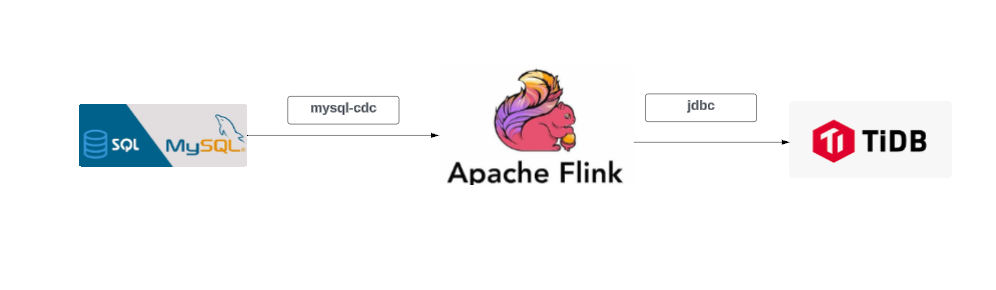
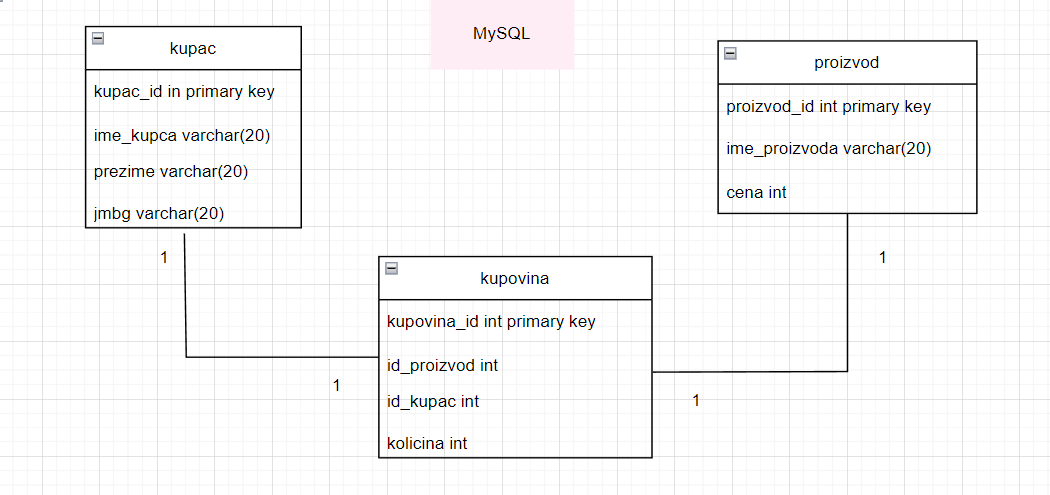
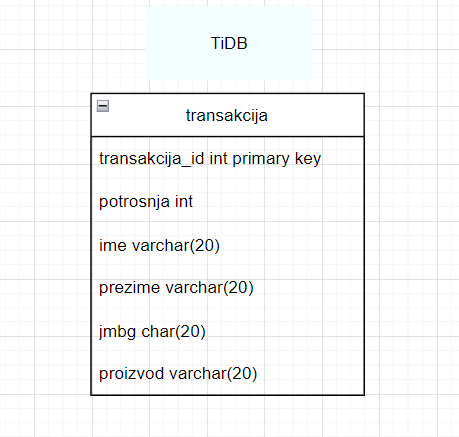
MySQL -> Flink -> TiDB







1. Pokrenem Docker
2. Pokrenem kontejner flink-tidb-rdw-master

Kontejner sadrzi sledece :

* tidb
* tikv-1
* taskmanager-1
* mysql-1
* pd-1
* jobmanager-1

(tidb, tikv-1 i pd-1 vezuju se za tidb. taskmanager-1 I jobmanager-1 slike vezuju se za flink.)

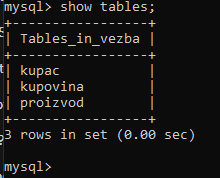
Za taskmanager I jobmanager vidimo da su slike flink:1.12.2-scala\_2.12.

