Openworks 2023 analytics workshop Instructions

Run the following steps on your VM

1. Provide instructor your email to be added to the team account (or sign up on your own) **cloud.mariadb.com**

2. Let's start by deploying Xpand in SkySQL

   “+” -> “Launch Cloud Database” -> “Transactions” -> “Xpand Distributed SQL” -> “AWS” -> “us-east-1” -> “4x16” -> “io1” -> “200GB” -> “20x IOPS” -> “+ x3 nodes” -> “<19 characters firstname lastname>xpand” -> “Add the current IP” -> “Disable SSL/TLS” -> “Launch Service”

3. Install docker

```
sudo su
sudo yum install -y yum-utils git
sudo yum-config-manager --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
sudo yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
```

4. Start docker

```
sudo systemctl start docker
```

5. Clone the columnstore docker project

```
sudo rm -rf mariadb-columnstore-docker
git clone
https://github.com/mariadb-corporation/mariadb-columnstore-docker.git
```

6. Change directory to the project and clone the .env_example

```
cd mariadb-columnstore-docker
sudo cp .env_example .env
sudo cp .secrets_example .secrets
sed -i "s|MAXSCALE=.*|MAXSCALE=true|g" .env
```

7. Start the project which will launch your containers (~3 min)

```
./run_project
```

   a. **(Don't run)** unless you get stuck and need to restart from step 5

```
docker compose down
docker rm -f $(docker ps -a -q)
docker volume rm $(docker volume ls -q)
docker compose up -d
./run_project
```

8. Lets copy the data into our containers filesystem (~ 2 min)

```
cd ..; time ls *.csv *.ddl | xargs -I % sh -c 'docker cp %  mcs1:/'
```

9. Enter the primary node container

```
docker exec -it mcs1 bash
```

10. Check the cluster status and start if it shows offline

```
mcsStatus
```

11. Create user, schema and Import the data (~ 12 min)

```
mariadb -e "CREATE DATABASE IF NOT EXISTS cs;"
mariadb -e "CREATE DATABASE IF NOT EXISTS inno;"
password="Ab1c23%$"
mariadb -e "CREATE USER DBA@'%' IDENTIFIED BY 'Ab1c23%^$'"
mariadb -e "GRANT ALL ON *.* TO  DBA@'%'";
mariadb -u DBA -p'Ab1c23%^$' -e  "select 'Auth Success'"
mariadb inno < schemas.ddl
mariadb cs < schemas.columnstore.ddl
mariadb cs -e "create table flights_unordered like flights;"

cpimport cs airlines airlines.csv -s "," -E '"'
cpimport cs airports airports.csv -s "," -E '"'
cpimport cs flights flights.csv -s "," -E '"'
cpimport cs flights_unordered flights_unordered.csv -s "," -E '"'
time mariadb inno -e "LOAD DATA INFILE '/flights.csv' INTO TABLE flights
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED BY '\n'"
&
```

12. Run some sample queries - connect via the mariadb client first

```
mariadb cs
```

```
--(1) Total Flights (~0.199s)
select count(*) from flights;

--(2) Which airlines arrive on schedule? (~1.250s)
select AVG(arr_delay) arrival_delay, carrier from flights where carrier in
('UA', 'AA', 'AS', 'DL', 'HA', 'NW', 'US', 'WN') group by carrier order by
carrier;

--(3) How has the arrival delay been trending over the years? (~1.813)
SELECT AVG(arr_delay) arrival_delay, carrier, Year FROM flights GROUP BY
carrier, Year ORDER BY Year,carrier desc limit 100;

--(4) Which are the worst airports? based on the number of delays (3.067)
select avg(dep_delay) delay, origin, count(*) AS cnt from flights group by
origin having cnt > 500000 order by delay desc limit 10;

--(5) What is the best day of the week to travel? (~0.866s)
```

```
select day_of_week, AVG(arr_delay) avg_delay from flights group by day_of_week
order by avg_delay desc;

--(6) Most popular flight routes in USA (~5.999s)
select origin, dest, count(*) count from flights group by origin, dest order
by count desc limit 10;
```

13. Stats comparing transactional engine vs analytical (Nothing to run just an FYI)

`LOAD DATA INFILE '/flights.csv' INTO TABLE inno.flights` - 482s vs ~93s

Query 1 - `select count(*) from flights;` - 613s vs ~0.199s

Query 2 - `select AVG(arr_delay) arrival_delay` - 244s vs ~1.25s

Query 5 - `select day_of_week, AVG(arr_delay)` - 1661s vs ~0.86s

14. See how the data is organized by partition/segment (1 segment contains 2 extents)

```
SELECT calShowPartitions('flights','year');

SELECT calShowPartitions('flights_unordered','year');
```

