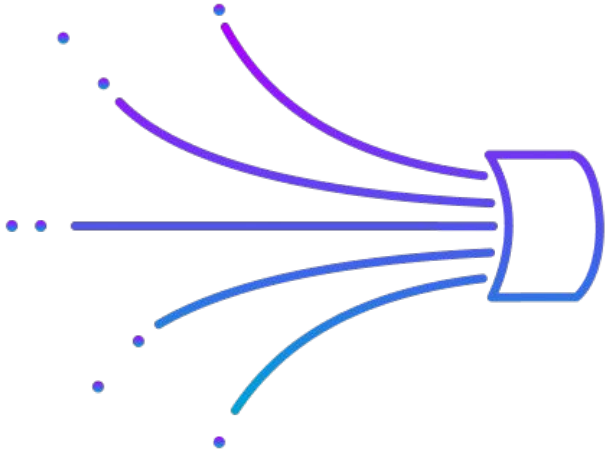
- Data Stored on SSDs
- Scan >200M+ rows / second / core
- Streaming or Batch Insert
- ~80% Compression
- Row  lock on update and delete



# MemSQL Pipelines

# Pipelines

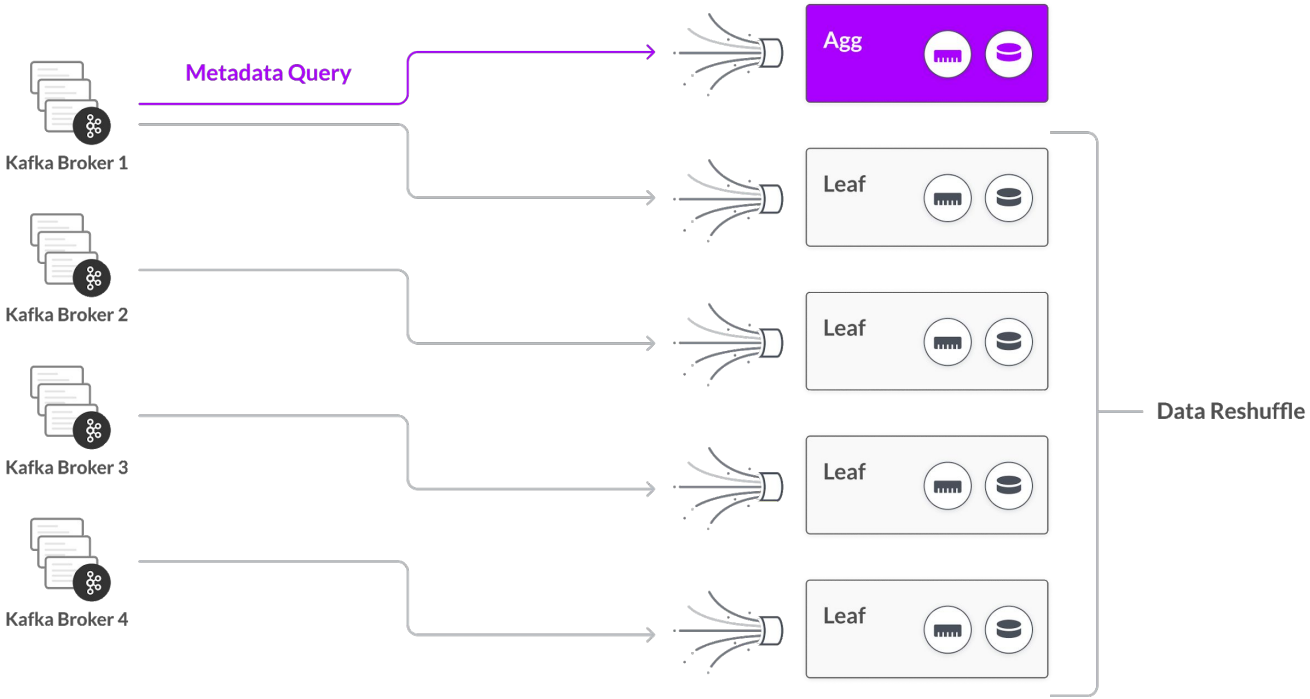
- Streaming ingestion
- Fast parallel bulk loading
- Transactional Consistency
- Exactly-Once Semantics
- Ingest Data into a Stored Procedure
- Native integrations with Kafka, AWS S3, Azure Blob, HDFS, NFS, etc.



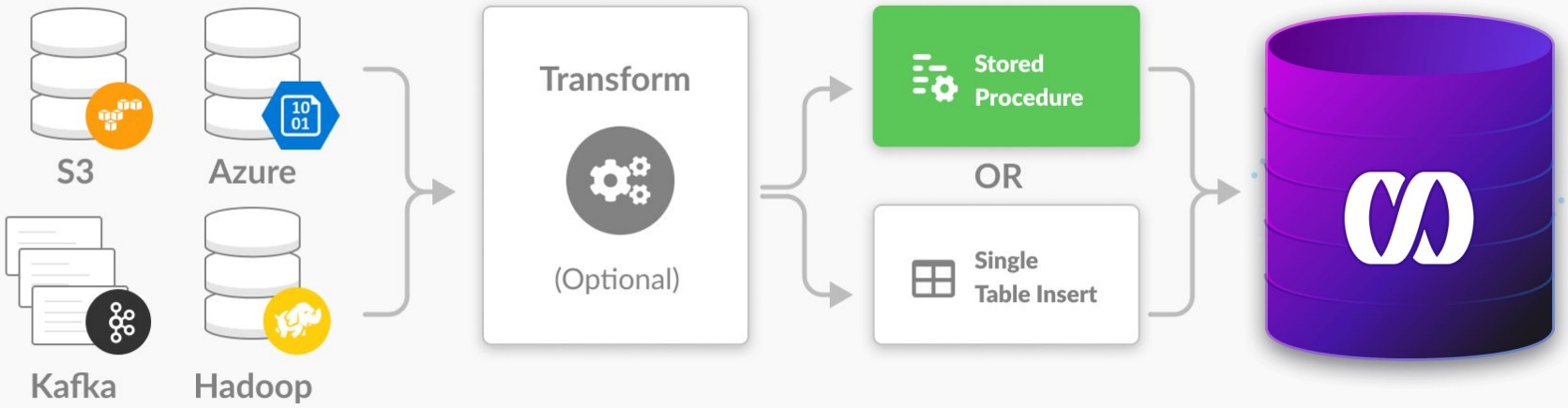
# MemSQL Pipelines Sequence


1. MemSQL polls for changes from a data source system.
2. MemSQL pulls the data into it's memory space (no commit) where a transform can be applied.
3. The data is committed in a transaction (and in parallel)

# MemSQL Pipelines Sequence



# Pipeline to Procedure



# Geospatial Functionality

# Accelerating the Value of Geospatial Data

## Store, query, and index geographic data types including polygons and points to support area, distance, location analytics in real-time

MemSQL makes geospatial data a first-class citizen: just as easy to use, at scale, at great speed and high throughput, as any other kind of relational data. MemSQL geospatial is a partial implementation of the OpenGIS standard for spatial processing.