1. Pozicioniramo se u folderu gde se nalazi .yaml fajl I pokrenemo cmd.

Pokrenemo mysql

Sa use vezba; izaberemo bazu

Kreiramo sledece tabele Ili ukoliko ih vec imamo kreirane sa show tables; mozemo ih prikazati.



docker-compose exec mysql mysql -uroot

DROP DATABASE IF EXISTS vezba;

CREATE DATABASE vezba;

USE vezba;

create table kupac (

kupac\_id int primary key,

ime\_kupca varchar(20),

prezime varchar(20),

jmbg varchar(13)

);

create table proizvod(

proizvod\_id int primary key,

ime\_proizvoda varchar(20),

cena int

);

create table kupovina(

kupovina\_id int primary key,

id\_proizvod int,

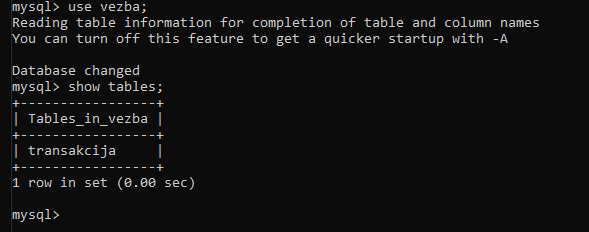
id\_kupac int,

kolicina int

);

1. Pokrenemo novi cmd. Pokrenemo sledecu komandu.

docker-compose exec mysql mysql –h tidb –u root –P 4000



docker-compose exec mysql mysql -htidb -uroot -P4000

DROP DATABASE IF EXISTS vezba;

CREATE DATABASE vezba;

USE vezba;

create table transakcija(

transakcija\_id int primary key,

potrosnja int,

ime varchar(20),

prezime varchar(20),

jmbg varchar(13),

proizvod varchar(20)

);

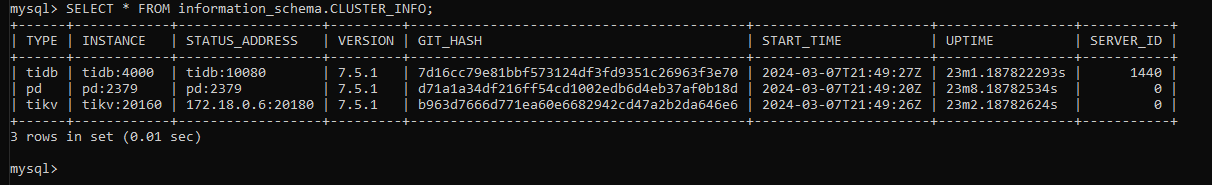
Ovei upiti daju informacije o postojanju pojedinih klastera. Konkretno u ovom slucaju postoji samo tikv.

SELECT \* FROM information\_schema.CLUSTER\_INFO WHERE type = 'ticdc';

SELECT \* FROM information\_schema.CLUSTER\_INFO WHERE type = 'tiup';

SELECT \* FROM information\_schema.CLUSTER\_INFO WHERE type = 'tikv';

Ili komandu mozemo napisati ovako:



1. Flink SQL Client.

docker-compose exec jobmanager ./bin/sql-client.sh embedded -l ./connector-lib

docker-compose exec jobmanager: Ovaj deo komande koristi docker-compose exec komandu da bi se pokrenula komanda unutar kontejnera koji je definisan kao jobmanager servis u Docker Compose fajlu.

Obrati paznju na to kao su u komandi back-slash(/) postavljene.

Kreiram tabele, neke su sa connector-om mysql-cdc, a neke sa jdbc.

create table kupac (

kupac\_id int primary key,

ime\_kupca varchar(20),

prezime varchar(20),

jmbg varchar(13)

) WITH (

'connector' = 'mysql-cdc',

'hostname' = 'mysql',

'port' = '3306',

'username' = 'root',

'password' = '',

'database-name' = 'vezba',

'table-name' = 'kupac'

);

create table proizvod(

proizvod\_id int primary key,

ime\_proizvoda varchar(20),

cena int

) WITH (

'connector' = 'mysql-cdc',

'hostname' = 'mysql',

'port' = '3306',

'username' = 'root',

'password' = '',

'database-name' = 'vezba',

'table-name' = 'proizvod'

);

create table kupovina(

kupovina\_id int primary key,

id\_proizvod int,

id\_kupac int,

kolicina int

) WITH (

'connector' = 'mysql-cdc',

'hostname' = 'mysql',

'port' = '3306',

'username' = 'root',

'password' = '',

'database-name' = 'vezba',

'table-name' = 'kupovina'

);

create table transakcija(

transakcija\_id int primary key,

potrosnja int,

ime varchar(20),

prezime varchar(20),

jmbg varchar(13),

proizvod varchar(20)