15. See the underlying individual extents

```
select objectid from calpontsys.syscolumn where tablename='flights' and
columnname='year';
select PARTITION_ID,SEGMENT_ID,DBROOT, MIN_VALUE, MAX_VALUE from
information_schema.columnstore_extents where object_id=3022 order by
PARTITION_ID ASC, BLOCK_OFFSET ASC, SEGMENT_ID ASC;

select objectid from calpontsys.syscolumn where tablename='flights_unordered'
and columnname='year';
select PARTITION_ID,SEGMENT_ID,DBROOT, MIN_VALUE, MAX_VALUE from
information_schema.columnstore_extents where object_id=3065 order by
PARTITION_ID ASC, BLOCK_OFFSET ASC, SEGMENT_ID ASC;
```

16. Compare ordered columnstore data versus the unordered

```
-- order data is faster (0.082s)
select origin, dest, count(*) count from flights where fl_date="2021-01-1"
group by origin, dest order by count desc limit 10;

-- view underlying extents (notice twice as slow 0.170s )
select origin, dest, count(*) count from flights_unordered where
fl_date="2021-01-1"  group by origin, dest order by count desc limit 10;
```

17. Now let's learn how to analyze the query execution plan of queries in columnstore

Note: you can run `select calFlushCache();` to invalidate caches and simulate first time run.

```
select calSetTrace(1);
select origin, dest, count(*) count from flights where fl_date="2021-01-1"
group by origin, dest order by count desc limit 10;
select calGetTrace();

select calSetTrace(1);
```

```
select origin, dest, count(*) count from flights_unordered where
fl_date="2021-01-1"  group by origin, dest order by count desc limit 10;
select calGetTrace();

select calFlushCache();
```

Learn More:
https://mariadb.com/kb/en/analyzing-queries-in-columnstore/#viewing-the-columnstore-query-plan

18. Mass deleting data

```
SELECT calShowPartitions('flights','fl_date');
SELECT calDisablePartitions('flights','0.0.1');
SELECT calShowPartitions('flights','fl_date');

SELECT calEnablePartitions('flights','0.0.1');
SELECT calShowPartitions('flights','fl_date');
#SELECT calDropPartitions('flights','0.0.1');
```

19. Load data infile S3 (new feature in latest version)

Yes a little slower but fully mariadb client driven, does not require cli/sudo access

```
create table airports_clone like airports;
SET columnstore_s3_region='us-west-1';
SET columnstore_s3_key='AKIA5437P37NLC3SVTEY';
SET columnstore_s3_secret='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';
CALL columnstore_info.load_from_s3("s3://openworks-workshop/data/",
"airports.csv", "cs", "airports_clone", ",", "\"", "" );
exit;
exit; -- returns to ec2 vm
```

20. **(Likely already done for you)** Install aws cli and configure credentials

```
yum install unzip -y;
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip";
unzip awscliv2.zip;
sudo ./aws/install;
sudo mv /usr/local/bin/aws /usr/bin/aws;
aws configure set default.s3.max_concurrent_requests 50
aws configure;
aws s3 ls s3://openworks-workshop;
```

Key: AKIA5437P37NLC3SVTEY
Secret: 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

21. Download backup script and xpand schema

```
aws s3 cp s3://openworks-workshop/data/schemas.xpand.ddl .;
aws s3 cp s3://openworks-workshop/parallel-import.bash .;
```

Any columnstore questions?

Extra if time permits see the end for backup/restore

22. Ideally your Xpand cluster in skysql is deployed now and you can begin deploying your serverless analytics environment so that we can use that later.

