https://tinyurl.com/ju44kmp8

CS 544 - Fall 2024 Lecture Snippets

Lecture 41 - Cloud Deployment

Server account creation (replace PROJECT)

- gcloud auth login
- gcloud iam service-accounts create docker
- gcloud projects add-iam-policy-binding PROJECT
  --member=serviceAccount:docker@PROJECT.iam.gserviceaccount.com
  --role=roles/artifactregistry.reader
- gcloud projects add-iam-policy-binding PROJECT
  --member=serviceAccount:docker@PROJECT.iam.gserviceaccount.com
  --role=roles/artifactregistry.writer
- gcloud iam service-accounts keys create docker.key
  --iam-account=docker@PROJECT.iam.gserviceaccount.com

Registry authentication

- gcloud auth activate-service-account --key-file=docker.key
- gcloud auth print-access-token | **docker login** -u oauth2accesstoken --password-stdin us-docker.pkg.dev/prep
- **docker logout** us-docker.pkg.dev/cs544-f24/prep

Lecture 39 - BigQuery

```
# practice data 1
import pandas as pd
import numpy as np
df = pd.DataFrame({"x1": np.random.randint(0, 10, 100).astype(float),
                   "x2": np.random.randint(0, 3, 100).astype(float)})
df["y"] = df["x1"] + df["x2"] + np.random.rand(len(df))

# practice data 2
df = pd.DataFrame({"x1": np.random.choice(["A", "B", "C"], 100),
                   "x2": np.random.randint(0, 10, 100).astype(float)})
df["y"] = df["x1"].map({"A":1, "B":5, "C":15}) + 2 * df["x2"] + np.random.rand(len(df))
```

Lecture 38 - BigQuery

 # example 1: BigQuery Table
config = bigquery.LoadJobConfig(source_format=???, write_disposition="WRITE_TRUNCATE")
job = bq.load_table_from_uri(???, ???, job_config=config)
job.result()

 # example 2: External Table (GCS)
config = bigquery.ExternalConfig(source_format=???)
config.source_uris = [???]
# config.autodetect = True
table = bigquery.Table(???)
table.external_data_configuration = config
bq.create_table(table, exists_ok=True)

 # example 3: external table (sheets)
form: https://forms.gle/AfeLnF4ydAZ5HpHn7
sheet:
https://docs.google.com/spreadsheets/d/13e14LzDDm9U4y2KddlKFAy7exNdbo1OwJa-OTe4ywiw/edit?usp=sharing
config = bigquery.ExternalConfig(????)
config.source_uris = ["????"]
# config.autodetect = True
table = bigquery.Table("????")
table.external_data_configuration = config
bq.create_table(table, exists_ok=True)

 # Wisconsin Capitol
ST_GEOGPOINT(-89.384107, 43.074715)

Lecture 37 - BigQuery

Links

- https://pages.cs.wisc.edu/~harter/cs544/data/hdma-wi-2021.parquet
- https://console.cloud.google.com/marketplace
- https://developers.google.com/identity/protocols/oauth2/scopes
- https://cloud.google.com/sustainability/region-carbon
- https://cloud.google.com/bigquery/pricing#on_demand_pricing

Jupyter/VM setup

- pip3 install jupyterlab google-cloud-bigquery google-cloud-bigquery-storage pyarrow tqdm ipywidgets pandas geopandas matplotlib db-dtypes **pandas-gbq**

- gcloud auth application-default login --scopes=openid,https://www.googleapis.com/auth/cloud-platform,https://www.googleapis.com/auth/drive.readonly
- gcloud auth application-default revoke
- python3 -m jupyterlab --no-browser
- %load_ext google.cloud.bigquery

