Apache Trino

Ahmet Aygün

02220224565

1. Giriş

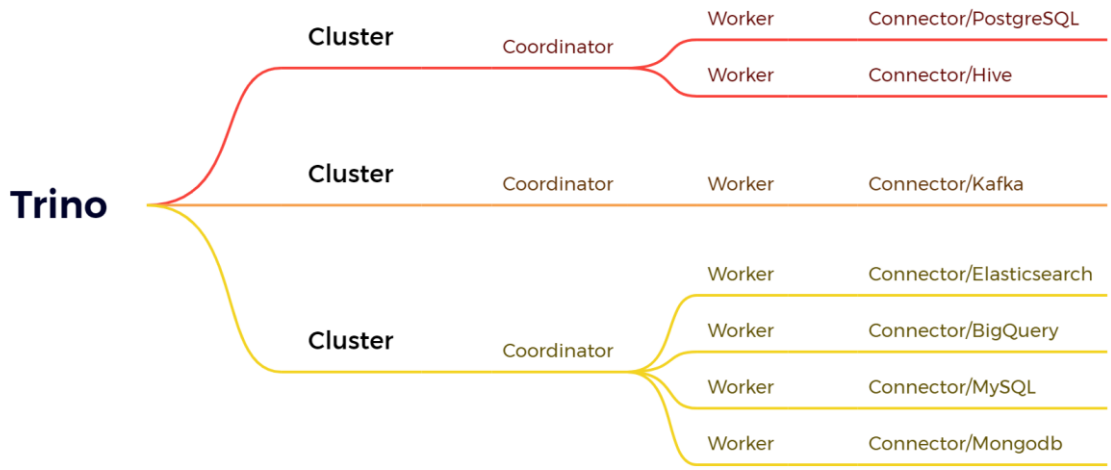
Apache Trino Nedir?

- Apache Trino, **açık kaynak** bir **SQL sorgulama motoru** olup, büyük veri kümelerini analiz etmek ve sorgulamak için tasarlanmıştır.
- Trino, özellikle **dağıtık veri kaynaklarına** (Hadoop, S3, Google Cloud Storage, Azure Data Lake, RDBMS) erişim sağlamak ve tek bir sorgu yüzeyinde sorgulama yapmak için idealdir.
- Facebook'ta **PrestoDB** olarak geliştirilmeye başlanmış, daha sonra topluluk tarafından Apache Trino olarak yeniden yapılandırılmıştır.

Kullanım Senaryoları:

- **Veri Göllerinde Sorgulama** (Data Lakes).
- **ETL İşlemleri** için SQL tabanlı çözümler.
- **Gerçek Zamanlı Analitik ve Raporlama**.
- **Veri Kaynağı Entegrasyonu**: Farklı veri kaynaklarını tek bir SQL sorgusu ile birleştirme.

2. Trino'nun Mimarisi



Mimari Bileşenler

Trino'nun mimarisi, genellikle **3 temel bileşenden** oluşur:

1. **Coordinator (Koordinatör):**
    a. Kullanıcı sorgularını alır ve sorguyu bir işlem planına dönüştürür.
    b. **İş planlama**, **optimizasyon**, ve **task scheduling** (görev zamanlaması) sorumluluklarına sahiptir.
    c. Veriyi işlemek için işçi nodelarını (worker) yönlendirir.
2. **Worker (İşçi) Node:**
    a. Koordinatörün oluşturduğu planı alır ve veriyi işlemek için görevleri yürütür.
    b. **Paralel ve dağıtık işlem** yaparak büyük veri setlerinde yüksek performans sağlar.
3. **Connector (Bağlayıcılar):**
    a. Farklı veri kaynaklarına bağlantı sağlar.
    b. **JDBC Connector** (MySQL, PostgreSQL), **Hive Connector**, **Kafka Connector**, **MongoDB Connector** gibi birçok seçenek sunar.
    c. Her bağlayıcı, veri kaynağından veri okuma ve yazma işlemlerini optimize eder.

Mimari Akış Şeması

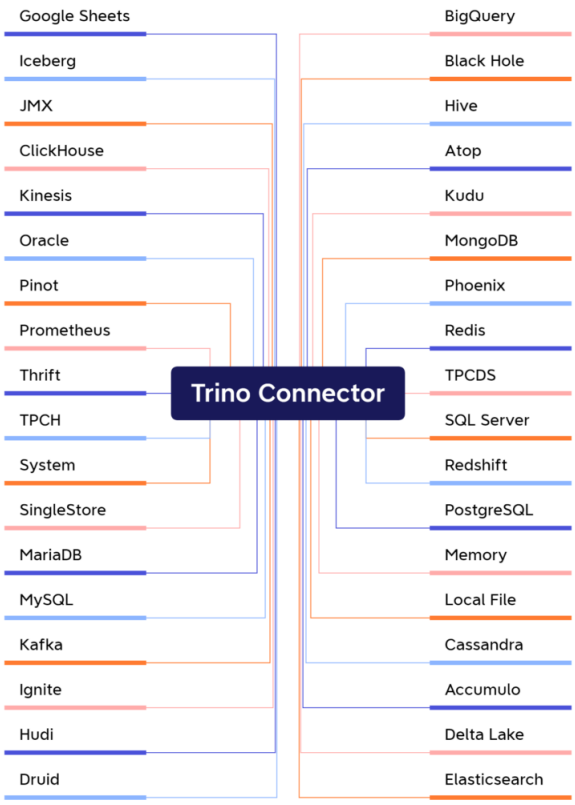
- Kullanıcı **Trino CLI**, **JDBC/ODBC**, veya **Web Arayüzü** üzerinden sorgu gönderir.
- **Koordinatör**, sorguyu alır ve **işlem planı** oluşturur.
- İş planı, **task** ve **split** olarak ikiye ayrılır. Task, belirli bir işlem adımını ifade eder; split ise veri parçalarını belirtir.

- İşçi nodelar, verilen görevleri yerine getirir ve sonuçları koordinatöre döner.
- Sonuçlar kullanıcıya iletilir.

3. Temel Kavramlar

Trino, ilişkisel veritabanları, anahtar-değer depoları ve nesne depolama gibi harici veri kaynaklarına SQL tabanlı erişim sağlar. Trino'da aşağıdaki kavramların anlaşılması önemlidir:

- **Konnektör (Connector):** Trino'yu bir veri kaynağına uyarlar. Her katalog belirli bir bağlayıcı ile ilişkilidir.



