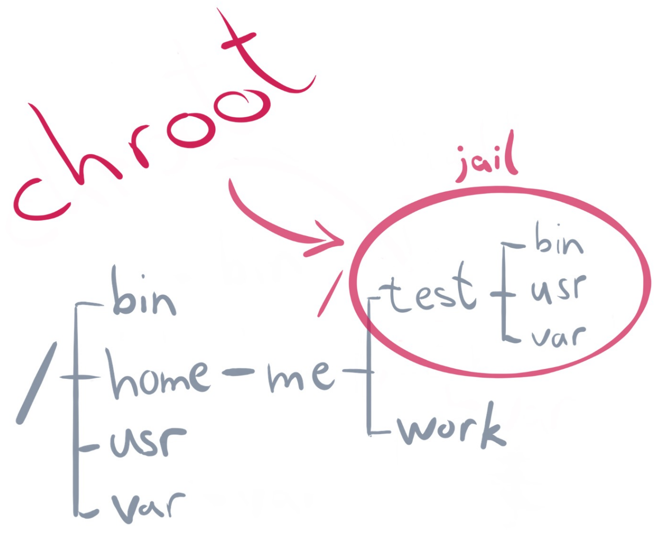
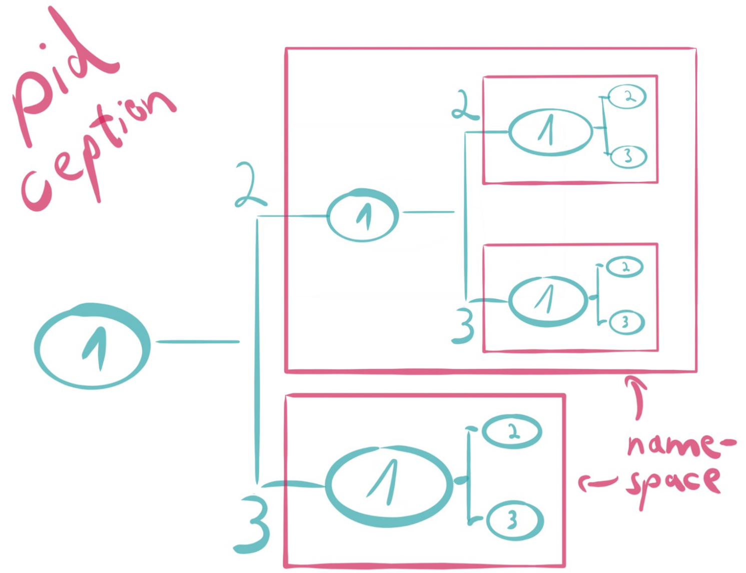
Урок 6. Деплой приложения

Docker container

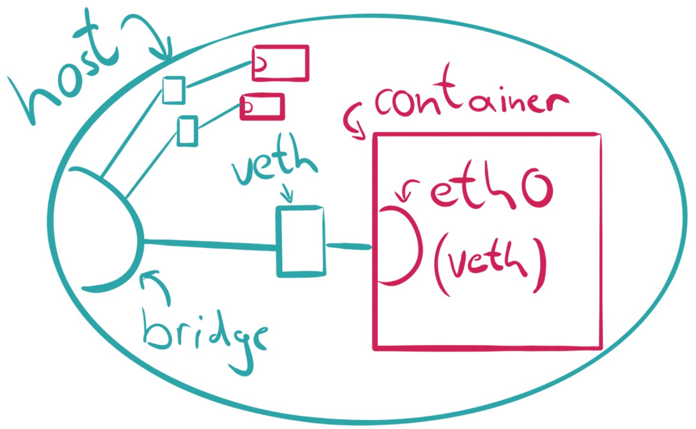
Containers are only **isolated groups of processes running on a single host**, which fulfill a set of “common” features.