    “+” -> “Launch Cloud Database” -> “Analytics” -> “Serverless Analytics Spark SQL” -> “AWS” -> “us-east-1” -> “<19 characters firstname lastname>spark” -> “allow access from ip” -> “Launch”

23. Now lets jump into Xpand while serverless analytics deploys
    In the skysql portal -> “Connect” -> “Connecting using MariaDB CLI”
    Password can be copied from the **Default password** section above

24. From our VM, lets connect
    But first Install mariadb client

```
yum install wget -y ; wget
https://dlm.mariadb.com/enterprise-release-helpers/mariadb_es_repo_setup
;chmod +x mariadb_es_repo_setup;
./mariadb_es_repo_setup --token="xxxxxxx-xxxxx" --apply
--mariadb-server-version="10.6"
yum install MariaDB-client -y
```

25. Check the ip address of the VM where on

```
echo $( curl -s 'https://api.ipify.org?format=json');
```

26. Add the external ip address from the prior step to your skysql Xpand instance whitelist
    “Manage” -> “Security Access”
    For the left value example (add /32 to the end of it): “**54.188.139.59/32**”
    For the right value add any notes about the ip address like “**allens docker vm box**”
    Now click **“Save”**

27. Now let's connect from the VM
    The security settings from prior should take 2-3 min to take effect.
    Get this information by hitting the “**Connect**” button on your xpand tile from the portal

```
mariadb --host dbpwf10350625.sysp0001.db1.skysql.mariadb.com --port 3306
--user dbpwf10350625 --default-character-set=utf8 -p"cpaMBm3654qKcPC+.waTU"
```

28. Prepare variables based of the prior step

```
CONNECT="mariadb --host dbpwf10350625.sysp0001.db1.skysql.mariadb.com --port
3306 --user dbpwf10350625 --default-character-set=utf8"
PASSWORD="cpaMBm3654qKcPC+.waTU"
```

29. Create an easy alias to connect to skysql

```
echo "alias connect='$CONNECT -p'$PASSWORD'" >> ~/.bashrc
source ~/.bashrc
export CONNECT=$CONNECT
```

```
export PASSWORD=$PASSWORD
connect -e "select 'Auth Success via Alias' as Login"
$CONNECT -p"$PASSWORD" -e "select 'Auth Success via ENV vars' as Login"
```

30. Import the data into Xpand on SkySQL

```
cd /home/centos
connect -e "CREATE DATABASE IF NOT EXISTS bts;"
connect bts < schemas.xpand.ddl
time connect bts -e "LOAD DATA INFILE '/home/centos/airlines.csv' INTO TABLE
airlines FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED
BY '\n'";
time connect bts -e "LOAD DATA INFILE '/home/centos/airports.csv' INTO TABLE
airports FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"' LINES TERMINATED
BY '\n'";
```

31. Parallel Import the ~69 million records into Xpand with helper script

```
time bash parallel-import.bash bts flights flights.csv 16 1000000
/tmp/csv_splits;
```

<break>

32. Why Xpand?

MariaDB Server InnoDB versus MariaDB Xpand InnoDB

Query 1 - `select count(*) from flights` ~ 613s vs ~8s

Query 2 - `select AVG(arr_delay) arrival_delay` ~ 244s vs ~23s

Query 5 - `select day_of_week, AVG(arr_delay)` ~ 1661s vs ~27s

```
Connect bts
```

```
--(1) Total Flights (~58s first query then ~8s)
select count(*) from flights;

--(2) Which airlines arrive on schedule? (~23s)
select AVG(arr_delay) arrival_delay, carrier from flights where carrier in
('UA', 'AA', 'AS', 'DL', 'HA', 'NW', 'US', 'WN') group by carrier order by
carrier;
```

33. Now now lets baseline some queries before adding the columnar index

```
-- (1) Total Flights in the first week of march 2020; (~16s)
select count(*) from flights where fl_date between "2020-03-01" and
"2020-03-07";

-- (2) Which airlines arrive on schedule in December from ATL? (~5.9s)
select arrival_delay, airline from (select AVG(arr_delay) arrival_delay,
```

```
carrier from flights  where month=12 and origin="ATL" group by carrier ) f
inner join airlines a on f.carrier=a.iata_code order by arrival_delay;

-- (3) What was Delta airline's JFK flight delay average last year? (~6s)
SELECT carrier, AVG(arr_delay),Year as arrival_delay FROM flights WHERE year =
YEAR(CURRENT_DATE) - 1 and origin="JFK" and carrier="DL" GROUP BY carrier,
year;

-- (4) What percent of flights got cancelled per airline last year? (~5.5s)
select airline, total_flights,(total_cancellations/total_flights)*100 as
percent_cancelled  from (SELECT carrier, COUNT(*) as total_flights,
SUM(cancelled) as total_cancellations FROM flights WHERE year = 2022 AND month
= MONTH(CURRENT_DATE) - 1 GROUP BY carrier) f inner join airlines a on
f.carrier=a.iata_code order by percent_cancelled desc;

-- (5) Top 5 average carrier delay flights on Christmas Eve 2022 (~12.3s)
SELECT origin, dest, AVG(carrier_delay) as average_carrier_delay FROM flights
WHERE fl_date = "2022-12-24" GROUP BY origin, dest ORDER BY
average_carrier_delay DESC LIMIT 5;
```

