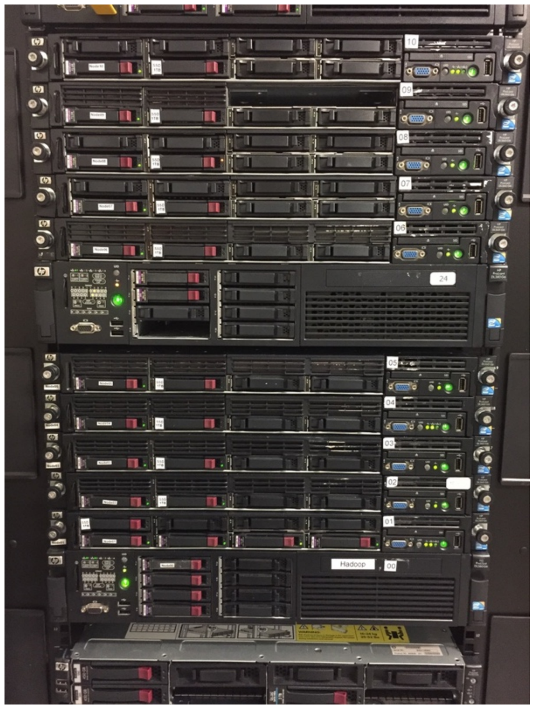
Hands-On Exercise 2-1
Running PostgreSQL DBMS using Docker

1 CSCluster at UIS



11-node Hadoop nodes

- CPU: Intel Xeon **4C/8T** per node
- Memory: **266 GB** in total
  - Master (head) node: 74GB RAM
  - 2nd Master node: 48GB RAM
  - 9 worker nodes: 16GB/node
- Storage: **10 TB SSD** in total
  - Master node: 400GB,
  - 10 worker nodes: 1TB SSD/node
- Hadoop: **Cloudera CDH 6.3**
  - HDFS, MapReduce
  - Spark 2, HBase, Hive
  - Pig, HUE, and etc.

VMs for Docker/K8S (You will use these VMs)

- CPU: Intel Xeon **8** core/node
- Memory: **16GB** RAM/node
- Storage: **100GB**

3-node Cassandra cluster (NoSQL-Column family DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **48GB** RAM (16GB/node)
- Storage: **3TB SSD** (1TB/node)

PostgreSQL node (Relational DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **18 GB RAM**
- Storage: **1TB SSD**

MongoDB node (NoSQL-Document DB)

- CPU: Intel Xeon **4C/8T** per node
- Memory: **16 GB RAM**
- Storage: **1TB SSD**

2 Accessing your VM for Docker and Kubernetes

2.1 Accessing campus network

If you are in UIS campus, you are fine. If you are not in UIS campus, you should install a Cisco VPN client software. The VPN client software gives remote users a secure and encrypted VPN (Virtual Private Network) connection to the UIS campus network. Please see below website, https://www.uis.edu/informationtechnologyservices/connect/vpn/

2.2 Accessing your VM using SSH

After you install the VPN client software and make a VPN connection to UIS network, you can access your VM using a terminal (Mac), PowerShell (Windows), or Putty (Windows).

- Check the IP address for your VM in the 'Course Information' under 'Modules' in Canvas

Type below command in your preferred SSH shell client:

```
ssh your-login@10.92.128.36
```

your-login: your UIS NetID (for example, **slee675** from slee675@uis.edu)
Initial password: See the Virtual Machine IPs page in the 'Course Information' under 'Modules' in Canvas.

After you logged in to your VM, you will see below prompt. The 'us2004lts' is a name of your VM and stands for 'Ubuntu Sever 20.04 LTS' that we are using as an OS.

```
your-login@us2004lts:~$
```