Process ID (pid)

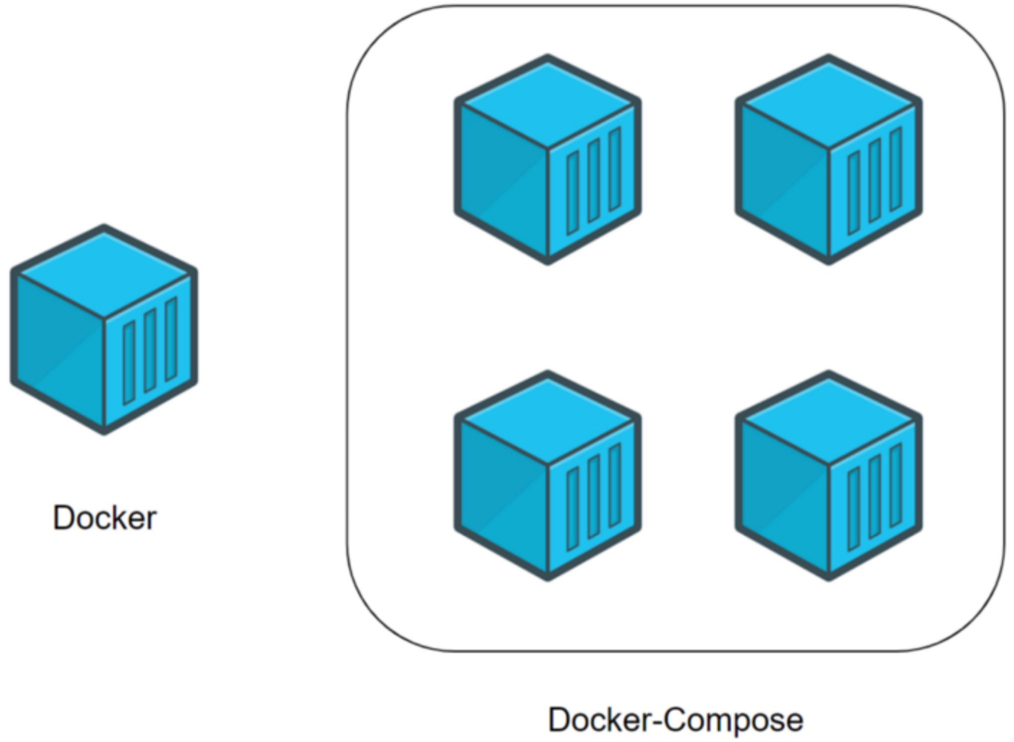


Network (net)



https://medium.com/@saschagrunert/demystifying-containers-part-i-kernel-space-2c53d6979504