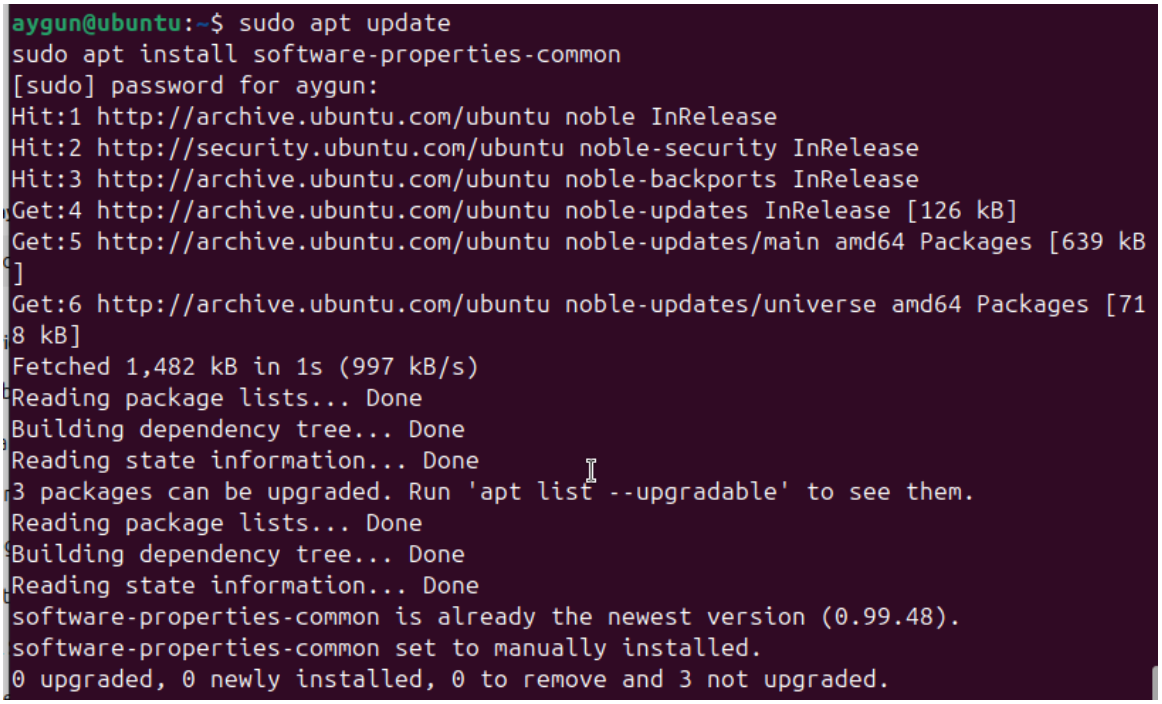
- **Katalog (Catalog):** Bir veri kaynağına erişmek için ayrıntıları tanımlar; şemaları içerir ve kullanılacak belirli bir bağlayıcıyı yapılandırır.

- **Şema (Schema):** Tabloları organize etmenin bir yolu. Katalog ve şema birlikte sorgulanabilecek bir tablo kümesi tanımlar.
- **Tablo (Table):** Veri türleriyle adlandırılmış sütunlar halinde düzenlenmiş sıralanmamış satırlar kümesi.

4. Apache Trino Kurulumu

Gereksinimler

- **Java 11 veya üzeri** kurulu olmalıdır.
- **Python 2.6.x**, **2.7.x**, veya **3.x** sürümlerinden biri de gereklidir.
- Trino'nun çalışacağı makineler için **ağ bağlantısı**, yeterli **RAM** ve **CPU kaynağı** gereklidir.

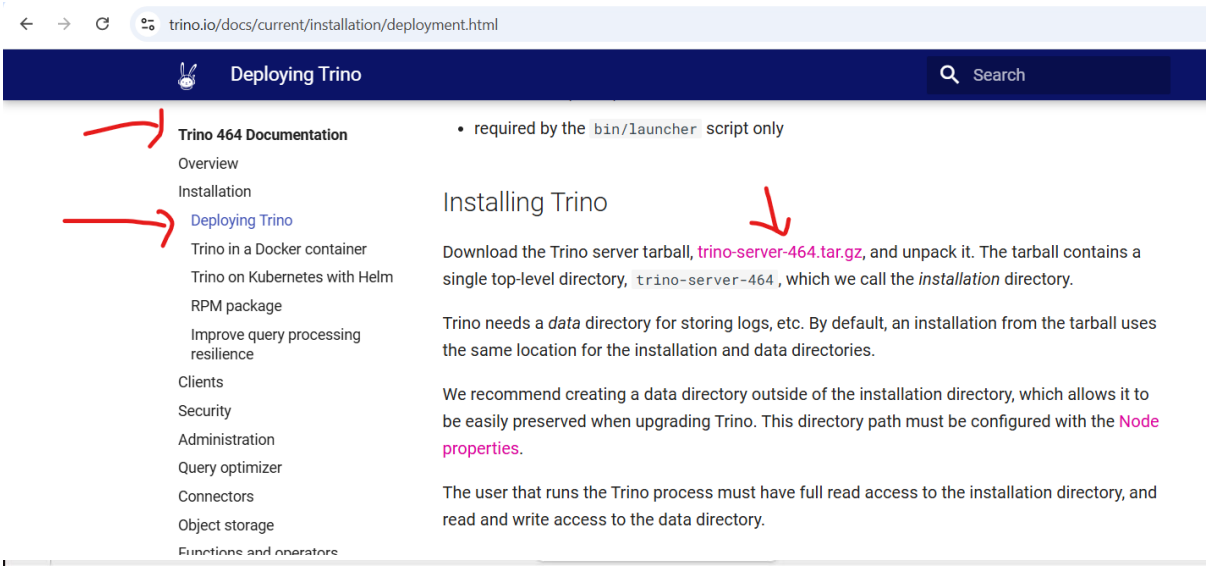


```
aygun@ubuntu:~$ sudo apt install openjdk-11-jdk
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is he
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock-frontend. It is he
ld by process 10261 (aptd)
```

