**Couchbase Autonomous Operator Workshop**

**Lab Handbook**

Lab 6: Analytics

# Legend - Please Read!

We use the following formatting conventions in all lab modules:

* Yellow highlighted values (in the text or on the screenshots) must be replaced with the correct value in your environment, e.g., 192.168.61.101
* Orange highlighted values are used for additional emphasis in the context of a task
* Commands in **Courier New** font in bold are what you are expected to execute (see below):
  + The “**$**” prefix indicates the command to be executed in the terminal on your local computer (Mac and Linux).
  + The “**>**” prefix is only used to indicate Windows-specific commands to be executed on your local computer. We will continue to refer to the Windows *Command Prompt* (*cmd.exe*) as “terminal” in all the labs, and all universal examples of commands will assume the Mac and/or Linux prefix, i.e. “**$**”, and some commands may need to be modified before being run on Windows, e.g. change / to \, remove trailing & character to run commands in the background.
  + The text without any prefix is the output of the command.
  + The “**#**” prefix indicates a command that has to be run inside the shell of a pod. Commands in **blue bold font** are executed within the pod’s shell.

|  |
| --- |
|  |

* **Red font** indicates important instructions, please pay very close attention to those.
* Items in **bold** or *italic* font are names or UI elements to pay attention to and/or click on.

Unless stated otherwise, all commands, parameters, settings, logins, passwords, etc. are case-sensitive!

The output of the commands in your terminal will likely differ from the output shown in the labs. Generally, this should not be an issue, as long as the output is similar, and there are no obvious error messages. Please ask for help if you get stuck.

Remember: copying and pasting the code and/or commands may produce errors due to auto-formatting in text editors, e.g. regular dash characters get replaced with extended ASCII dashes, regular quotation marks with fancy quotes; some special characters may be invisible. Such characters will not let your code run correctly. If this happens to you, try typing the commands instead of copying and pasting them.

# 

# Lab Setup

In this lab, we’ll learn how to:

* Add a pod/node with Analytics service enabled to the cluster using Couchbase Autonomous Operator
* Import additional data to load 1,000,000 orders via **cbimport**
* Inspect the catalogs in the Metadata dataverse in the cluster
* Create DATASETs to create the target datasets in Analytics for the information of interest.
* Activate the connectivity between Analytics and a Couchbase Server cluster instance. In our case, the data of interest is in the local server (i.e., it is managed by the Data service in the same cluster).
* Perform ad-hoc Analytics queries and inspect the results - These queries are on a dataset consisting of 1,000,000 orders stemming from 24 products

## Step 1: Deploy a node for the Analytics service

Edit the *couchbase-cluster.yaml* file and add the section highlighted in orange below to add a node to the cluster to run the Analytics service and increase the data service quota as well as the couchmart bucket’s quota.

This file should be located under the *couchbase-operator-workshop/resources* directory on your machine.

**Note**: Use spaces and not tabs when editing/modifying YAML files; [tab characters are invalid inside YAML files](https://yaml.org/faq.html) and will result in errors. It is recommended to use consistent indentation sizing, e.g. the file shown below uses 2 spaces for indentation.

|  |
| --- |
| apiVersion: couchbase.com/v1  kind: CouchbaseCluster  metadata:  name: cb-example  spec:  baseImage: couchbase/server  version: enterprise-6.0.1  authSecret: cb-example-auth  exposeAdminConsole: true  adminConsoleServices:  - data  cluster:  dataServiceMemoryQuota: 1024  indexServiceMemoryQuota: 256  searchServiceMemoryQuota: 256  eventingServiceMemoryQuota: 256  analyticsServiceMemoryQuota: 1024  indexStorageSetting: memory\_optimized  autoFailoverTimeout: 30  autoFailoverMaxCount: 3  autoFailoverOnDataDiskIssues: true  autoFailoverOnDataDiskIssuesTimePeriod: 120  autoFailoverServerGroup: false  buckets:  - name: couchmart  type: couchbase  memoryQuota: 512  replicas: 1  ioPriority: low  evictionPolicy: valueOnly  conflictResolution: seqno  enableFlush: true  enableIndexReplica: false  servers:  - size: 3  name: data\_service  services:  - data  - size: 1  name: index\_query\_service  services:  - index  - query  - size: 1  name: search\_service  services:  - search  - size: 1  name: analytics\_service  services:  - analytics |

Once the file has been updated with the desired configuration, run the command shown below to deploy the additional pod to the Couchbase cluster.