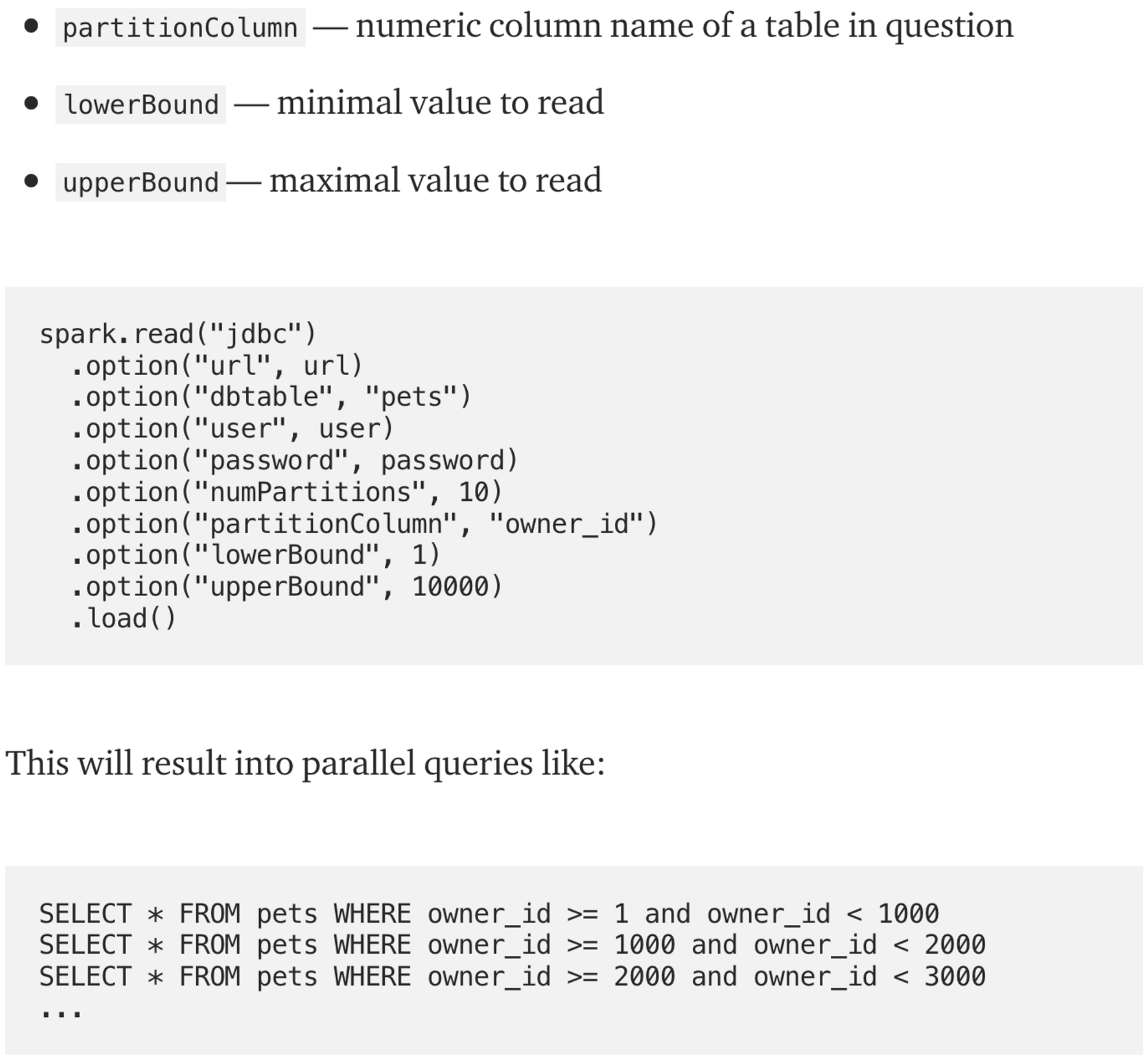
Docker compose



https://habr.com/ru/company/ruvds/blog/450312/

JDBC parallel loading

https://medium.com/@radek.strnad/tips-for-using-jdbc-in-apache-spark-sql-396ea7b2e3d3