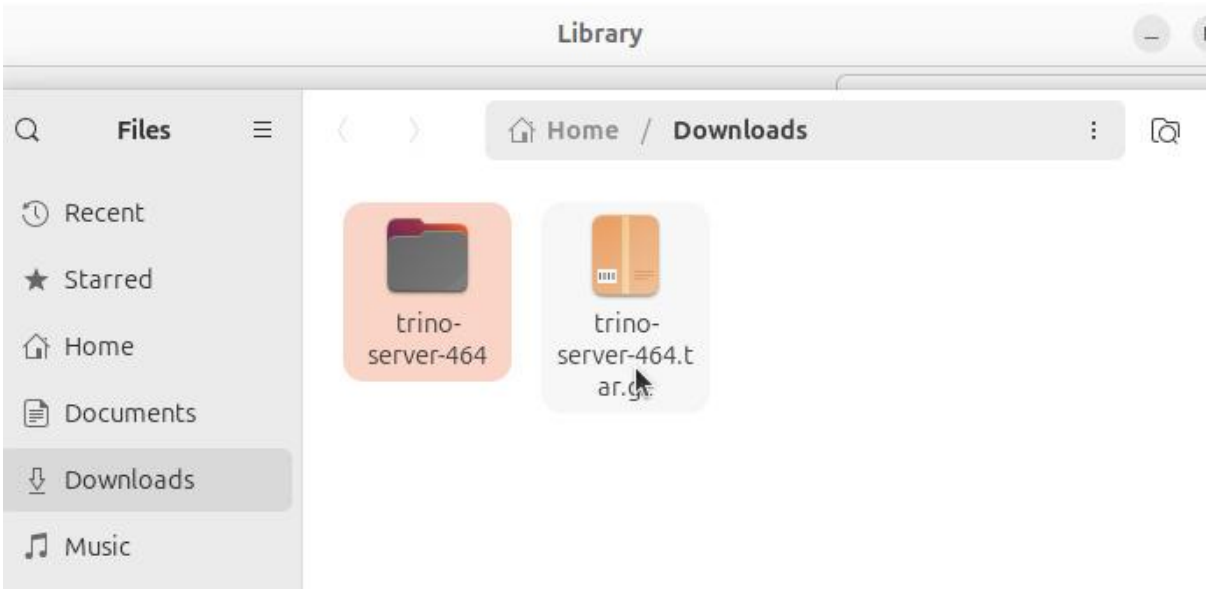
```
Setting up openjdk-11-jdk:amd64 (11.0.25+9-1ubuntu1~24.04) ...
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jconsole to pr
ovide /usr/bin/jconsole (jconsole) in auto mode
aygun@ubuntu:~$ java -version
openjdk version "11.0.25" 2024-10-15
OpenJDK Runtime Environment (build 11.0.25+9-post-Ubuntu-1ubuntu124.04)
OpenJDK 64-Bit Server VM (build 11.0.25+9-post-Ubuntu-1ubuntu124.04, mixed mode,
 sharing)
```

```
aygun@ubuntu:~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2
  dpkg-dev fakeroot g++ g++-13 g++-13-x86-64-linux-gnu g++-x86-64-linux-gnu
  gcc gcc-13 gcc-13-x86-64-linux-gnu gcc-x86-64-linux-gnu javascript-common
  libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan8 libbinutils libcc1-0 libctf-nobfd0 libctf0 libdpkg-perl
  libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-13-dev libgprofng0
  libhwasan0 libitm1 libjs-jquery libjs-sphinxdoc libjs-underscore liblsan0
  libpython3-dev libpython3.12-dev libquadmath0 libsframe1 libstdc++-13-dev
  libtsan2 libubsan1 lto-disabled-list make python3-dev python3-setuptools
  python3-wheel python3.12-dev zlib1g-dev
Suggested packages:
  binutils-doc gprofng-gui bzip2-doc debian-keyring g++-multilib
  g++-13-multilib gcc-13-doc gcc-multilib autoconf automake libtool flex bison
  gcc-doc gcc-13-multilib gcc-13-locales gdb-x86-64-linux-gnu apache2
```

```
aygun@ubuntu:~$ python3.11 --version
Python 3.11.10
```


trino.io/docs/current/installation/deployment.html
Deploying Trino
Search
Trino 464 Documentation
Overview
Installation
Deploying Trino
Trino in a Docker container
Trino on Kubernetes with Helm
RPM package
Improve query processing resilience
Clients
Security
Administration
Query optimizer
Connectors
Object storage
• required by the bin/launcher script only
Installing Trino
Download the Trino server tarball, trino-server-464.tar.gz, and unpack it. The tarball contains a single top-level directory, trino-server-464, which we call the installation directory.
Trino needs a data directory for storing logs, etc. By default, an installation from the tarball uses the same location for the installation and data directories.
We recommend creating a data directory outside of the installation directory, which allows it to be easily preserved when upgrading Trino. This directory path must be configured with the Node properties.
The user that runs the Trino process must have full read access to the installation directory, and read and write access to the data directory.


https://trino.io/docs/current/installation/deployment.html
Deploying Trino
trino-server-464.tar.gz
1m 26s left — 55.4 of 771 MB (10.1 MB/sec)
Show all downloads
Python
• version 2.6.x, 2.7.x, or 3.x
• required by the bin/launcher script only
Installing Trino
Download the Trino server tarball, trino-server-464.tar.gz, and unpack it. The tarball contains a single top-level directory, trino-server-464, which we call the installation directory.
Trino needs a data directory for storing logs, etc. By default, an installation from the tarball uses the same location for the installation and data directories.
Linux operating sys
Java runtime enviro
Python
Installing Trino
Configuring Trino
Node properties
JVM config
Temporary direct
Config properties


Library
Files
Home / Downloads
Recent
Starred
Home
Documents
Downloads
Music
trino-server-464
trino-server-464.tar.gz

5. Demo Uygulaması

Uygulamayı Docker kullanarak geliştireceğiz. Uygulamada kullandığımız kodlara buradan erişebilirsiniz. İlk olarak Trino'nun yapılandırma dosyalarını tanıyarak başlayalım.

Trino'yu başlatmadan önce, bir dizi yapılandırma dosyası sağlamamız gerekiyor:

- config.properties
- jvm.config
- node.properties

Varsayılan olarak, yapılandırma dosyalarının kurulum dizini içindeki etc dizininde olması beklenir. Bu üç dosyayı olmadığında Trino çalışmaz.

config.properties

```
coordinator=true
node-scheduler.include-coordinator=true
http-server.http.port=8080
discovery-server.enabled=true
discovery.uri=http://localhost:8080
```

Bu, trino kümesindeki her düğüm için birincil yapılandırma dosyasıdır. Burada ayarlanabilecek çok sayıda seçenek var demo uygulamasında varsayılan ayarları kullanarak devam edeceğiz.

- **coordinator=true|false**: Bu Trino örneğinin bir koordinatör olarak işlev görmesine izin verir. Varsayılan olarak true değer alır. Değerin olarak ayarlanması, false sunucuyu çalışan olarak ayırır.