Remember: use your unique namespace value and make sure your current working directory (cwd) is still *couchbase-operator-workshop*!

|  |
| --- |
| **$ kubectl apply -f resources/couchbase-cluster.yaml --namespace myuniquename**  couchbasecluster.couchbase.com/cb-example configured |

Wait a few seconds, then execute the following command to monitor the status of the newly deployed pod, i.e. *cb-example-0005*:

|  |
| --- |
| **$ kubectl get pods --namespace myuniquename -w**  NAME READY STATUS RESTARTS AGE  cb-example-0000 1/1 Running 0 103m  cb-example-0001 1/1 Running 0 102m  cb-example-0002 1/1 Running 0 101m  cb-example-0003 1/1 Running 0 38m  cb-example-0004 1/1 Running 0 28m  cb-example-0005 1/1 Running 0 2m13s  couchbase-operator-66ccb5946-nmss9 1/1 Running 0 106m  couchmart-669bdcf89d-v7kxr 1/1 Running 0 106m |

**Note:** the -w option makes this is a blocking call; hit CTRL-C to kill the process.

## Step 2: Use N1QL to modify the product profit margins

Navigate to the [Couchbase Web Console](http://localhost:8093/ui/index.html#!/query/workbench) of the index and query node, i.e. [http://localhost:8093](http://localhost:8093/ui/index.html#!/query/workbench). Use the credentials below to login to the Couchbase Web Console (case-sensitive!):

|  |  |
| --- | --- |
| ***Username*** | **Administrator** |
| ***Password*** | **password** |

If you are unable to access the Couchbase web console then check to make sure that you still have local port 8093 forwarded to the index and query service pod’s port 8091. Run the following command to check which ports are currently being forwarded from your local machine to the corresponding pods:

|  |
| --- |
| **$ $ ps o command= | grep '[k]ubectl port-forward'**  kubectl port-forward cb-example-0000 8091:8091 --namespace myuniquename  kubectl port-forward cb-example-0003 8093:8091 --namespace myuniquename  kubectl port-forward cb-example-0004 8094:8091 --namespace myuniquename |

If 8093 is not listed above use the following command to forward the port to the index & query service pod:

|  |
| --- |
| **$ kubectl port-forward cb-example-0003 8093:8091 --namespace myuniquename &** |

Remember: use your unique namespace value!

Run the following *CREATE INDEX* N1QL statement in the [Query Workbench](http://localhost:8093/ui/index.html#!/query/workbench):

|  |
| --- |
| **CREATE INDEX couchmart\_product\_idx ON couchmart(type) WHERE type = "product"** |

Wait for that query to complete indicating that the index is ready, then run the following *UPDATE* N1QL statement to set a profit margin, i.e. cost vs price, of 20% on all documents of type *product:*

|  |
| --- |
| **UPDATE couchmart set cost=price \* 0.8 WHERE type = "product"** |

**Note:** You may decide to set a different profit margin (we used a reverse 20% discount in the query above) but you can specify any discount factor you like. (i.e cost = price \* 0.55 for a higher markup) or set a different markup for different products.

## Step 3: Use cbimport to import 1M documents

Access the shell of any one of the Couchbase pods in your namespace using *kubectl exec*. The name of all your pods can be determined by running *kubectl get pods* (shown below). Remember: use your unique namespace value!

|  |
| --- |
| **$ kubectl get pods --namespace myuniquename**  NAME READY STATUS RESTARTS AGE  cb-example-0000 1/1 Running 0 11h  cb-example-0001 1/1 Running 0 11h  cb-example-0002 1/1 Running 0 11h  cb-example-0003 1/1 Running 0 10h  cb-example-0004 1/1 Running 0 10h  cb-example-0005 1/1 Running 0 10h  couchbase-operator-66ccb5946-nmss9 1/1 Running 0 12h  couchmart-669bdcf89d-v7kxr 1/1 Running 0 12h |

The following command is an example of accessing the shell of *cb-example-0000*:

|  |
| --- |
| **$ kubectl exec -it cb-example-0000 bash --namespace myuniquename**  **#** |

Remember: use your unique namespace value!