34. Now add the columnar index to this table (~15 min)

```
ALTER TABLE bts.flights ADD COLUMNAR INDEX operational_analytics
(`carrier`,`year`,`month`,`day`,`day_of_week`,`fl_date`,`fl_num`,`origin`,`des
t`,`cancelled`,`cancellation_code`,`arr_delay`,`carrier_delay`,`weather_delay`
);
```

35. Rerun the queries to compare with columnar index?

MariaDB Xpand InnoDB vs MariaDB Xpand Columnar Indexes

Query 1 - `select count(*) where fl_date between` ~ 16s vs ~0.4s

Query 2 - `select AVG(arr_delay) where month= and origin= join` ~ 5.9s vs ~1.6s

Query 4 - `select SUM(canceled) where year= and month= join` ~ 5.5s vs ~0.7s

Query 5 - `select AVG(carrier_delay) where fl_date=` ~12.3s vs ~0.8s

```
-- (1) Total Flights in the first week of march 2020; (~0.4s)
select count(*) from flights where fl_date between "2020-03-01" and
"2020-03-07";

-- (2) Which airlines arrive on schedule in December from ATL? (~1.6s)
select arrival_delay, airline from (select AVG(arr_delay) arrival_delay,
carrier from flights  where month=12 and origin="ATL" group by carrier ) f
inner join airlines a on f.carrier=a.iata_code order by arrival_delay;

-- (3) What was Delta airline's JFK flight delay average last year? (~0.7s)
SELECT carrier, AVG(arr_delay),Year as arrival_delay FROM flights WHERE year =
YEAR(CURRENT_DATE) - 1 and origin="JFK" and carrier="DL" GROUP BY carrier,
year;

-- (4) What percent of flights got cancelled per airline last year? (~0.66s)
select airline, total_flights,(total_cancellations/total_flights)*100 as
percent_cancelled  from (SELECT carrier, COUNT(*) as total_flights,
SUM(cancelled) as total_cancellations FROM flights WHERE year = 2022 AND month
```

```
= MONTH(CURRENT_DATE) - 1 GROUP BY carrier) f inner join airlines a on
f.carrier=a.iata_code order by percent_cancelled desc;

-- (5) Top 5 average carrier delay flights on Christmas Eve 2022 (~0.8s)
SELECT origin, dest, AVG(carrier_delay) as average_carrier_delay FROM flights
WHERE fl_date = "2022-12-24" GROUP BY origin, dest ORDER BY
average_carrier_delay DESC LIMIT 5;
```

Any Questions?

36. Now lets jump into serverless analytics

a. Find the appropriate "**connect**" button for your serverless analytics deployment
b. Then click the link under "**Getting started using SkyBook**"

37. Select "**Notebook**" -> "**Create new note**" -> **any name** -> **save**

38. Find your "catalog" - your database should automatically exist (if SA created after)

```
show catalogs;
```

*Hint - Shift + Enter = Run code block

a. Create a connection to our mariadb columnstore deployment

```
- on EC2 VM
docker ps;
echo $( curl -s 'https://api.ipify.org?format=json');
```

Confirm port 3311 is bound to maxscale

Run the following in Serverless Analytics - Replacing the IP address and other appropriate fields

```
CREATE OR REPLACE MARIADB CATALOG AllensColumnstore options
('url'='jdbc:mariadb://54.196.128.235:3311?useSSL=false&trustServerCertifica
te=true', 'user'='DBA', 'password'='Ab1c23%^$')
```

39. Define variables to reference our catalogs easier - Replace with names of catalogs from prior

```
%scala
{
/* Name of xpand node */
z.put("catalog_xpand", s"allenherreraxpand")
z.put("catalog_columnstore", s"AllensColumnstore")
}
```

40. Copy the data from columnstore into serverless analytics saving it in parquet format (~10 min)
* Note: remove `%scala` from the first line

```
use spark_catalog;
create table airlines using parquet as select *  from
{catalog_columnstore}.cs.airlines;
```

```
create table airports using parquet as select *  from
{catalog_columnstore}.cs.airports;
create table flights using parquet as select *  from
{catalog_columnstore}.cs.flights;
```