- **node-scheduler.include-coordinator=true|false:** Koordinatör üzerinde çalışma planlamasına izin verir. Varsayılan olarak true değer alır.
- **http-server.http.port=8080 – http-server.https.port=8443:** HTTP/HTTPS bağlantısı için sunucu için kullanılan bağlantı noktalarını belirtir. Trino, tüm dahili ve harici iletişim için HTTP kullanır.
- **discovery-server.enabled=true:** Trino, kümedeki tüm düğümleri bulmak için keşif hizmetini kullanır. Her Trino örneği, başlangıçta keşif hizmetine kaydolur.
- **discovery.uri=http://localhost:8080:** Keşif sunucusunun URI'si. Trino koordinatöründe gömülü keşif sürümünü çalıştırırken, Trino koordinatörünün URI'si olmalıdır.

jvm.config

Trino, Java tabanlı bir uygulamadır. Bu dosya ise Trino'yu çalıştıran Java Sanal Makinesinin (JVM) bellek ayarlarını tutan bir yapılandırma dosyasıdır.

```
-server
-Xmx16G
-XX:-UseBiasedLocking
-XX:+UseG1GC
-XX:G1HeapRegionSize=32M
-XX:+ExplicitGCInvokesConcurrent
-XX:+HeapDumpOnOutOfMemoryError
-XX:+ExitOnOutOfMemoryError
-XX:ReservedCodeCacheSize=512M
-XX:PerMethodRecompilationCutoff=10000
-XX:PerBytecodeRecompilationCutoff=10000
-Djdk.nio.maxCachedBufferSize=2000000
-Djdk.attach.allowAttachSelf=true
```

node.properties

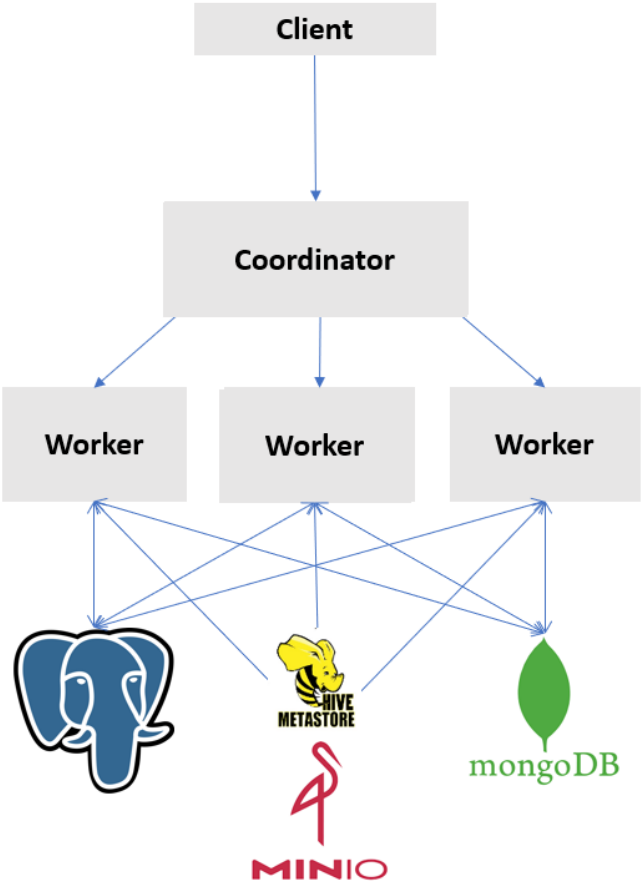
Bu yapılandırma, kümedeki düğümleri benzersiz bir şekilde tanımlamak ve düğümdeki dizinlerin konumlarını belirtmek için kullanılır.

- **node.environment=demo:** Ortamın gerekli adı . Bir kümedeki tüm Trino düğümleri aynı ortam adına sahip olmalıdır. Burada belirtilen ad, Trino Web UI başlığında görünür.
- **node.id=some-random-unique-string:** Bu Trino kurulumu için isteğe bağlı benzersiz bir tanımlayıcı. Bu, her düğüm için benzersiz olmalıdır. Bu tanımlayıcı, Trino'nun yeniden başlatılması veya yükseltilmesi boyunca tutarlı kalmalı ve bu nedenle belirtilmelidir. Atlanırsa, her yeniden başlatmada rastgele bir tanımlayıcı oluşturulur.
- **node.data-dir=/var/trino/data:** Trino'nun günlük dosyalarını ve diğer verileri depoladığı dizinin isteğe bağlı dosya sistemi yolu. Varsayılan olarak kurulum dizini içindeki var klasörüdür.

Repo üzerinde iki uygulama örneği bulunuyor.

- Single Node
- Multi Node

Ben anlatmaya multi-node üzerinden devam edeceğim. Konfigürasyon işlemine başlayalım.



Konfigürasyon

Koordinatör ve işçi sunucuların konfigürasyon ayarlarında farklılık olduğu için iki ayrı klasör oluşturuyoruz. Buradaki farklılık yalnız config properties dosyası içerisinde.

```
|   docker-compose-multinode.yaml
|   README.md
|
\---etc
    +---coordinator
    |   |   config.properties
    |   |   jvm.config
    |   |   log.properties
    |   |   node.properties
    |   |
    |   \---catalog
    |           tpcds.properties
    |           tpch.properties
    |
    \---worker
        |   config.properties
        |   jvm.config
        |   log.properties
        |   node.properties
        |
        \---catalog
                tpcds.properties
                tpch.properties
```

coordinator/config.properties

```
coordinator=true
node-scheduler.include-coordinator=false
http-server.http.port=8080
discovery.uri=http://coordinator:8080
```

worker/config.properties

NOT: Bir Trino cluserındaki tüm işçi konfigürasyonları aynı olmalıdır.

```
coordinator=false
http-server.http.port=8080
discovery.uri=http://coordinator:8080
```

node.properties, jvm.config ve log.properties dosyaları içerikleri aynı her iki sunucu içinde aynı.

node.properties

```
node.environment=docker
node.data-dir=/data/trino
plugin.dir=/usr/lib/trino/plugin
```

jvm.config:

```
-server
-Xms2G
-Xmx2G
-XX:InitialRAMPercentage=80
-XX:MaxRAMPercentage=80
-XX:G1HeapRegionSize=32M
-XX:+ExplicitGCInvokesConcurrent
-XX:+ExitOnOutOfMemoryError
-XX:+HeapDumpOnOutOfMemoryError
-XX:-OmitStackTraceInFastThrow
-XX:ReservedCodeCacheSize=512M
-XX:PerMethodRecompilationCutoff=10000
-XX:PerBytecodeRecompilationCutoff=10000
-Djdk.attach.allowAttachSelf=true
-Djdk.nio.maxCachedBufferSize=2000000
-XX:+UnlockDiagnosticVMOptions
-XX:+UseAESCTRIntrinsics
```

Veri kaynaklarına bağlantı sağlayabilmek için etc/catalog dizinine konnektör dosyalarımızı eklememiz gerekiyor. Konnektör dosyaları bir veritabanına bağlanmak için gerekli olan ip, port, kullanıcı adı, şifre vs. gibi bilgileri içerir.