) WITH (

'connector' = 'jdbc',

'driver' = 'com.mysql.cj.jdbc.Driver',

'url' = 'jdbc:mysql://tidb:4000/vezba?rewriteBatchedStatements=true',

'table-name' = 'transakcija',

'username' = 'root',

'password' = ''

);

create table print\_kupac WITH ('connector' = 'print') LIKE kupac (EXCLUDING ALL);

create table print\_proizvod WITH ('connector' = 'print') LIKE proizvod (EXCLUDING ALL);

create table print\_kupovina WITH ('connector' = 'print') LIKE kupovina (EXCLUDING ALL);

create table print\_transakcija WITH ('connector' = 'print') LIKE transakcija (EXCLUDING ALL);

set 'sql-client.execution.result-mode' = 'changelog';

Nakon kreiranja tabela cini su konektori jdbc I mysql-cdc, kreiramo I tabela ciji je konekor print.Nakon toga kreiramo I sql upite koji predstavljaju job-ove;

insert into transakcija (transakcija\_id, potrosnja, ime, prezime, jmbg, proizvod)

select kupovina.kupovina\_id, kupovina.kolicina \* proizvod.cena as potrosnja, kupac.ime\_kupca, kupac.prezime, kupac.jmbg, proizvod.ime\_proizvoda

from kupovina

inner join kupac on kupac.kupac\_id = kupovina.id\_kupac

inner join proizvod on proizvod.proizvod\_id = kupovina.id\_proizvod;

insert into print\_kupac select \* from kupac;

insert into print\_proizvod select \* from proizvod;

insert into print\_kupovina select \* from kupovina;

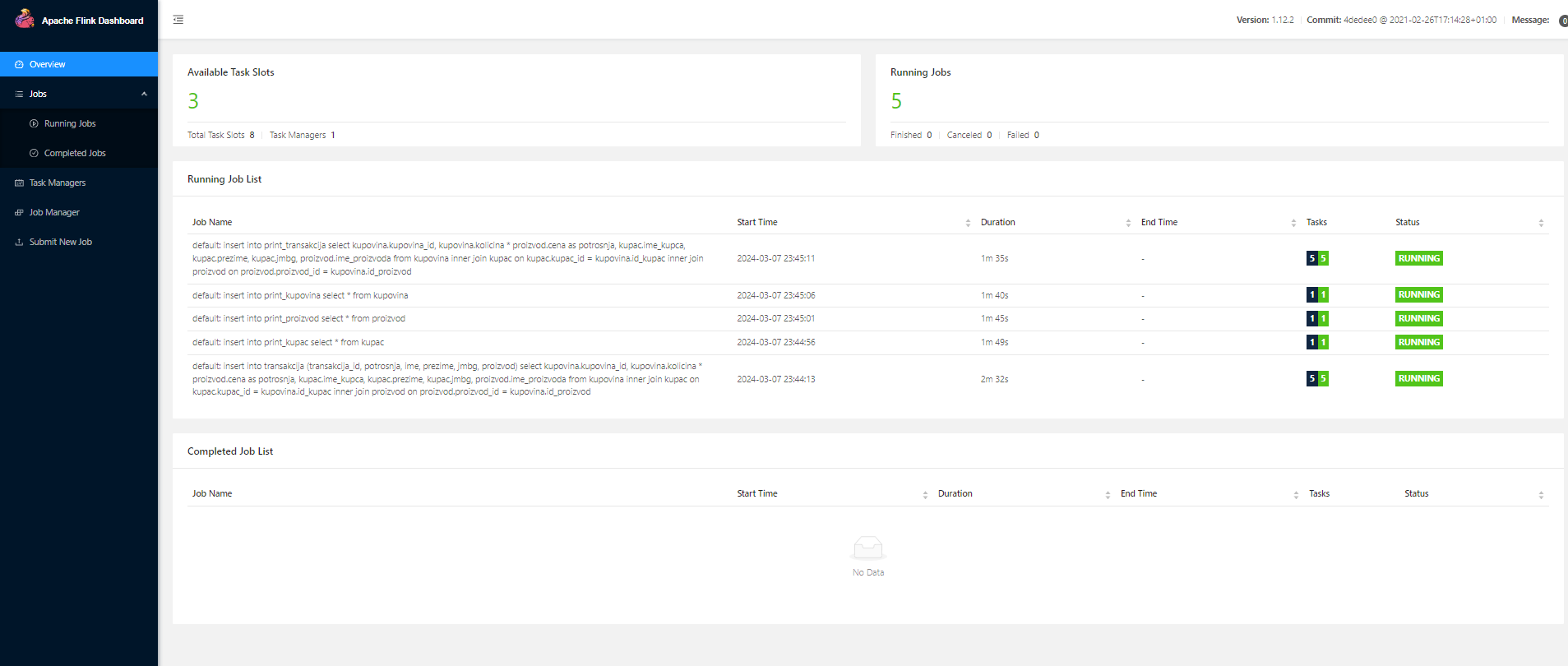
insert into print\_transakcija

select kupovina.kupovina\_id, kupovina.kolicina \* proizvod.cena as potrosnja, kupac.ime\_kupca, kupac.prezime, kupac.jmbg, proizvod.ime\_proizvoda