Issue the following commands from inside one of the Couchbase pod’s shell:

|  |
| --- |
| **#** **wget http://bit.ly/cbsummitdata1M -O /tmp/couchmart\_1M\_formatted\_keys.json**  --2019-02-12 16:43:37-- http://bit.ly/couchmartjson1M  Resolving bit.ly (bit.ly)... 67.199.248.10, 67.199.248.11  Connecting to bit.ly (bit.ly)|67.199.248.10|:80... connected.  ...  **#** **/opt/couchbase/bin/cbimport json -c couchbase://localhost -b couchmart -u Administrator -p password -f list -d file:///tmp/couchmart\_1M\_formatted\_keys.json -g %\_id%**  Json 'file:///tmp/couchmart\_1M\_formatted\_keys.json' imported to 'http://localhost:8091' successfully |

**Note:** Wait for the last message to be displayed to indicate that the data was imported successfully.

The *cbimport*command will take some time to run and when it has completed successfully there will be 1,000,027 documents (or more depending on how many orders were placed) in the *couchmart* bucket, which can be verified on the [Couchbase Web Console](http://localhost:8091/ui/index.html#!/buckets?openedBucket=couchmart) under **Buckets**.

## 

## Step 4: Port forward and access the Analytics query workbenc

In a new terminal window, run the following *kubectl port-forward* command to map local port 8095 to remote port 8091 of the new pod we just added that is running the analytics service. Remember: use your unique namespace value!

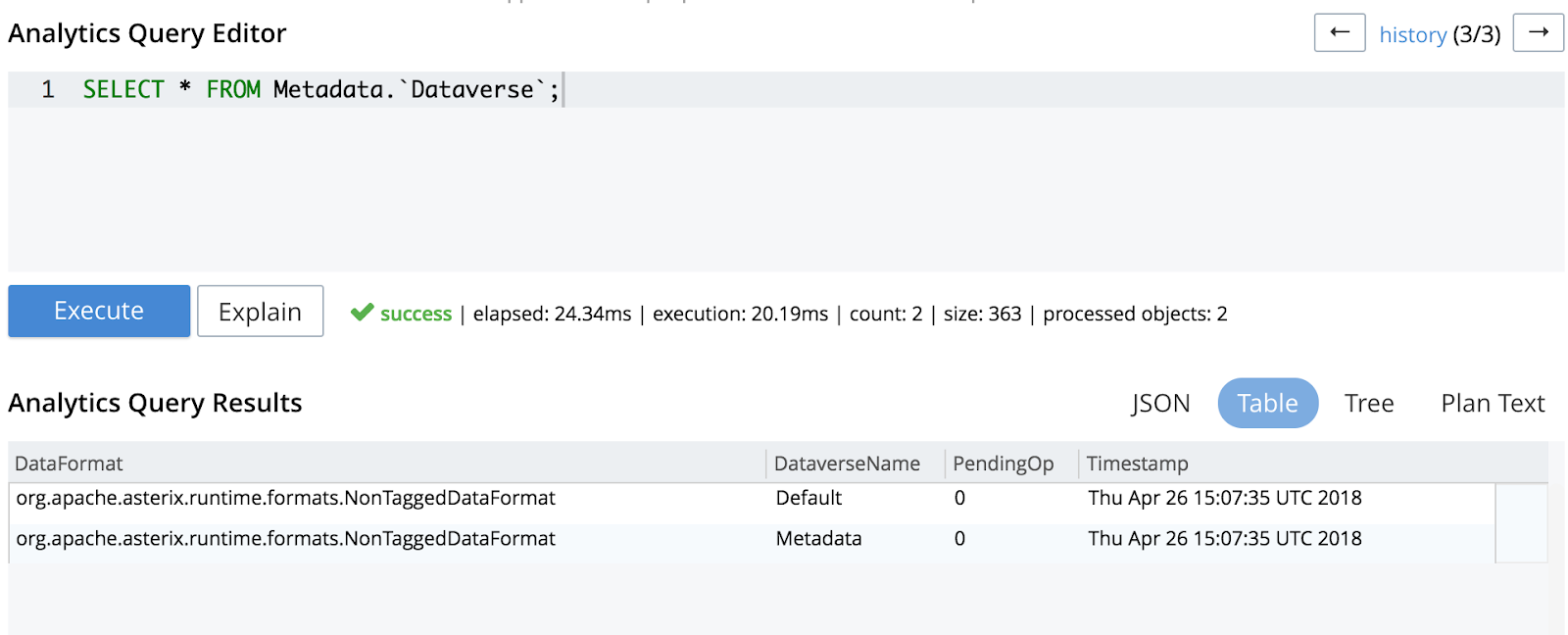
|  |
| --- |
| **$ kubectl port-forward cb-example-0005 8095:8091 --namespace myuniquename &** |