41. See the tables in your “catalog”

```
use {catalog_xpand}.bts;
show databases;
show tables;
```

42. Describe a table from Xpand

```
describe table {catalog_xpand}.bts.airlines;
```

43. Preview the data in Xpand - federated query

```
SELECT * FROM {catalog_xpand}.bts.airlines;
```

44. Once the import from Columnstore -> Serverless Analytics is done
    Run some sample analytic queries - view them as charts or graphs
    * might need to configure settings to use correct dimensions for charts
    * rerun the queries to onboard more resources / first couple queries ~30s

```
use spark_catalog;

--(1) Total Flights; (~ss)
select count(*) from flights;

--(2) Which airlines arrive on schedule? (~2s)
select AVG(arr_delay) arrival_delay, carrier from flights where carrier in
('UA', 'AA', 'AS', 'DL', 'HA', 'NW', 'US', 'WN') group by carrier order by
carrier;

--(3) How has the arrival delay been trending over the years? (~2s)
SELECT AVG(arr_delay) arrival_delay, carrier, Year FROM flights GROUP BY
carrier, Year ORDER BY Year,carrier desc limit 100;

--(4) Which are the worst airports? based on the number of delays (~2s)
select avg(dep_delay) delay, origin, count(*) AS cnt from flights group by
origin having cnt > 500000 order by delay desc limit 10;

--(5) What is the best day of the week to travel? (~1s)
select day_of_week, AVG(arr_delay) avg_delay from flights group by day_of_week
order by avg_delay desc;

--(6) Most popular flight routes in USA (~2s)
select origin, dest, count(*) count from flights group by origin, dest order
by count desc limit 10;

--(7) Most popular flight routes in USA (~2s)
select concat(origin,'_',dest) as flight, count(*) count from flights group by
concat(origin,'_',dest) order by count desc limit 10;
```

45.

```
```

46. Extra Time Exercise - Backup & Restore - Download backup/restore scripts

```
aws s3 cp s3://openworks-workshop/simpleBackup.bash .;
aws s3 cp s3://openworks-workshop/simpleRestore.bash .;
```

Copy backup script to all nodes

```
docker cp simpleBackup.bash mcs1:/ ;
docker cp simpleBackup.bash mcs2:/ ;
docker cp simpleBackup.bash mcs3:/ ;
docker cp simpleRestore.bash mcs1:/ ;
docker cp simpleRestore.bash mcs2:/ ;
docker cp simpleRestore.bash mcs3:/ ;
```

Turn off columnstore to ensure consistency

```
docker exec -it mcs1 bash
mcsShutdown
```

Open two other SSH windows

```
ssh centos@54.226.6.71

ssh centos@54.226.6.71
```

On window (1) Begin backup of node 1

```
bash simpleBackup.bash
```

In another window (2) start the backup of node 2

```
docker exec -it mcs2 bash
bash simpleBackup.bash
```

In another window (3) start the backup of node 3

```
docker exec -it mcs3 bash
bash simpleBackup.bash
```

After you see a message similar to
“[+] Backup Complete @ /tmp/backups/04-18-2023”

Start columnstore & drop a database in columnstore

```
mcsStart
mariadb cs -e "drop table flights_unordered";
```

Find your restore command

```
cat /tmp/backups/04-26-2023/restore.job
```

On node (1) turn off columnstore and copy mysql backup to the other nodes (~3 min)

```
mcsShutdown
/etc/init.d/mariadb stop
exit;
mkdir /tmp/tmpbackups/
docker cp mcs1:/tmp/backups/04-26-2023/mysql /tmp/tmpbackups/
docker cp /tmp/tmpbackups/mysql mcs2:/tmp/backups/04-26-2023/
docker cp /tmp/tmpbackups/mysql mcs3:/tmp/backups/04-26-2023/
docker exec -it mcs1 bash
```

Stop mariadb server on each node - replace with mcs2, mcs3 for other nodes

```
/etc/init.d/mariadb stop
```

Now lets restore each node starting with mcs1 (1) but running the same on (2) and (3)

```
chmod +x simpleRestore.bash
./simpleRestore.bash -l 04-26-2023 -bl /tmp/backups/ -bd Local -s
LocalStorage --dbroots 3
```

After the restore - start mariadb on each node (mcs1, mcs2, mcs3)

```
/etc/init.d/mariadb start
```

Start columnstore

```
mcsStart
```

Check the table you deleted

```
mariadb cs -e "select count(*) from flights_unordered";
```

47.

```

```