- **mongo.properties**

```
connector.name=mongodb
mongodb.connection-url=mongodb://admin:admin@mongodb:27017/
```

- **mysql.properties**

```
connector.name=mysql
connection-url=jdbc:mysql://mysql:3306
connection-user=admin
connection-password=admin
```

- **minio.properties**

Minio bağlantısı için aynı zamanda Hive Metastore için konfigürasyon ayrı yapılması gerekiyor. Yapılandırma dosyası xml formatında. Yine bu dosya içerisinde sunucu bilgileri, kullancı adı, şifre vb. gibi bilgileri bulunmakta bu bilgileri kendi bilgilerimize uyarlamamız gerekiyor. Örnek konfigürasyon dosyayına repo üzerinden erişebilirsiniz.

```
connector.name=hive
hive.metastore.uri=thrift://hive-metastore:9083
hive.s3.aws-access-key=trainkey
hive.s3.aws-secret-key=trainsecret
hive.s3.endpoint=http://minio:9000
hive.s3.path-style-access=true
```

Docker ile Trino Kurulumu

Hızlıca kullandığımız servisleri inceleyelim ve uygulamamızı çalıştıralım.

```
version: '3.7'
services:
coordinator:
container_name: mn-coordinator
image: 'trinodb/trino:408'
ports:
- "9091:8080"
volumes:
- ./etc/coordinator:/etc/trino
networks:
- trino-multinode-network
worker1:
container_name: mn-worker1
image: 'trinodb/trino:408'
ports:
- "8081:8081"
volumes:
- ./etc/worker:/etc/trino
networks:
- trino-multinode-network
worker2:
container_name: mn-worker2
image: 'trinodb/trino:408'
ports:
- "8082:8081"
volumes:
- ./etc/worker:/etc/trino
networks:
- trino-multinode-network
worker3:
container_name: mn-worker3
image: 'trinodb/trino:408'
ports:
- "8083:8081"
volumes:
- ./etc/worker:/etc/trino
networks:
- trino-multinode-network
```

```
mysql:
container_name: mn-mysql
image: mysql:latest
hostname: mysql
environment:
MYSQL_ROOT_PASSWORD: admin
MYSQL_USER: admin
MYSQL_PASSWORD: admin
ports:
- '3307:3306'
networks:
- trino-multinode-network
mongodb:
container_name: mn-mongo
image: 'mongo:latest'
hostname: mongodb
ports:
- '27018:27017'
environment:
MONGO_INITDB_ROOT_USERNAME: admin
MONGO_INITDB_ROOT_PASSWORD: admin
networks:
- trino-multinode-network
mariadb:
container_name: mariadb
hostname: mariadb
image: mariadb:10.5.8
ports:
- 3308:3306
environment:
MYSQL_ROOT_PASSWORD: admin
MYSQL_USER: admin
MYSQL_PASSWORD: admin
MYSQL_DATABASE: metastore_db
networks:
- trino-multinode-network
hive-metastore:
hostname: hive-metastore
image: 'bitsondatadev/hive-metastore:latest'
ports:
- '9083:9083' # Metastore Thrift
volumes:
- ./conf/metastore-site.xml:/opt/apache-hive-metastore-3.0.0-bin/conf/metastore-
site.xml:ro
environment:
METASTORE_DB_HOSTNAME: mariadb
depends_on:
- mariadb
networks:
- trino-multinode-network
minio:
restart: always
image: minio/minio:RELEASE.2024-11-27T18-10-45Z
```

```
container_name: minio
hostname: minio
ports:
- "9000:9000"
- "9001:9001"
networks:
- trino-multinode-network
command: server /data --console-address ':9001' --address ':9000'
env_file:
- .env
environment:
- MINIO_ROOT_USER=${AWS_ACCESS_KEY_ID}
- MINIO_ROOT_PASSWORD=${AWS_SECRET_ACCESS_KEY}
volumes:
- ./data:/data
volumes:
minio-data:
driver: local
networks:
trino-multinode-network:
driver: bridge
```

- coordinator: Trino'nun çalıştığı ana konteyner
- İşçi sunucular
    - worker1
    - worker2
    - worker3
- Bağlantı yapacağımız veri tabanları
    - mysql
    - mongodb
    - minio

Evet ayarlamaları yaptık. Servisleri de tanıdığımıza göre artık çalıştırabiliriz.

```
docker-compose -f docker-compose-multinode.yaml up --build -d
```

Konteynerların durmunu kontrol edelim.

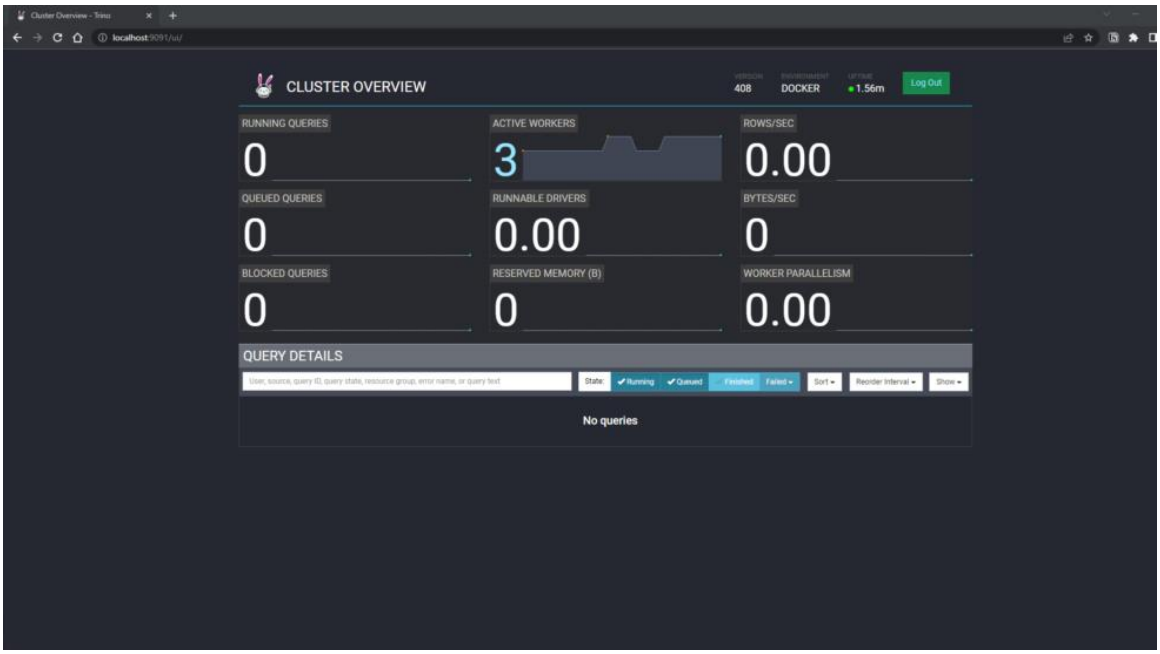
```
docker-compose -f docker-compose-multinode.yaml ps
```