Navigate to the [Couchbase Web Console](http://localhost:8095) of the newly added node, e.g. <http://localhost:8095>. Use the credentials below to login to the Couchbase Web Console (case-sensitive!):

|  |  |
| --- | --- |
| ***Username*** | **Administrator** |
| ***Password*** | **password** |

In the [Analytics Query Workbench](http://localhost:8095/ui/index.html#!/cbas/workbench), run the following query:

|  |
| --- |
| **SELECT \* FROM Metadata.`Dataverse`** |



For a fresh Couchbase installation, you should be able to see the two dataverses. One is the **Metadata** dataverse, which is the system catalog. The second one, **Default**, which we will be using for the remainder of this lab -for simplicity- is available for holding analytics data.

## 

## Step 5: Create datasets and link them to the bucket

Our next task is to tell the **Default** dataverse about the Couchbase Server data (bucket with source data in the Data service) that we want to shadow, and which shadow datasets we want the Analytics data for that bucket to live.

Next, let’s create two shadow datasets to track the products and orders respectively (each command should be run individually):

|  |
| --- |
| **CREATE DATASET products ON couchmart WHERE `type` = "product"**  **CREATE DATASET orders ON couchmart WHERE `type` = "order"** |

Now, let’s connect the bucket to actually initiate the shadowing relationship between the two datasets and the Data bucket:

|  |
| --- |
| **CONNECT LINK Local** |

## Step 6: Verify that the datasets exist

Now let’s verify that the two datasets were successfully created (you should have already seen a successful indicator in the query editor for each query, but this is to verify what was done through the system catalog).

|  |
| --- |
| **SELECT ds.BucketName, ds.DatasetName, ds.`Filter`**  **FROM Metadata.`Dataset` ds**  **WHERE ds.DataverseName = "Default"** |