from kupovina

inner join kupac on kupac.kupac\_id = kupovina.id\_kupac

inner join proizvod on proizvod.proizvod\_id = kupovina.id\_proizvod;

job-ove mozemo da pratimo u taskmanager app

1. U MySQL unosimo sledece podatke

docker-compose exec mysql mysql –u root

use vezba;

insert into kupac values (1, 'Vida', 'Arsic','2806998979813');

insert into kupac values (2, 'Vuk', 'Arsic','1303200212345');

insert into kupac values (3, 'Nemanja', 'Arsic','1234567891234');

insert into kupac values (4, 'Nebojsa', 'Arsic','0107567891234');

insert into proizvod values (1, 'Parfem', 10000);

insert into proizvod values (2, 'Puder', 4000);

insert into proizvod values (3, 'Maskara', 2000);

insert into kupovina values (1, 1, 1, 2);

insert into kupovina values (2, 2, 3, 3);

insert into kupovina values (3, 3, 2, 1);

insert into kupovina values (4, 3, 2, 1);

insert into kupovina values (5, 3, 2, 1);

insert into kupovina values (6, 3, 2, 1);

insert into kupovina values (7, 1, 1, 2);

insert into kupovina values (8, 2, 3, 3);

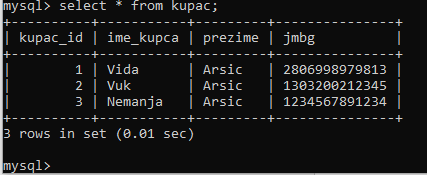
insert into kupovina values (9, 3, 2, 1);

insert into kupovina values (10, 3, 2, 1);

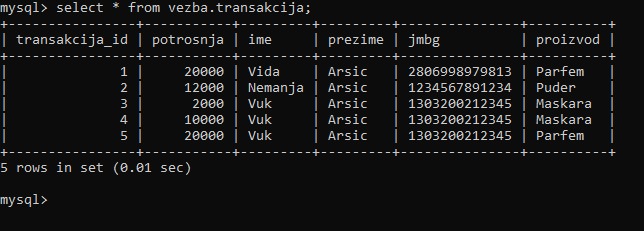
insert into kupovina values (11, 3, 2, 1);

insert into kupovina values (12, 3, 2, 1);

kad podatke unesemo, mozemo da ih prikazemo



docker-compose exec mysql mysql –h tidb –u root –P 4000 -e"select \* from vezba.transakcija";



https://dev.to/yared123yared/introduction-to-pingcap-tidb-part-3-building-a-net-core-web-api-and-connect-to-tidb-mysql-server-4e35

\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\

CREATE TABLE cena (

id INT primary key,

granica int

) WITH (

'connector' = 'filesystem',

'path' = 'file:///C:/Users/arsic/Downloads/flink-tidb-rdw-master/meta/meta.csv',

'format' = 'csv'

);

insert into cena values (1, 10000);

KAFKA

bin\zookeeper-server-start.sh config\zookeeper.properties

bin\kafka-server-start.sh config\server.properties

bin\kafka-topics.sh --create --topic quickstart-events --bootstrap-server localhost:9092

bin\kafka-topics.sh --describe --topic quickstart-events --bootstrap-server localhost:9092

**Step 2: configuring the .Net web API project**

After we set up our MySQL server well the next step will be configuring the .Net core web API for connecting with TiDB MySQL.

Create a new .Net web API project by using this command.

dotnet new webapi -n <name>

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System;

namespace fixit.DTO

{

    public class TransakcijaDto

    {

        [Required]

        public int Transakcija\_id { get; set; }

        [Required]

        public int Potrosnja { get; set; }

        [Required]

        public string Ime { get; set; }

        [Required]

        public string Prezime { get; set; }

        [Required]

        public string Jmbg { get; set; }

        [Required]

        public string Proizvod { get; set; }

    }

}