Eğer sorun yoksa aşağıdaki şekilde bir çıktı alacağız. 'STATUS' kısmında hepsinin başladığını görüyoruz.

```
NAME                          COMMAND                  SERVICE         STATUS              PORTS
mariadb                       "docker-entrypoint.s…"   mariadb         running             0.0.0.0:3308->3306/tcp
minio                         "/usr/bin/docker-ent…"   minio           running             0.0.0.0:9000-9001->9000-9001/tcp
mn-coordinator                "/usr/lib/trino/bin/…"   coordinator     running (starting)  0.0.0.0:9091->8080/tcp
mn-mongo                      "docker-entrypoint.s…"   mongodb         running             0.0.0.0:27018->27017/tcp
mn-mysql                      "docker-entrypoint.s…"   mysql           running             33060/tcp, 0.0.0.0:3307->3306/tcp
mn-worker1                    "/usr/lib/trino/bin/…"   worker1         running (starting)  8080/tcp, 0.0.0.0:8081->8081/tcp
mn-worker2                    "/usr/lib/trino/bin/…"   worker2         running (starting)  8080/tcp, 0.0.0.0:8082->8081/tcp
mn-worker3                    "/usr/lib/trino/bin/…"   worker3         running (starting)  8080/tcp, 0.0.0.0:8083->8081/tcp
multinode-hive-metastore-1    "sh -c /entrypoint.sh"   hive-metastore  running             0.0.0.0:9083->9083/tcp
```

-

Şimdi Trino arayüzüne gidelim. Arayüzün gelmesi biraz zaman alabiliyor. *Her Trino sunucusu, genellikle Trino Web Kullanıcı Arayüzü* olarak adlandırılan bir web arayüzü sağlar . Trino Web Kullanıcı Arayüzü, Trino sunucusu ve sunucudaki sorgu işleme hakkındaki ayrıntıları gösterir. Bir kullanıcı adı ile oturum açmanız gerekmektedir. Varsayılan olarak, hiçbir kimlik doğrulaması yapılandırılmadığı için ‘trino’ kullanıcı adı giriş yapabilirsiniz. Normalde Trino Web Arayüzü 8080 portunda çalışmaktadır. Ancak doker-compose dosyasında 9091 portuna yönlendirme yaptığımız için http://localhost:9091/ui/ üzerinde arayüze erişebiliriz.