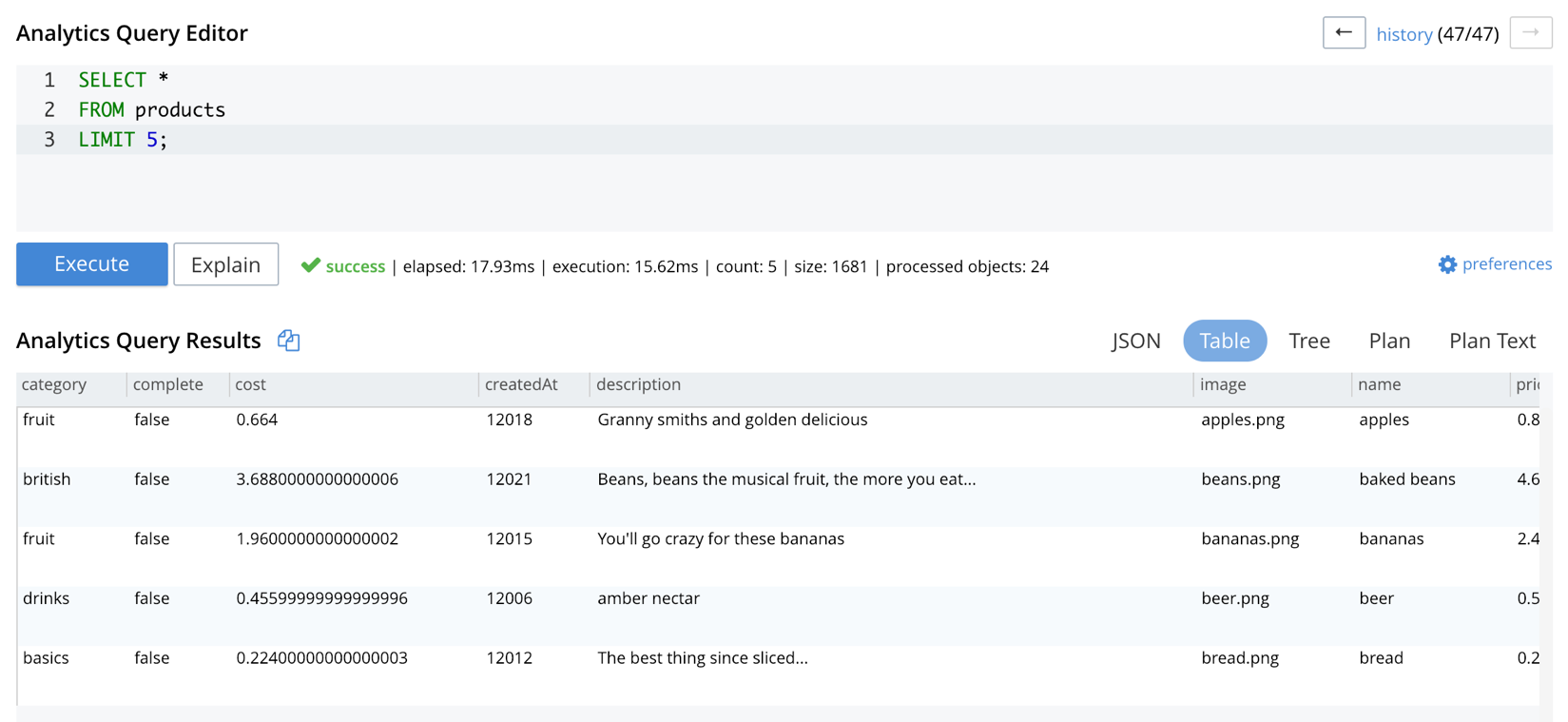
## 

## 

## Step 7: Try some simple Analytics queries

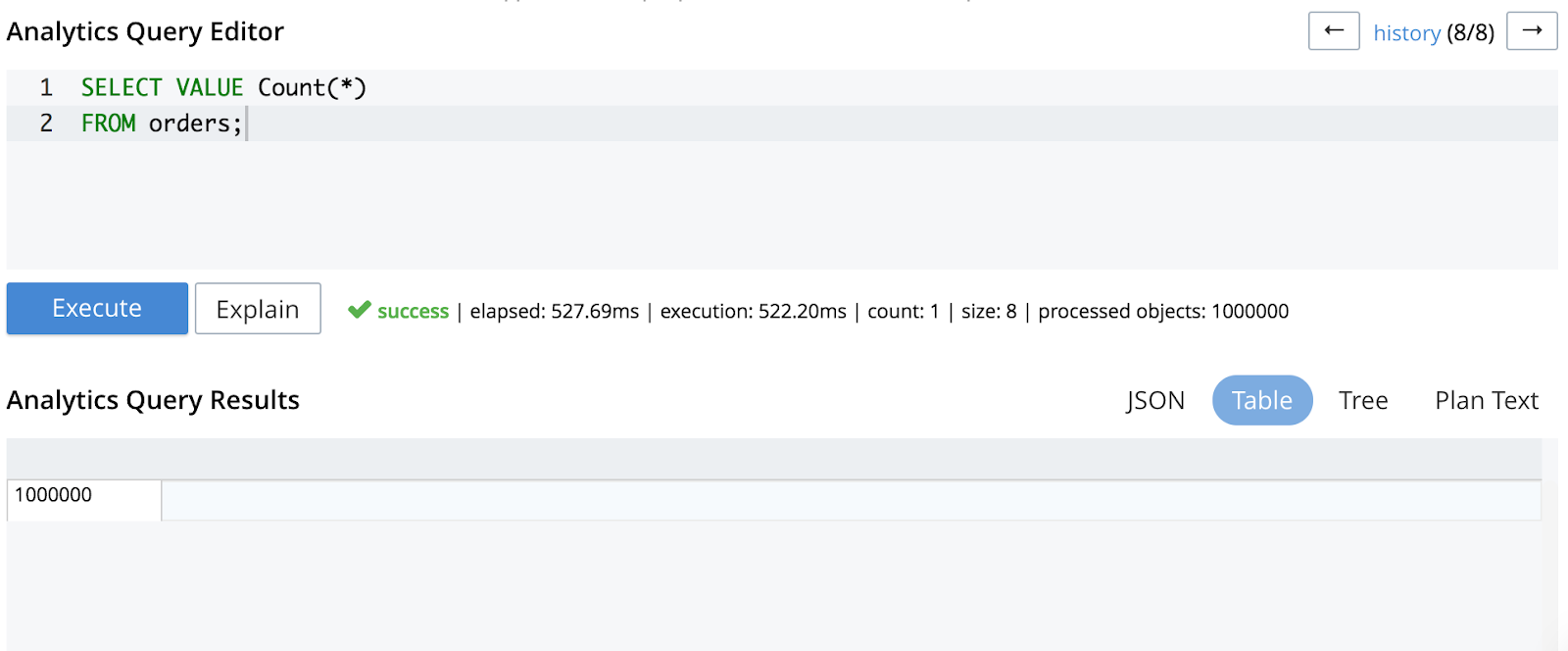
Now let’s give the Analytics query a spin by running a very simple query.