Lecture 34 - Spark Streaming

```
FROM ubuntu:22.04

# apt
RUN apt-get update; apt-get install -y wget curl openjdk-11-jdk
python3-pip net-tools lsof nano
ENV JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

# Jupyter
RUN pip3 install jupyterlab==4.0.3 pandas==2.1.1 matplotlib==3.8.0
kafka-python==2.0.2 \
    grpcio==1.58.0 grpcio-tools==1.58.0

# Kafka (see https://kafka.apache.org/quickstart, KRaft config)
RUN wget
https://downloads.apache.org/kafka/3.6.2/kafka_2.13-3.6.2.tgz && tar
-xf kafka_2.13-3.6.2.tgz && rm kafka_2.13-3.6.2.tgz

# from bin/kafka-storage.sh random-uuid
ENV KAFKA_CLUSTER_ID=dCHffFWYTCKWXiesmJMN9w

# SPARK
RUN wget
https://dlcdn.apache.org/spark/spark-3.5.3/spark-3.5.3-bin-hadoop3.tg
z && tar -xf spark-3.5.3-bin-hadoop3.tgz && rm
spark-3.5.3-bin-hadoop3.tgz
RUN pip3 install pyspark==3.5.3

CMD sh -c "cd /kafka_2.13-3.6.2 && \
    bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c
config/kraft/server.properties && \
    bin/kafka-server-start.sh config/kraft/server.properties
--override auto.create.topics.enable=false"
```

```
docker run -d --name=broker -p 4040:4040 -p 5000:5000 -v `pwd`/nb:/nb
kafka-demo
docker exec -d broker python3 -m jupyterlab --no-browser --ip=0.0.0.0
--port=5000 --allow-root --NotebookApp.token=''


# Spark session (with Kafka jar)
from pyspark.sql import SparkSession
spark = (SparkSession.builder.appName("demo")
         .config('spark.jars.packages',
'org.apache.spark:spark-sql-kafka-0-10_2.12:3.5.3')
         .getOrCreate())
```

Lecture 33 - Kafka Reliability

```
# animal sighting protobuf
syntax = "proto3";

message Sighting {
    string beach = 1;
    string animal = 2;
}
```

python3 -m grpc_tools.protoc -I=. --python_out=. animals.proto

Lecture 32 - Kafka Demos

Dockerfile

```
FROM ubuntu:22.04

# apt
RUN apt-get update; apt-get install -y wget curl openjdk-11-jdk
python3-pip net-tools lsof nano
ENV JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64

# Python
RUN pip3 install jupyterlab==4.0.3 pandas==2.1.1 matplotlib==3.8.0
kafka-python==2.0.2 grpcio==1.58.0 grpcio-tools==1.58.0

# Kafka (see https://kafka.apache.org/quickstart, KRaft config)
```

```
RUN wget
https://downloads.apache.org/kafka/3.6.2/kafka_2.13-3.6.2.tgz && tar
-xf kafka_2.13-3.6.2.tgz && rm kafka_2.13-3.6.2.tgz

# from bin/kafka-storage.sh random-uuid
ENV KAFKA_CLUSTER_ID=dCHffFWYTCKWXiesmJMN9w

CMD sh -c "cd /kafka_2.13-3.6.0 && \
    bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c
config/kraft/server.properties && \
    bin/kafka-server-start.sh config/kraft/server.properties
--override auto.create.topics.enable=false"
```

Launch

```
docker run -d --name=broker -p 5000:5000 -v `pwd`/nb:/nb kafka-demo
docker exec -d broker python3 -m jupyterlab --no-browser --ip=0.0.0.0
--port=5000 --allow-root --NotebookApp.token=''
```

Starter Code

```
from threading import Thread, Lock

lock = Lock()
def Print(*args):
    with lock:
        print(*args)

Print("hi")
```

Lecture 26 - Cassandra

 # Cluster Setup

https://git.doit.wisc.edu/cdis/cs/courses/cs544/f24/main/-/tree/main/lec/26-cassandra?ref_type=heads

 # Cassandra from Python

```
from cassandra.cluster import Cluster
cluster = Cluster(["demo-db-1", "demo-db-2", "demo-db-3"])
cass = cluster.connect()
```

Lecture 24 - Spark ML

```
import pandas as pd
import numpy as np
```

```
df = pd.DataFrame({"x1": np.random.randint(0, 10, 100).astype(float),
                   "x2": np.random.randint(0, 3, 100).astype(float)})
df["y"] = df["x1"] + df["x2"] + np.random.rand(len(df))
df = spark.createDataFrame(df)
df
```