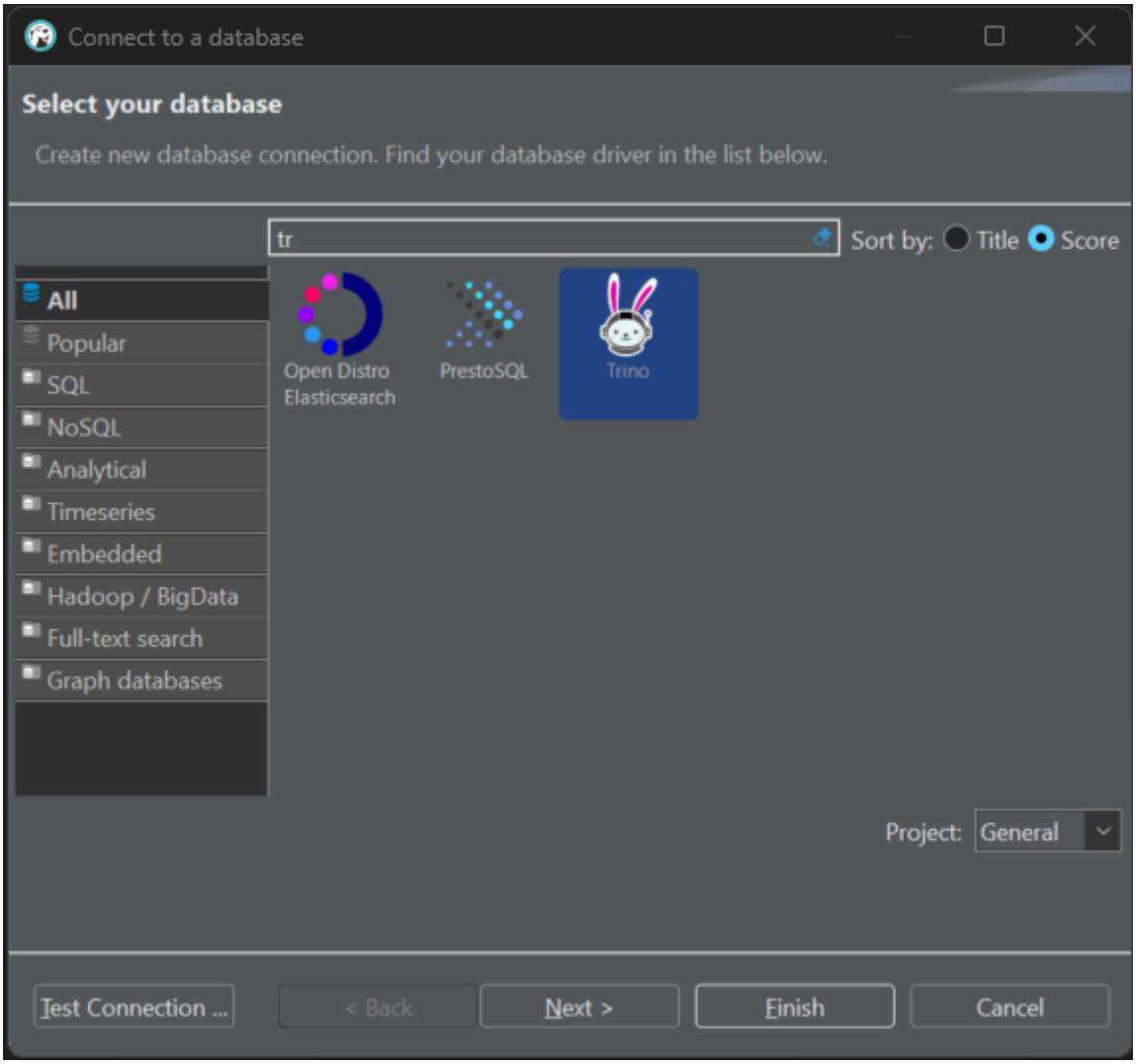


DBeaver SQL editörüyle Trino Bağlantısı

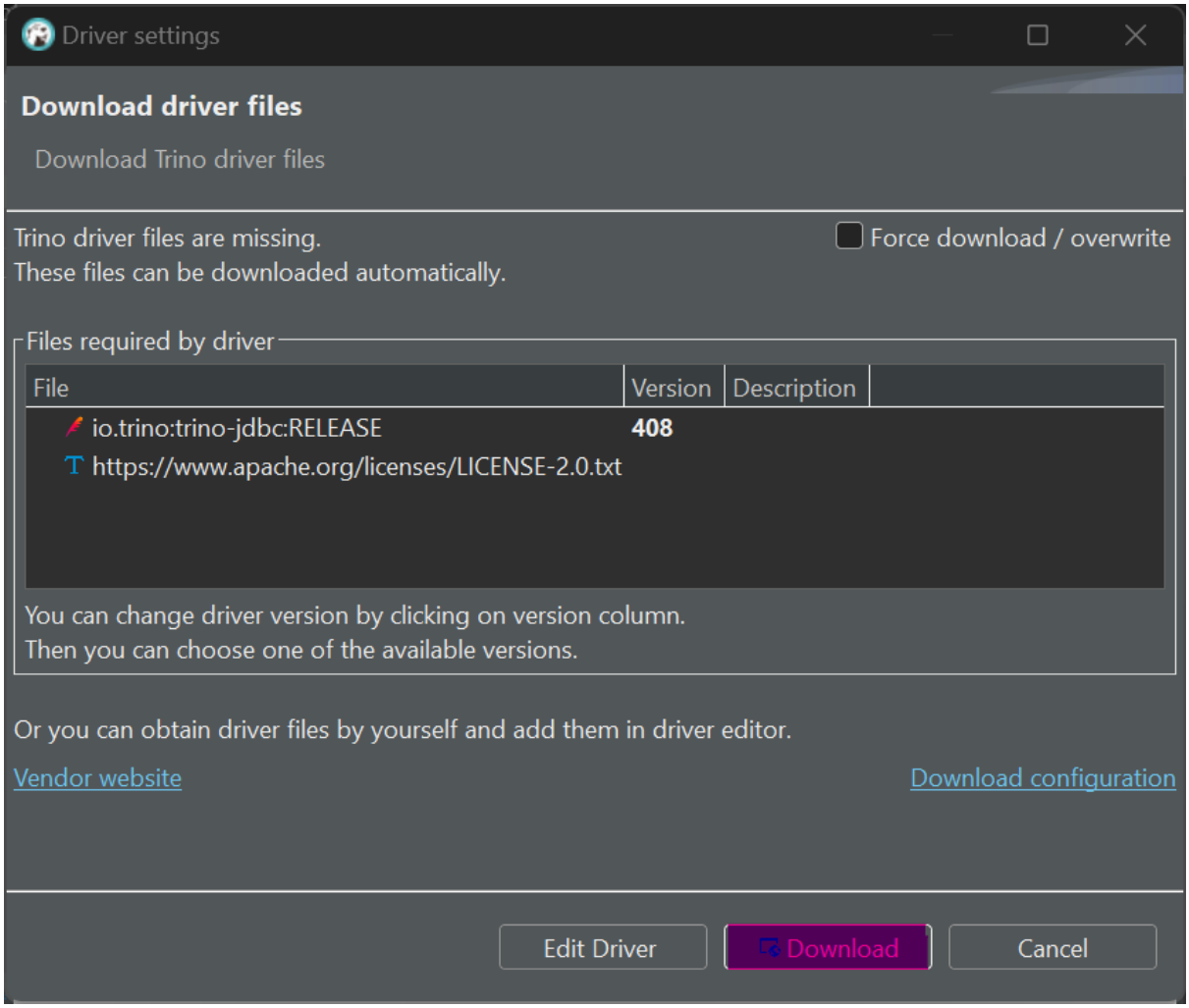
DBeaver SQL editörüyle Trino'ya bağlantı kurup sorgularımızı çalıştırabiliriz.

Bağlantı kurmak için aşağıdaki adımları takip edebilirsiniz.