|  |
| --- |
| **SELECT \* FROM products LIMIT 5** |



As you can see, the query returned 5 products from the “*products”* dataset. Now let’s run another query, which will involve a simple aggregation to return the total number of orders.

|  |
| --- |
| **SELECT VALUE Count(\*) FROM orders** |



## 

## Step 8: Find the top 10 spending customers

Let’s issue an Analytics query to find the top 10 spending customers across the 1,000,000 orders stored.

|  |
| --- |
| **SELECT ord.name, sum(prd.price)**  **FROM orders ord, ord.`order` ord\_pid, products prd**  **WHERE ord\_pid /\*+ bcast\*/ = meta(prd).id**  **GROUP BY ord.name**  **ORDER BY sum(prd.price) DESC**  **LIMIT 10** |



That query aggregates data across 1,000,000 records and runs in approximately 15 seconds on our environment’s hardware configuration with a single Analytics node; adding more nodes would improve performance of the query.

## 

## Step 9: Find the top 10 selling products

Now let’s find the top 10 selling products.

|  |
| --- |
| **SELECT prd.name, count(\*)**  **FROM orders ord, ord.`order` ord\_pid, products prd**  **WHERE ord\_pid /\*+ bcast\*/ = meta(prd).id**  **GROUP BY prd.name**  **ORDER BY count(\*) DESC**  **LIMIT 10** |



## 

## Step 10: Find the top 10 grossing products

Now let’s find the top 10 grossing products ordered further by units sold.

|  |
| --- |
| **SELECT prd.name, count(\*) AS `items sold`, sum(prd.price) - sum(prd.cost) as profit**  **FROM orders ord, ord.`order` ord\_pid, products prd**  **WHERE ord\_pid /\*+ bcast\*/ = meta(prd).id**  **GROUP BY prd.name**  **ORDER BY sum(prd.price) - sum(prd.cost) DESC, count(\*) DESC**  **LIMIT 10** |



## 

## Step 11: Find the top 10 profitable customers

Our final Analytics query is to find the top 10 profitable customers further ordered by the units sold.

|  |
| --- |
| **SELECT ord.name, count(\*) AS `items sold`, sum(prd.price) - sum(prd.cost) as profit**  **FROM orders ord, ord.`order` ord\_pid, products prd**  **WHERE ord\_pid /\*+ bcast\*/ = meta(prd).id**  **GROUP BY ord.name**  **ORDER BY sum(prd.price) - sum(prd.cost) DESC, count(\*) DESC**  **LIMIT 10** |