- **Data types:** MemSQL supports Points, Linestrings, Polygons and Multipolygons natively using the industry-standard Well-Known Text (WKT) format;
- **Topological filter functions:** Contains, Intersects, Approximation, and Within Distance;
- **Measurement functions:** Length, Area, Distance
- **Indexing:** MemSQL leverages its fast, massively parallel, lock-free indexing for geospatial objects;
- **Spatial Joins:** MemSQL tables can be joined by their spatial relationships.

# MPSQL

# Procedural SQL Language Extensions ( MPSQL )

- ## Stored Procedures (SPs)

  Stored procedures can accept input parameters, query tables using SQL statements, call UDFs, define custom logic using control flow statements and variable assignment, and optionally return a value. Stored procedures can also be called across databases.

- ## User-Defined Scalar-Valued Functions (UDFs)

  UDFs can accept input parameters, call other UDFs, define custom logic using control flow statements and variable assignment, and return a value.

- ## User-Defined Table-Valued Functions (TVFs)

  TVFs can accept input parameters, execute a single SELECT statement, and return the result as a table-typed value.

- ## User-Defined Aggregate Functions (UDAFs)

  UDAFs support creation of custom aggregation logic, beyond the built-in aggregate functions supplied by MemSQL.

# Client Connectivity

# Easy to Manage and Tune with MemSQL Studio

- Execute DML and DDL commands and observe results
- Monitor Performance and Capacity
- API-based Deployment and Configuration
- Full Administration and Scaling
- Query Profiler
- Visual Query Plans

# Compatibility with what is familiar

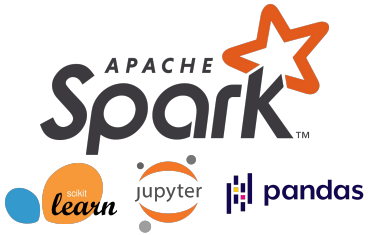

- Connectors for all major BI and analytics platforms
- Run queries using MemSQL Studio or MySQL Workbench
- memSQL Python Connector
- memSQL Spark Connector

memSQL

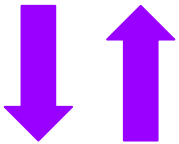
# MemSQL: Spark Connector 3.0

February 2020

# Spark and MemSQL: Better Together




MemSQL Spark Connector:
Full SQL Pushdown



memSQL

- Leverage existing models in-place with the MemSQL Spark Connector (true Spark DataSource)
- Push down models built in Python, Scala or Java and leverage the speed and scale of memSQL (20X performance improvement on gradient boosting)
- Applicable Spark dataframe operations get translated to true SQL and executed, with high performance and excellent concurrency

***Together: The fastest data platform for integration and analytics***

# Features

- Compatible with Spark 2.3 and 2.4
- True Spark data source compatible with JDBC Reader/Writer API
- Implemented as a native SparkSQL plugin
- Robust SQL Pushdown support (including partial pushdown)
- Leverages MemSQL LOAD DATA to accelerate ingest from Spark via compression
- Takes advantage of all the latest and greatest features in MemSQL 7.0+

# Requirements

You need the following to get started:

- MemSQL environment running version 6.7 or greater (6.8+ recommended)
- Apache Spark Environment 2.3+
- MariaDB JDBC driver 2.0 and above (this will get pulled in automatically as a dependency)

# Downloading

- You can find the connector and code at: https://github.com/memsql/memsql-spark-connector
- You can also add this to your library via spark-shell , using Maven, or SBT
- Instructions for each can be found on our documentation

memSQL

# Examples

February 2020

# Configuring the Connector: Example

**See all configuration options here**

```
spark.conf.set("spark.datasource.memsql.ddlEndpoint",
"memsql-master.cluster.internal")
spark.conf.set("spark.datasource.memsql.dmlEndpoints",
"memsql-master.cluster.internal,memsql-child-1.cluster.internal:3307")
spark.conf.set("spark.datasource.memsql.user", "admin")
spark.conf.set("spark.datasource.memsql.password", "s3cur3-pa$$word")
```

# Example: Load + Read Data

## Scala - reads data from table 'foo' in MemSQL

```
val df = spark.read
    .format("memsql")
    .option("ddlEndpoint", "memsql-master.cluster.internal")
    .option("user", "admin")
    .load("foo")
```

## Spark SQL - registers table 'bar' from MemSQL

```
CREATE TABLE bar USING memsql OPTIONS ('ddlEndpoint'='memsql-master.cluster.internal','dbtable'='foo.bar')
```

# Example: Write Data

## Scala - Write data from dataframe 'df' in Spark to table 'foo.bar'

```
df.write
    .format("memsql")
    .option("loadDataCompression", "LZ4")
    .mode(SaveMode.Overwrite)
    .save("foo.bar") // in format: database.table
```