# Example1: using terminal in Mac

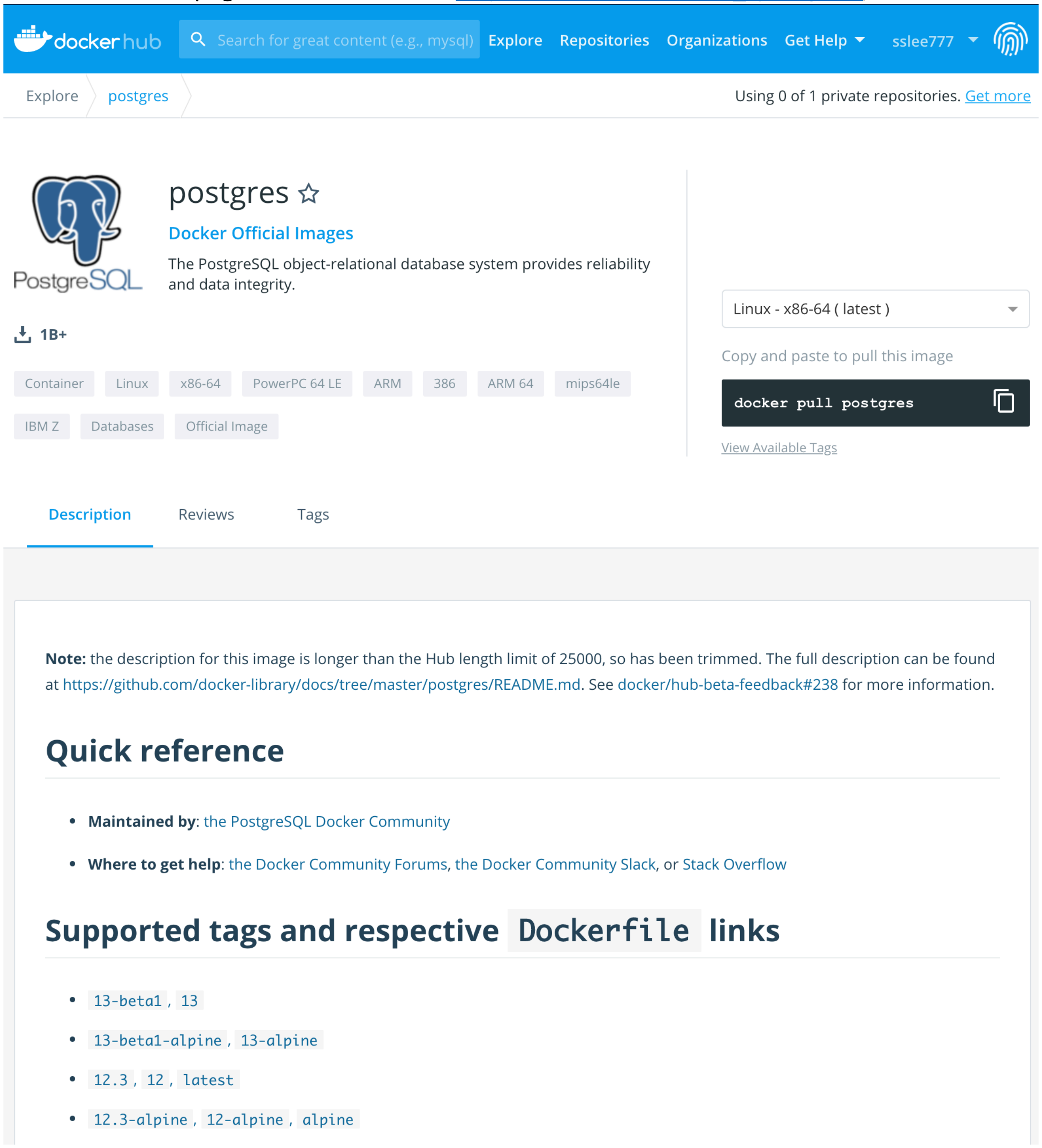
```
LeeMBP15:~ sslee777$ ssh sslee777@10.92.128.36
The authenticity of host '10.92.128.36 (10.92.128.36)' can't be established.
ECDSA key fingerprint is SHA256:dXhKfHsYIXe/53hvU+HOK2V6fVrTbz/QxmUhpnPXpzA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.92.128.36' (ECDSA) to the list of known hosts.
sslee777@10.92.128.36's password: <== Use initial password until you change it
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-33-generic x86_64)
...

sslee777@us2004lts:~$
```

3 Running PostgreSQL Database with Docker

3.1 PostgreSQL in the Docker Hub

PostgreSQL is a powerful, open source object-relational database system with over 30 years of active development that has earned it a strong reputation for reliability, feature robustness, and performance. See https://www.postgresql.org/

To run PostgreSQL on your local machine, let's try to pull the PostgreSQL image from the docker hub. The official page is can be found in https://hub.docker.com/_/postgres/, see below.