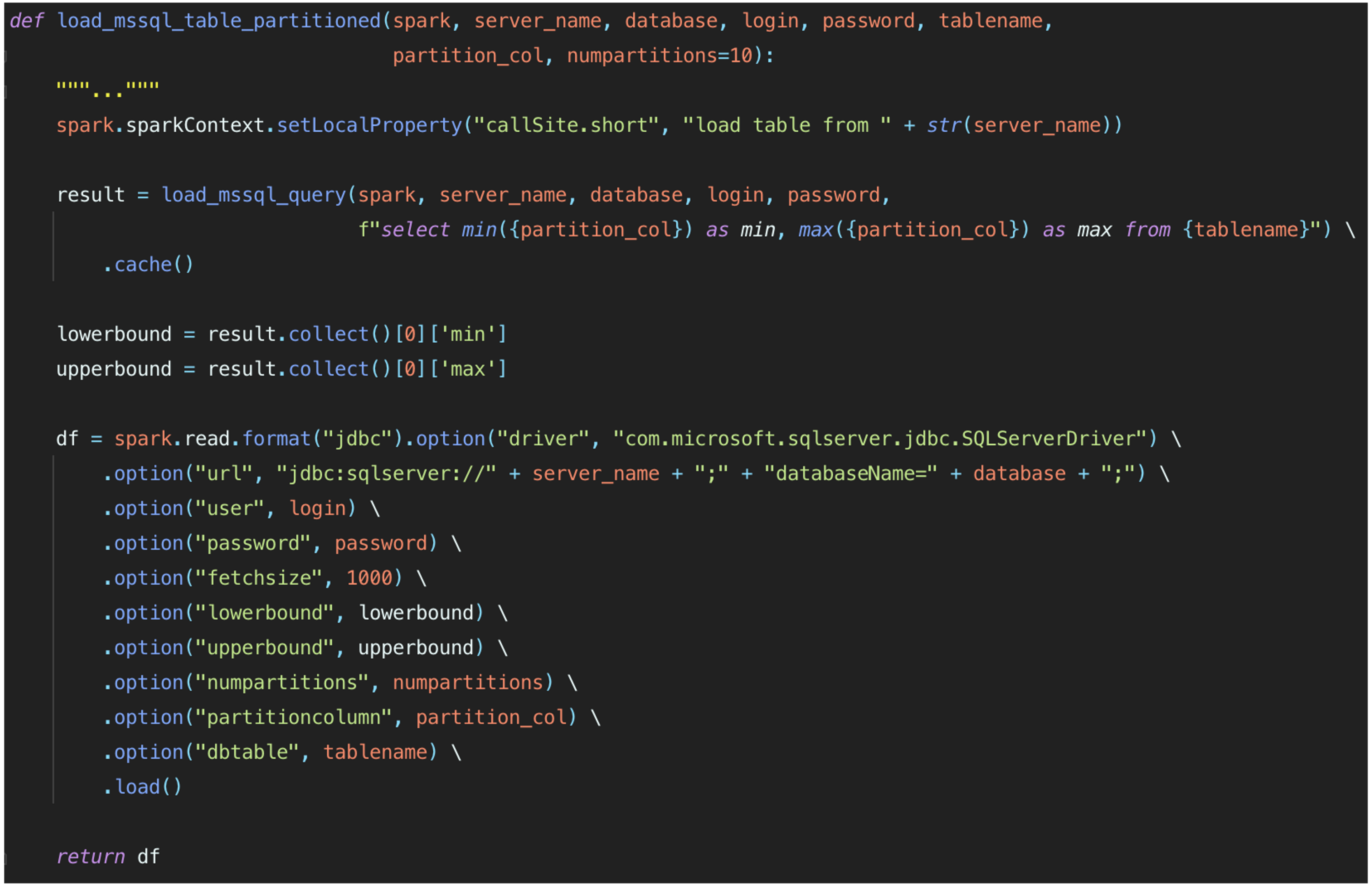
- `partitionColumn` — numeric column name of a table in question
- `lowerBound` — minimal value to read
- `upperBound` — maximal value to read

```
spark.read("jdbc")
  .option("url", url)
  .option("dbtable", "pets")
  .option("user", user)
  .option("password", password)
  .option("numPartitions", 10)
  .option("partitionColumn", "owner_id")
  .option("lowerBound", 1)
  .option("upperBound", 10000)
  .load()
```

This will result into parallel queries like:

```
SELECT * FROM pets WHERE owner_id >= 1 and owner_id < 1000
SELECT * FROM pets WHERE owner_id >= 1000 and owner_id < 2000
SELECT * FROM pets WHERE owner_id >= 2000 and owner_id < 3000
...
```