# Datasource API

**All operations supported by the dataSource API are supported by the connector**

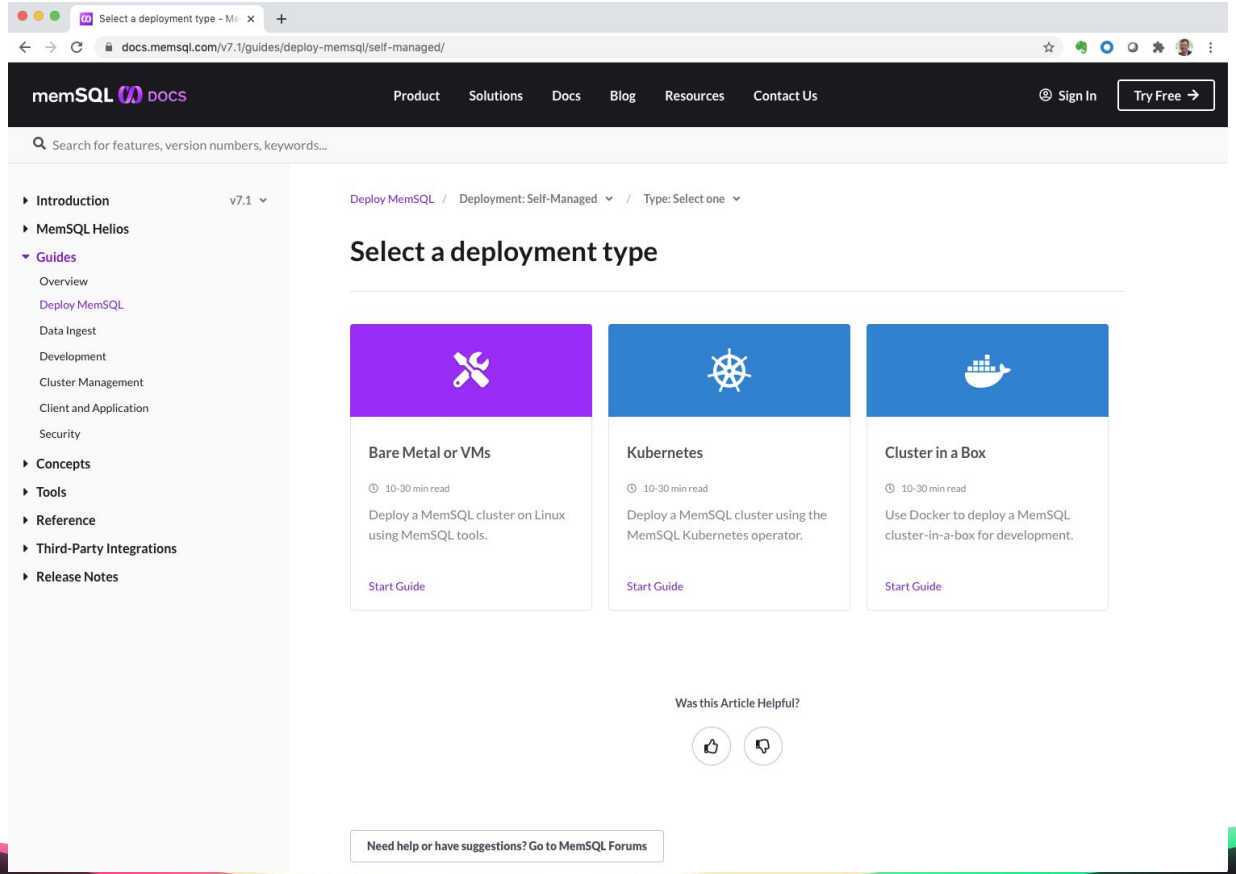
[Dataframe Operations - Spark Documentation]