Lecture 23 - Spark Internals and Performance

```
# CACHING and LOAD BALANCE
import requests

# https://spark.apache.org/docs/latest/monitoring.html#rest-api
# http://localhost:4040/api/v1/applications
# http://localhost:4040/api/v1/applications/{app_id}/executors
# look for "totalTasks"

r = requests.get(????)
r.raise_for_status()
r.json()

# JOIN strategies (one machine)

# kind_id, color
fruits = [
    ("B", "Yellow"),
    ("A", "Green"),
    ("C", "Orange"),
    ("A", "Red"),
    ("C", "Purple"),
    ("B", "Green")
]

# kind_id, name (assume no duplicate kind_id's)
kinds = [
    ("A", "Apple"),
    ("B", "Banana"),
    ("C", "Carrot")
]

# GOAL: print Yellow Banana, Green Apple, etc (any order)
```

Lecture 22 - Spark SQL

! wget https://pages.cs.wisc.edu/~harter/cs544/data/sf.zip
! wget https://raw.githubusercontent.com/tylerharter/us-federal-holidays/main/holidays2.csv

```
! hdfs dfs -cp holidays2.csv hdfs://nn:9000/
holidays = (spark.read.format("csv").option("header", True)
        .load("hdfs://nn:9000/holidays2.csv"))
holidays.createTempView("holidays")
```

https://git.doit.wisc.edu/cdis/cs/courses/cs544/f24/main/-/tree/main/lec/22-spark

Lecture 21 - Spark DataFrames

Swap

```
sudo fallocate -l 1G /swapfile
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile
```

Data

https://data.sfgov.org/Public-Safety/Fire-Department-Calls-for-Service/nuek-vuh3

```
! wget https://pages.cs.wisc.edu/~harter/cs544/data/sf.zip
```

Spark session

```
from pyspark.sql import SparkSession
spark = (SparkSession.builder.appName("cs544")
      .master("spark://boss:7077")
      .config("spark.executor.memory", "256M")
      .config("spark.sql.warehouse.dir", "hdfs://nn:9000/user/hive/warehouse")
      .enableHiveSupport()
      .getOrCreate())
```

Lecture 20 - Spark RDDs

```
https://git.doit.wisc.edu/cdis/cs/courses/cs544/f24/main/-/tree/main/
lec/20-spark?ref_type=heads

from pyspark.sql import SparkSession
spark = (SparkSession.builder.appName("cs544")
         .master("spark://boss:7077")
         .config("spark.executor.memory", "512M")
         .getOrCreate())
```

```
!wget https://pages.cs.wisc.edu/~harter/cs544/data/ghcnd-stations.txt
```

Lecture 18 - HDFS

https://git.doit.wisc.edu/cdis/cs/courses/cs544/f24/main/-/tree/main/lec/18-hdfs

Build

- docker build . -f hdfs.Dockerfile -t p4-hdfs
- docker build . -f notebook.Dockerfile -t p4-nb

https://hadoop.apache.org/docs/r3.3.6/hadoop-project-dist/hadoop-hdfs/WebHDFS.html

Lecture 15 - SQL

MySQL setup

- pip3 install jupyterlab SQLAlchemy mysql-connector-python pandas pyarrow
- docker run -d -m "1g" -p 127.0.0.1:3306:3306 -e MYSQL_DATABASE=cs544 -e MYSQL_ROOT_PASSWORD=abc mysql
- docker exec -it <CONTAINER> bash
- mysql -p cs544

Connect

```
from sqlalchemy import create_engine, text
engine =
create_engine("mysql+mysqlconnector://root:abc@127.0.0.1:3306/c
s544")
conn = engine.connect()
```

Create a Table

```
# table: users
# columns: id, name, phone
# name is required
# id uniquely identifies row
conn.execute(text("""
    create table ???? (????)
"""))
```

Pandas: small tables to SQL

```
import pandas as pd
url =
"https://raw.githubusercontent.com/cfpb/api/master/resources/da
tasets/hmda/code_sheets/"
df = pd.read_csv(url + "action_taken.csv")
df.to_sql("actions", conn, index=False, if_exists="replace")
df = pd.read_csv(url + "loan_type.csv")
df.to_sql("loan_types", conn, index=False, if_exists="replace")
df = pd.read_csv(url + "loan_purpose.csv")
df.to_sql("purposes", conn, index=False, if_exists="replace")
conn.commit()
```

Pandas: large table to SQL

```
import pyarrow as pa
import pyarrow.csv, pyarrow.parquet
pa.parquet.write_table(pa.csv.read_csv("hdma-wi-2021.csv"),
"loans.parquet")
t = pa.parquet.read_table("loans.parquet", columns=["lei",
"action_taken", "loan_type", "loan_amount", "interest_rate",
"loan_purpose", "income"])
# TODO: to_sql with chunking
```