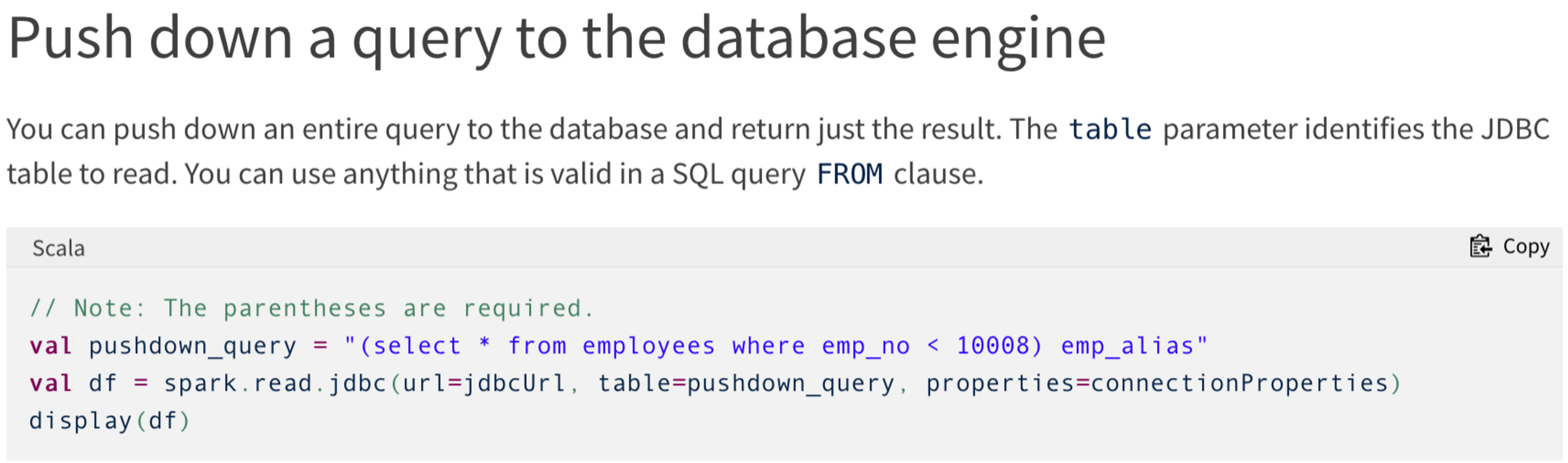
JDBC parallel loading



```python
def load_mssql_table_partitioned(spark, server_name, database, login, password, tablename,
                                 partition_col, numpartitions=10):
    """..."""
    spark.sparkContext.setLocalProperty("callSite.short", "load table from " + str(server_name))

    result = load_mssql_query(spark, server_name, database, login, password,
                              f"select min({partition_col}) as min, max({partition_col}) as max from {tablename}") \
        .cache()

    lowerbound = result.collect()[0]['min']
    upperbound = result.collect()[0]['max']

    df = spark.read.format("jdbc").option("driver", "com.microsoft.sqlserver.jdbc.SQLServerDriver") \
        .option("url", "jdbc:sqlserver://" + server_name + ";" + "databaseName=" + database + ";") \
        .option("user", login) \
        .option("password", password) \
        .option("fetchsize", 1000) \
        .option("lowerbound", lowerbound) \
        .option("upperbound", upperbound) \
        .option("numpartitions", numpartitions) \
        .option("partitioncolumn", partition_col) \
        .option("dbtable", tablename) \
        .load()

    return df
```

JDBC push down

https://docs.databricks.com/data/data-sources/sql-databases.html



Push down a query to the database engine

You can push down an entire query to the database and return just the result. The `table` parameter identifies the JDBC table to read. You can use anything that is valid in a SQL query `FROM` clause.

```scala
// Note: The parentheses are required.
val pushdown_query = "(select * from employees where emp_no < 10008) emp_alias"
val df = spark.read.jdbc(url=jdbcUrl, table=pushdown_query, properties=connectionProperties)
display(df)
```

Задание

Клонировать репозиторий https://github.com/eakot/spark-course.git

Добавить контейнер со спарк приложением, которое выгрузит распределение строк по возрасту в таблице public.bank по возрасту и сохранит в файл data/age.csv. Вывести также планы запросов с агрегацией на стороне спарка и базы

Ссылки

https://medium.com/@saschagrunert/demystifying-containers-part-i-kernel-space-2c53d6979504