# Documentation Page

MemSQL Spark Connector Reference: MTA Hackathon

# MemSQL Deployment Options

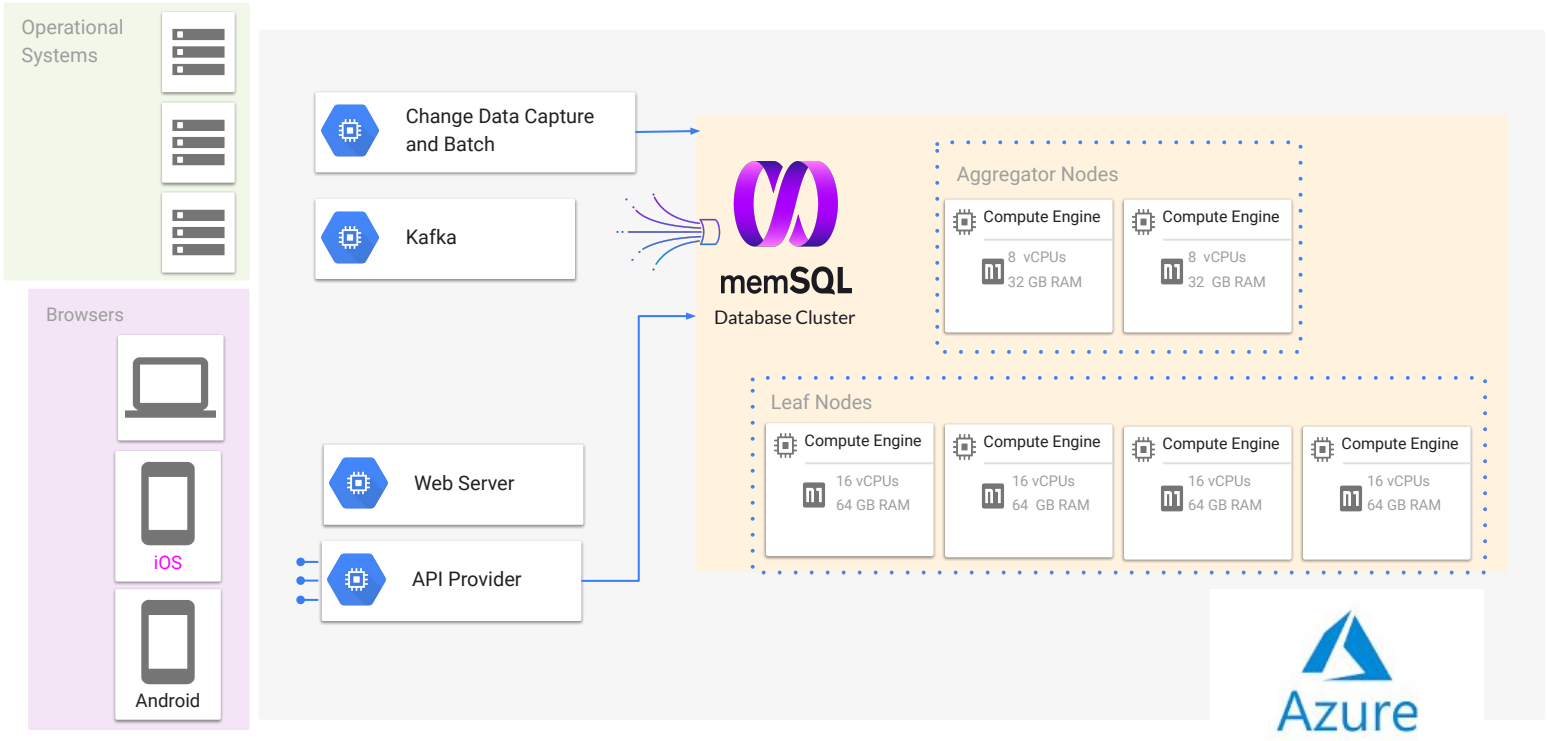
# MemSQL Deployment Options on MS Azure






memSQL

# MemSQL VM Deployment on MS Azure



# MemSQL VM Deployment on Azure

- **Determine appropriate number of VMs to spin up to support your MemSQL Cluster**
  - Determine the number of Aggregators (scales concurrency)
  - Determine the number of Leaves you would like to Spin Up (scales workload and data capacity)

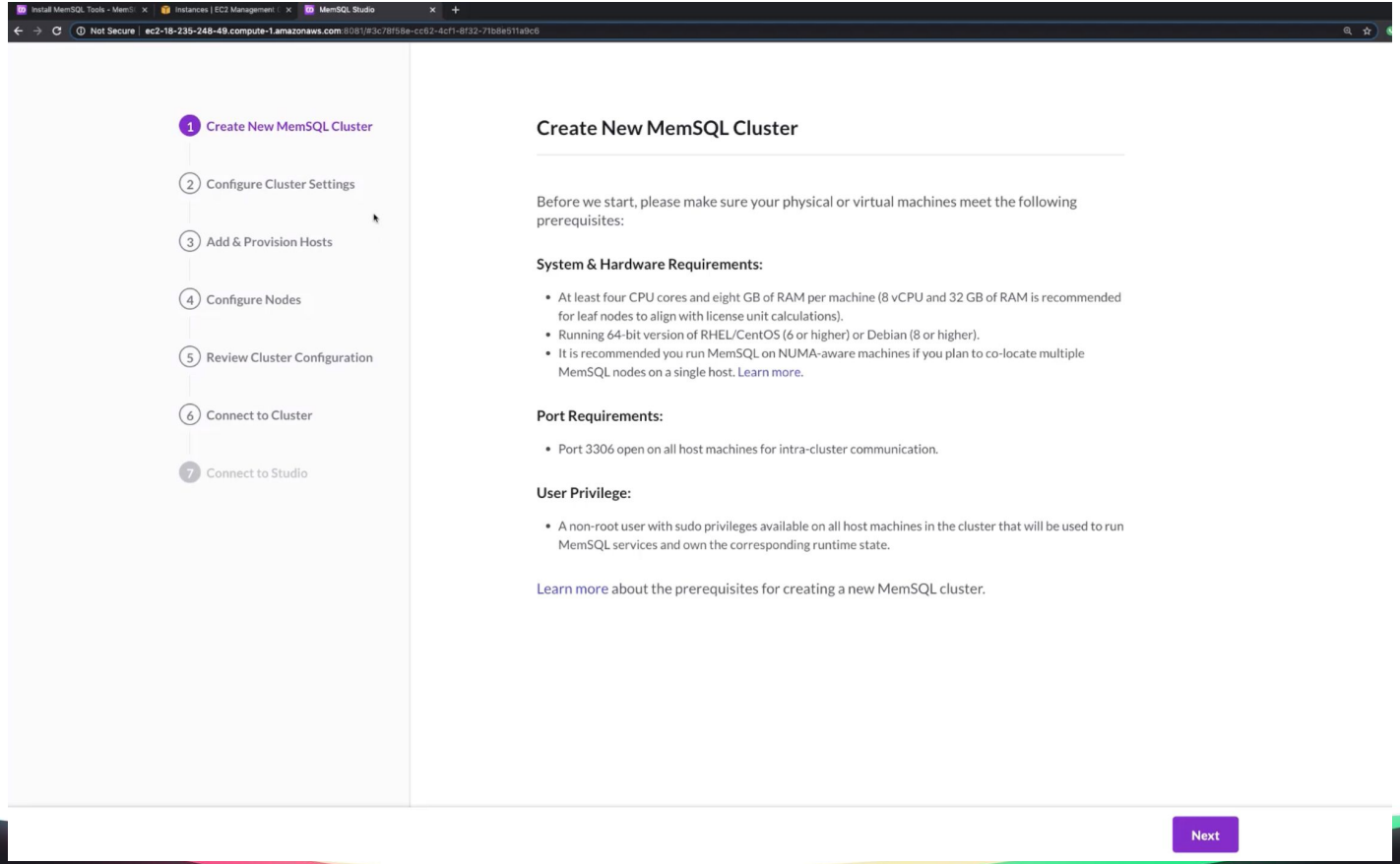
- **Follow Instructions from this MemSQL VM Deployment Guide starting here:**
  - https://docs.memsql.com/v7.1/guides/deploy-memsql/self-managed/linux/step-1/