Query practice (loans example)

what are all the possible actions? Practice SELECT/FROM.
what are the first 10 loans? Practice LIMIT.
projection: choosing what columns (SELECT)
selection: filtering rows (WHERE)
what are the first 10 interest rates and loan amounts (in thousands)? Practice SELECT.
what are the loans for individuals with income over $1 million? Practice WHERE.
what are the five biggest loans in terms of dollar amount? Practice ORDER BY.
what are the actions taken and types for those loans (show the text, not numbers)? Practice INNER JOIN.
what is a loan_purpose that doesn't appear in the loans table? Practice LEFT/RIGHT JOIN.
how many rows are in the table? Practice COUNT(*).
how many non-null values are in the income column? Practice COUNT(column).
what is the average interest rate for loans of type "Conventional"? Practice AVG.
how many loans are there of each type? Practice GROUP BY.
which loan types appear at least 10,000 times? Practice HAVING.

Lecture 14 - File Formats

- wget https://pages.cs.wisc.edu/~harter/cs544/data/hdma-wi-2021.zip
- unzip hdma-wi-2021.zip

Lecture 12 - Locking

```
bank_accounts = {"x": 25, "y": 100, "z": 200} # in dollars
lock = threading.Lock() # protects bank_accounts

def transfer(src, dst, amount):
    lock.acquire()
    success = False
    if bank_accounts[src] >= amount:
```

```
        bank_accounts[src] -= amount
        bank_accounts[dst] += amount
        success = True
    print("transferred" if success else "denied")
    lock.release()
```

Lecture 10 - Caching and PyArrow

Docker Image:
https://git.doit.wisc.edu/cdis/cs/courses/cs544/f24/main/-/tree/main/lec/10-arrow

STARTER

```
import pyarrow as pa
import pyarrow.compute as pc

batch = pa.RecordBatch.from_arrays([range(1,1_000_000),
                                    range(1,1_000_000),
                                    range(1,1_000_000)],
                                   names=["x", "y", "z"])
print(batch.nbytes / 1024**2)

with pa.ipc.new_file("test.arrow", schema=batch.schema) as f:
    for i in range(50):
        f.write_batch(batch)
```

Lecture 9 - Caching

```
import random
import requests
import pandas as pd

r =
requests.get("https://pages.cs.wisc.edu/~harter/cs544/data/wi-station
s/stations.txt")
r.raise_for_status()
stations = r.text.strip().split("\n")
stations = random.sample(stations, k=10)
workload = random.choices(stations, k=100, weights=[0.3, 0.2] +
[0.5/8]*8)

station = TODO
```

```
df =
pd.read_csv(f"https://pages.cs.wisc.edu/~harter/cs544/data/wi-stations/{station}.csv.gz",
            names=["station", "date", "element", "value", "m", "q", "s", "obs"], low_memory=False)
```

Lecture 7 - Compose

```
https://git.doit.wisc.edu/cdis/cs/courses/cs544/f24/main/-/tree/main/lec/07-compose
```

Lecture 6 - gRPC

Code Gen

```
python3 -m venv venv
source venv/bin/activate
pip install grpcio-tools==1.66.1 grpcio==1.66.1 protobuf==5.27.2
```

```
python3 -m grpc_tools.protoc -I=. --python_out=. --grpc_python_out=. ????.proto
```

Server Starter

```
import grpc
from concurrent import futures

print("start server")
server = grpc.server(futures.ThreadPoolExecutor(max_workers=1),
options=[("grpc.so_reuseport", 0)])
# TODO: add servicer
server.add_insecure_port("0.0.0.0:5440")
server.start()
server.wait_for_termination()
```

Client Starter

```
channel = grpc.insecure_channel("localhost:????")
stub = ????_pb2_grpc.????Stub(channel)
```

Lecture 4 - Docker

- apt-get update

- apt-get install unzip # python3 python3-pip
- pip install pandas==2.2.2 --break-system-packages

Lecture 2 - Shell

https://pages.cs.wisc.edu/~harter/cs544/data/ghcnd-stations.txt

https://code.visualstudio.com/
https://code.visualstudio.com/docs/remote/ssh