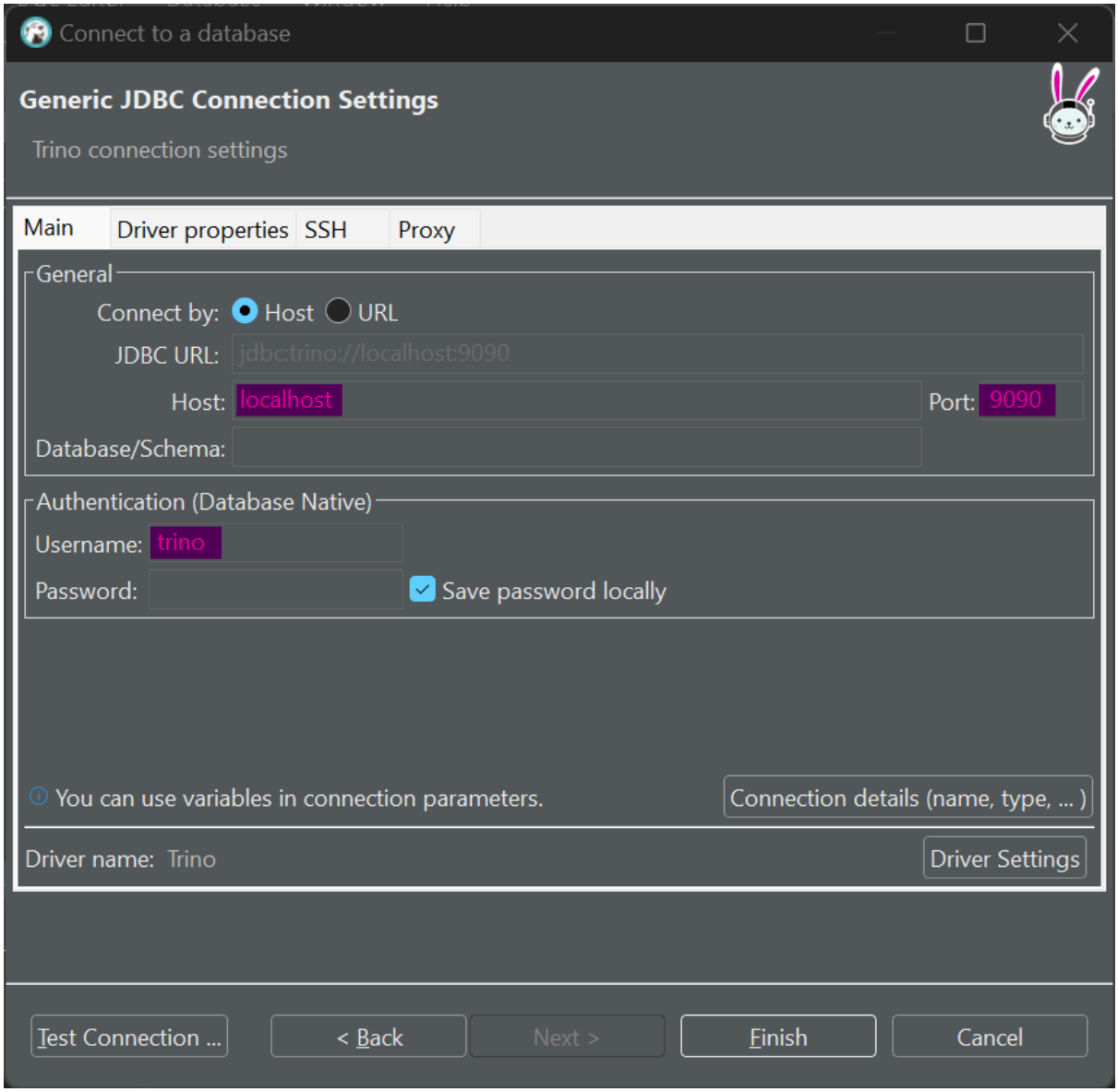
Adım-1: Veri tabanı seçimi



Adım-2: Gerekli olan sürücüleri indirme



Adım-3: Trino bağlantısı



Trino ile Sorgulama

Bağlantıyı sağladıktan sonra bir oturum açalım ve örnek sorgulama yapalım. Repoda paylaştığım multinode.sql içerisinde farklı sorgular mecvut. Yazıyı daha fazla uzatmamak için sadece Mysql ve Mongodb üzerinden join ile birleştip okuduğum veriyi Minio'ya ORC formatında nasıl kayıt edileceğini göstereceğim.

Minio üzerinde şema oluşturalım

Note: Kodlar çalıştırılmadan önce Minio üzerinde vbo isminde bucket oluşturulması gerekmektedir. Aksi halde hata oluşacaktır.

```
CREATE SCHEMA minio.vbo WITH (location = 's3a://vbo/')
```

Hangi şema üzerinde çalışacağımız belirtelim

```
USE minio.vbo;
```

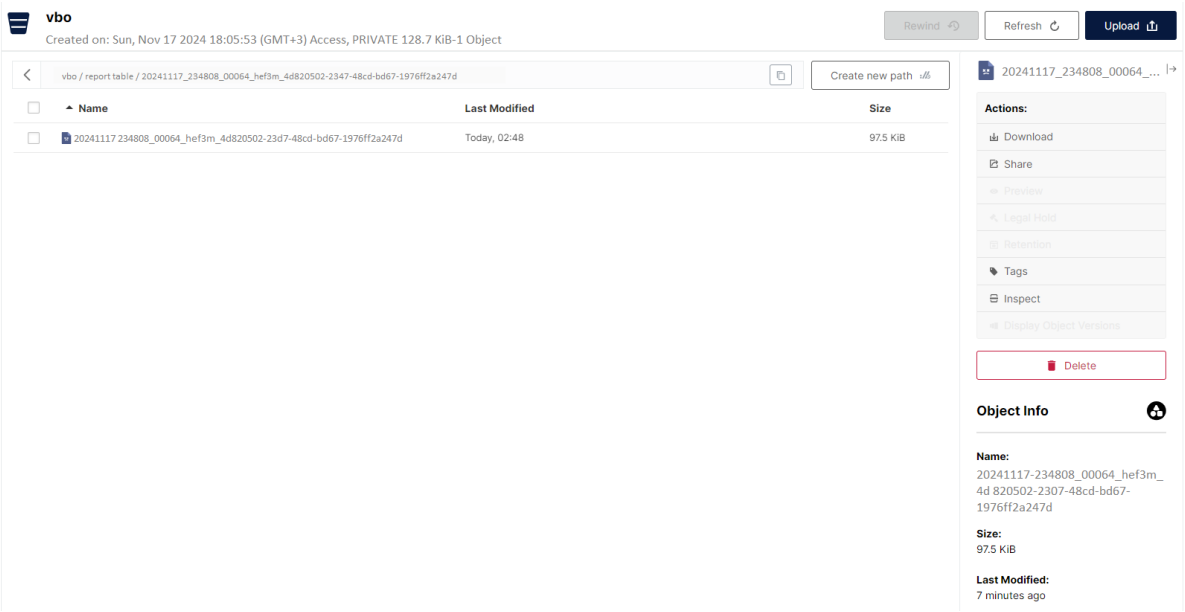
Aşağıdaki sorguda Mysql ve Mongo veri tabanlarına Trino'nun test için sunduğu veri katalogundan veri aktarıp daha sonra kayıt ettiğimiz tabloları birleştip Minio'ya ORC formatın da yazıyoruz.

```
create schema mysql.tiny;
create table mysql.tiny.nation AS SELECT * FROM tpch.tiny.nation;
create table mysql.tiny.region AS SELECT * FROM tpch.tiny.region ;
create schema mongo.tiny;
CREATE TABLE mongo.tiny.customer AS SELECT * FROM tpch.tiny.customer;
CREATE TABLE mongo.tiny.orders AS SELECT * FROM tpch.tiny.orders ;
create table minio.vbo.report_table WITH (format = 'ORC') as
select c.name as customer, o.totalprice, o.orderdate, n.name as nation, r.name as
region
from mongo.tiny.customer as c
inner join mongo.tiny.orders as o on o.custkey = c.custkey inner join
mysql.tiny.nation as n on n.nationkey = c.nationkey
inner join mysql.tiny.region as r on r.regionkey = n.regionkey
```

Son olarak kayıt ettiğimiz ORC formatındaki dosyayı Minio arayüzünden kontrol edelim. Bağlanmak için http://localhost:9001 adresine gidelim.

- Kullanıcı adı: trainkey
- Şifre: trainsecret

Aşağıdaki görselde sorgu sonucunun doysa olarak kayıt edildiğini görebiliriz.



Özetle, Trino, büyük veri setlerinde veri sorgulama işlemlerini hızlandırmak için kullanılan açık kaynaklı bir veri sorgulama motorudur. Bir veri tabanı değildir. Farklı veritabanlarına aynı anda bağlantı sağlayıp tek bir SQL syntax'i ile sorugulama yapmamız sağlayan bir araçtır. Aynı zamanda kullanıcı yetkilerinde bir sınırlama yoksa veriyi aktarımı içinde kullanabileceğimizi demo ile birlikte gözlemlemiş olduk.

Kaynaklar

- Trino 408 Documentation
- 4. Trino Architecture – Trino: The Definitive Guide [Book]
- bitsondatadev/trino-getting-started
- GitHub – Lewuathe/docker-trino-cluster: Multiple node presto cluster on docker container
- Presto and Minio on Docker Infrastructure