**MemSQL UI Installation Videos:**

Getting Started with MemSQL Setup UI

MemSQL Setup UI

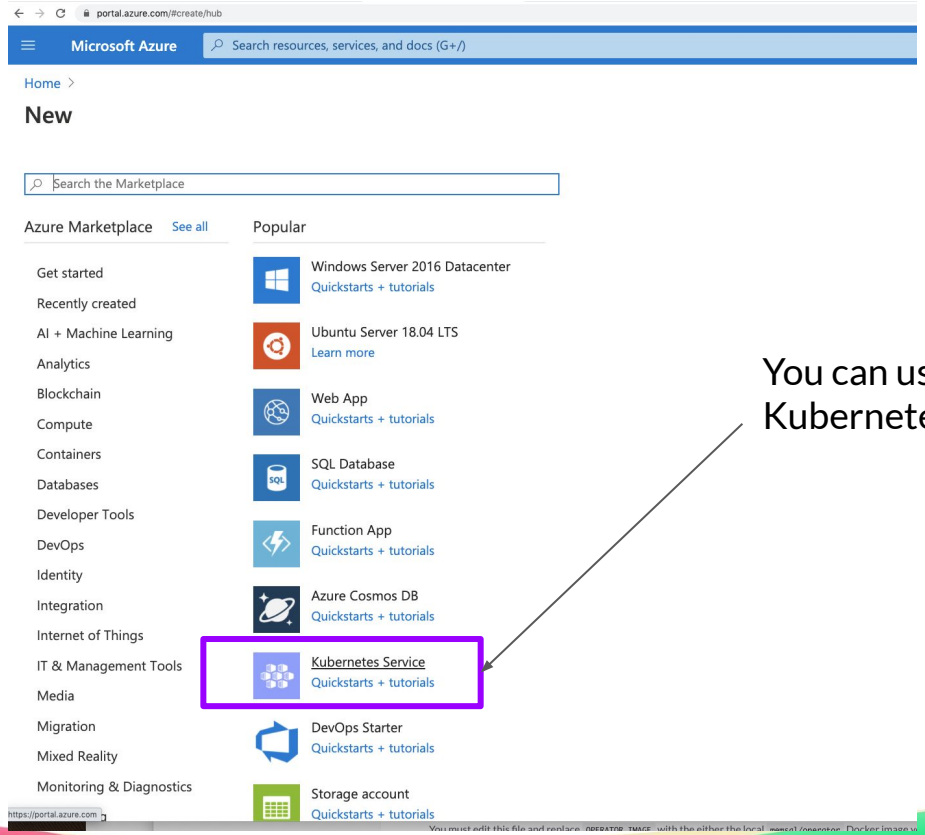
# MemSQL VM Deployment - UI Installation





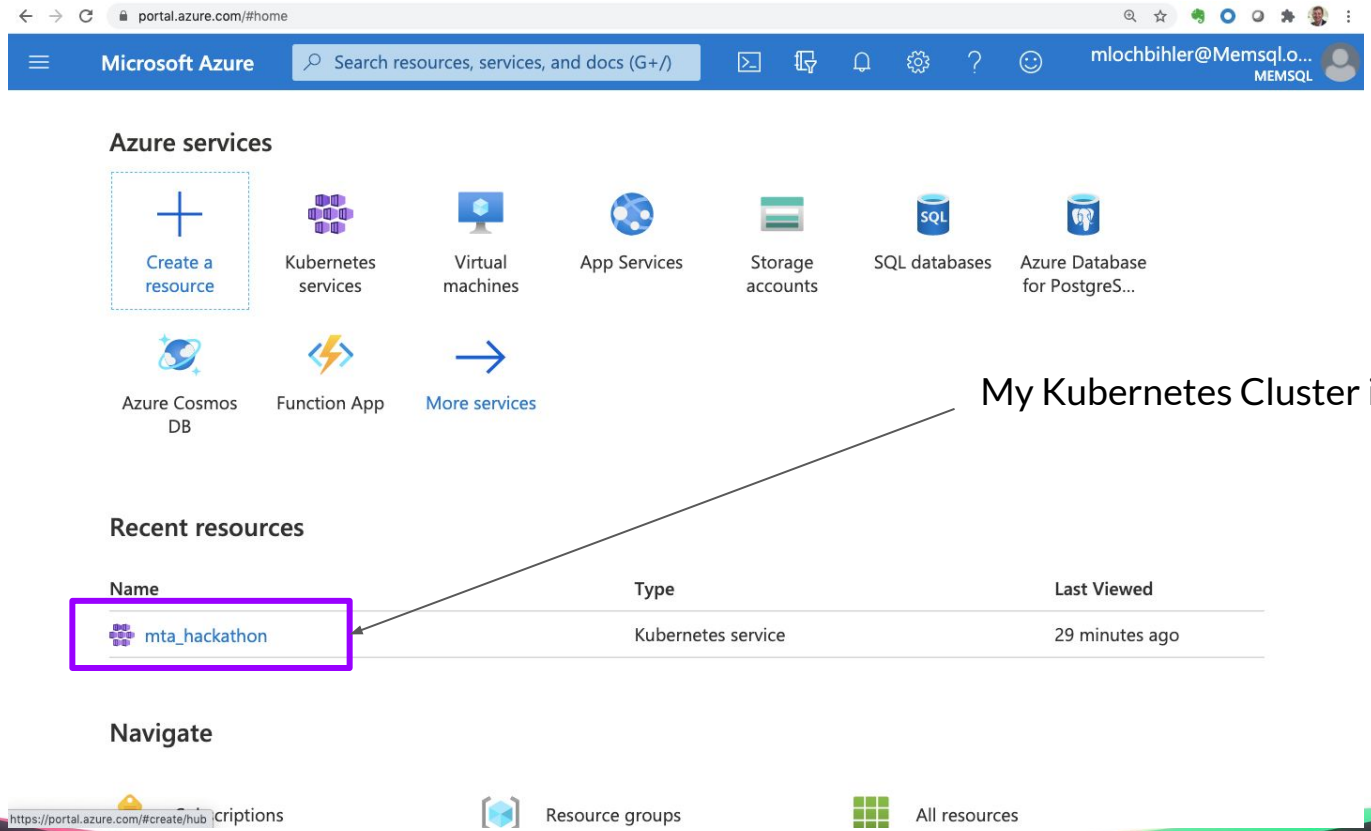

memSQL

# MemSQL Kubernetes Deployment on Azure

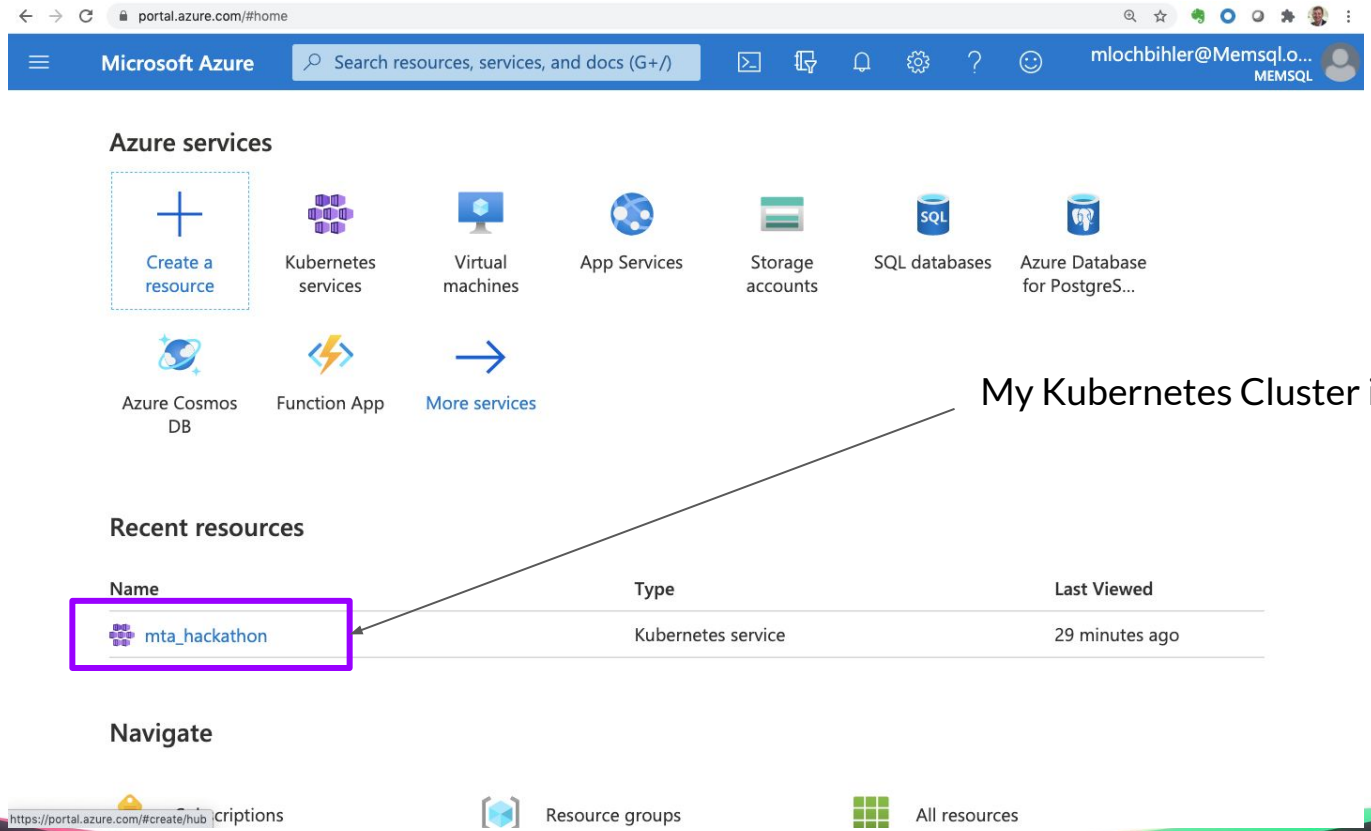



You can use the Kubernetes Service

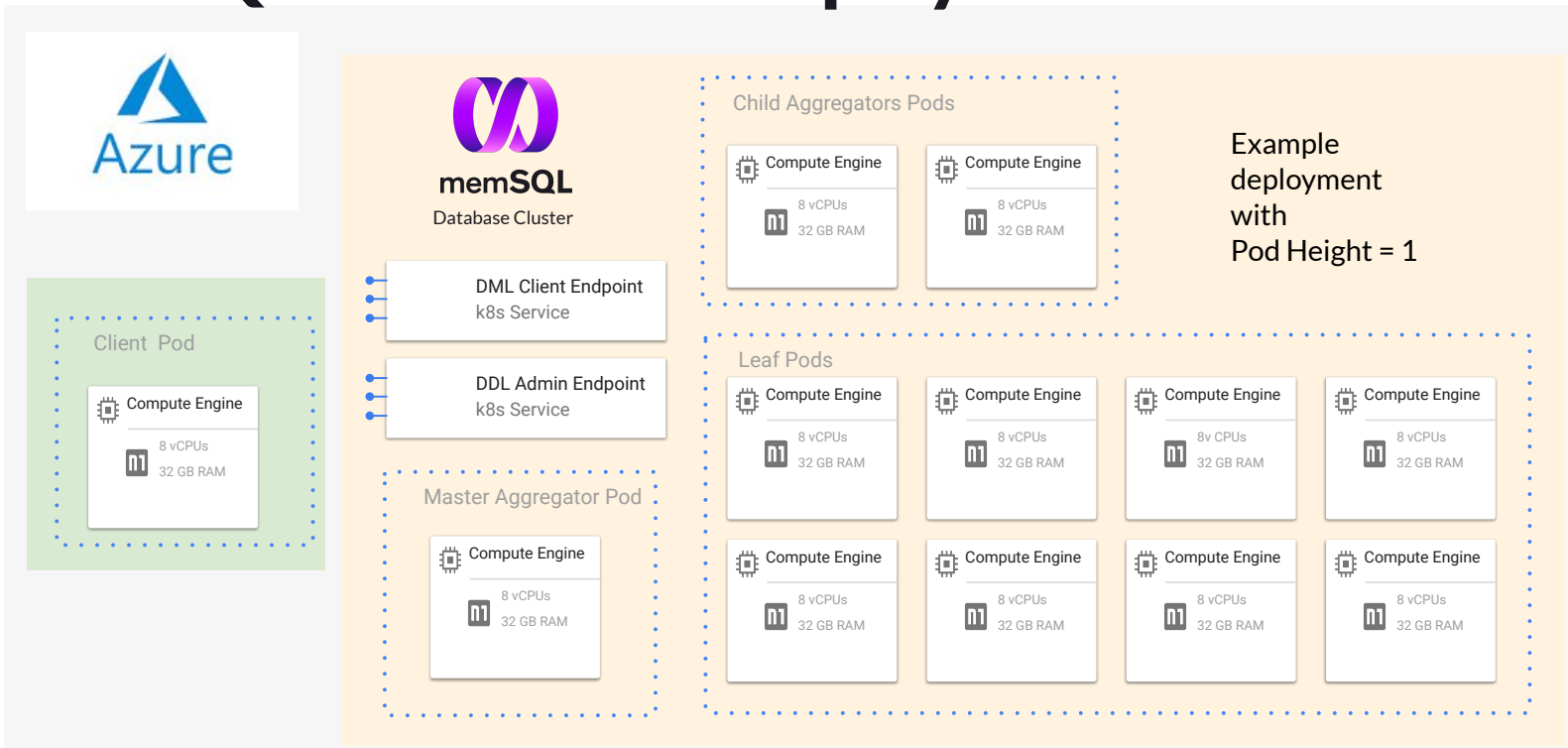
# MemSQL Kubernetes Deployment on Azure



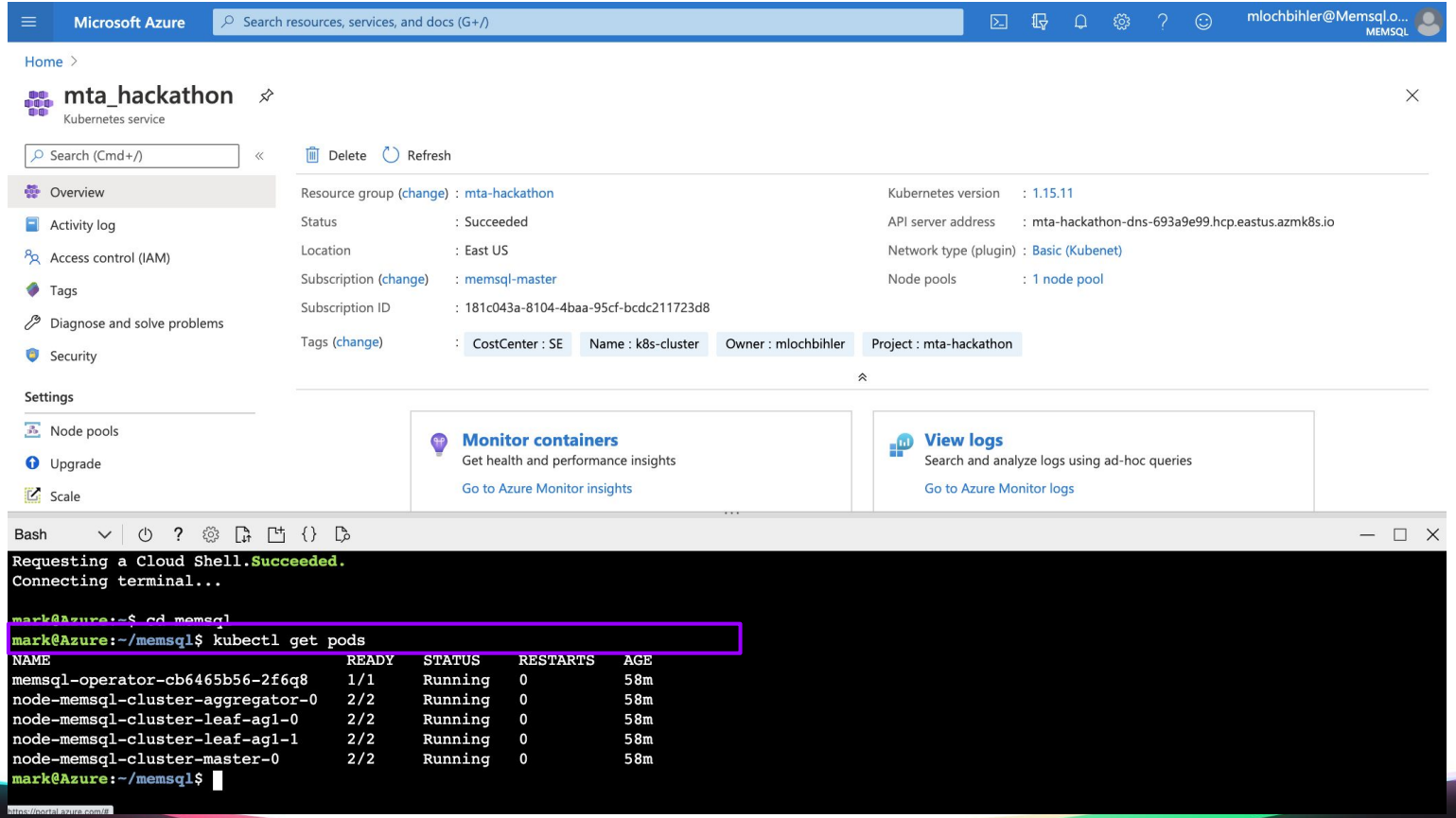
# MemSQL Kubernetes Deployment on Azure



# MemSQL Kubernetes Deployment on Azure



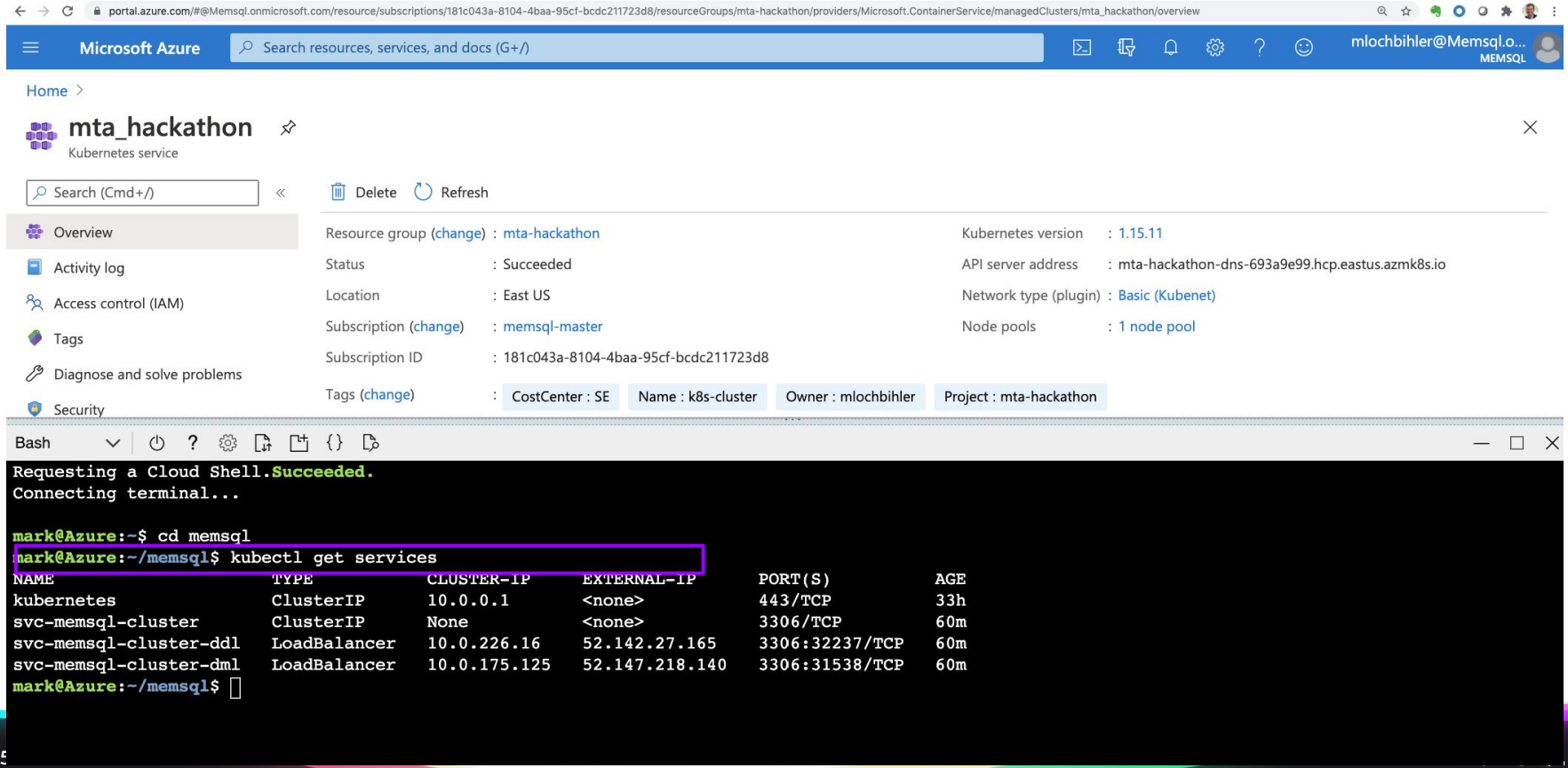
# MemSQL Kubernetes Deployment on Azure



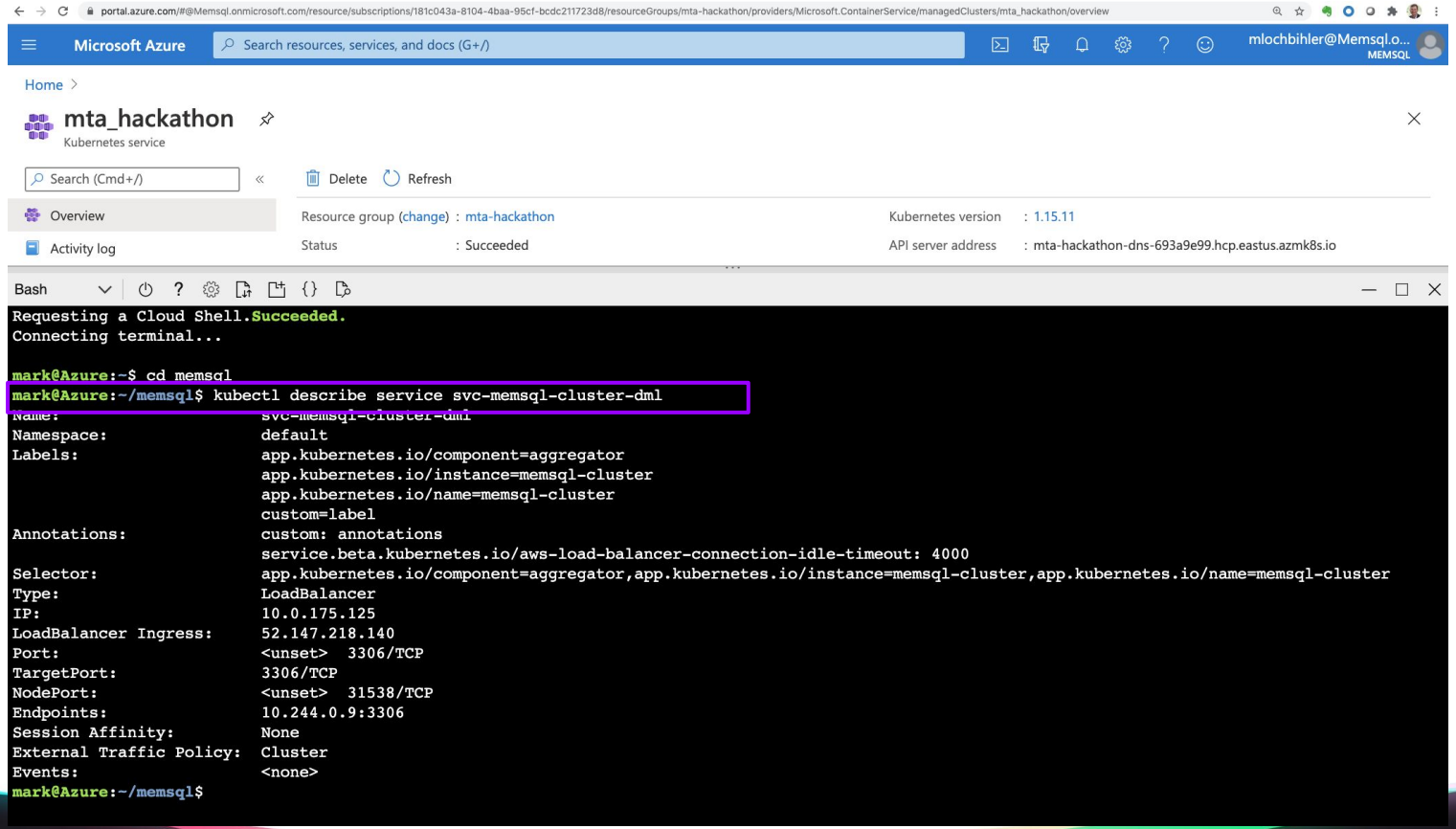



memSQL

# MemSQL Kubernetes Deployment on Azure



# MemSQL Kubernetes Deployment on Azure



# MemSQL Kubernetes Deployment on Azure