The PostgreSQL object-relational database system provides reliability and data integrity.

```
docker pull postgres
```

Note: the description for this image is longer than the Hub length limit of 25000, so has been trimmed. The full description can be found at https://github.com/docker-library/docs/tree/master/postgres/README.md. See docker/hub-beta-feedback#238 for more information.

Quick reference

- **Maintained by**: the PostgreSQL Docker Community
- **Where to get help**: the Docker Community Forums, the Docker Community Slack, or Stack Overflow

Supported tags and respective `Dockerfile` links

- `13-beta1`, `13`
- `13-beta1-alpine`, `13-alpine`
- `12.3`, `12`, `latest`
- `12.3-alpine`, `12-alpine`, `alpine`

3.2 Pulling the image

Now it’s time to pull the PostgreSQL image from the Docker hub. Make sure Docker is installed so you can run the commands below.
The following command downloads a latest PostgreSQL docker image.

```
sslee777@us2004lts:~$ docker pull postgres:latest
alpine: Pulling from library/postgres
cbdbe7a5bc2a: Pull complete
b52a8a2ca21a: Pull complete
e36a19831e31: Pull complete
7e7a80421a93: Pull complete
16f0b643a029: Pull complete
ba13a61f3843: Pull complete
147ca166d4ae: Pull complete
650f8b751c65: Pull complete
Digest: sha256:ec7cfff29672a2f676c11cc53ae7dafe63a57ccefc2b06ea423493227da29b9c
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:alpine
```

Let’s check whether the download was successful by run below command. You see postgres (latest) image in the list.

```
sslee777@us2004lts:~$ docker image ls
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
test                latest              a1ea900d3495        4 days ago          82.6MB
alpine              latest              a24bb4013296        7 days ago          5.57MB
postgres            latest              adf2b126dda8        3 weeks ago         313MB
ubuntu              latest              1d622ef86b13        6 weeks ago         73.9MB
hello-world         latest              bf756fb1ae65        5 months ago        13.3kB
```

3.3 Creating and running a container by using the image

You can run a Postgres container by specifying all the necessary information in one command. The command tells Docker to run a new container under a particular container name, defines the Postgres password, and downloads the latest Postgres release. See docker run command: https://docs.docker.com/engine/reference/commandline/run/

```
sslee777@us2004lts:~$ docker run --name mypostgresqlname -e
POSTGRES_PASSWORD=mysecretpassword -d postgres:latest

e64143c099c758fbbcce8c291d9868f81e427ada7758504a800de1fb295f8fca

sslee777@us2004lts:~$
```

Here are the operations that are broken down:

- **--name**: We’re naming the container ‘mypostgresqlname’.
- **-e**: Set the password to “mysecretpassword” using the environment tag -e.
- **-d**: We're running detached (-d) mode (so in the background).
- use postgres:latest to launch the container.

- **- p**: Port. We may specify a mapping between the port of your host machine and the port of your container, in format host_port:container_port. (Default port for PostgreSQL is 5432)
- **-v**: Volume. We may specify mapping between the project folder on your host machine and the PostgreSQL folder in your container, in format host_folder:container_port

We didn't specify ports and volumes in this exercise; however, we will do it in next week's exercise.

If you didn't pull the image before, docker will check whether you have the image in your local directory. If you don't have it in your local directory, Docker will pull the image automatically. Following is such an example, see below.

```
sslee777@us2004lts:~$ docker run --name mypostgresqlname -e
POSTGRES_PASSWORD=mysecretpassword -d postgres:latest
Unable to find image 'postgres:latest' locally
latest: Pulling from library/postgres
afb6ec6fdc1c: Pull complete
51be5f829bfb: Pull complete
e707c08f571a: Pull complete
98ddd8bce9b5: Pull complete
5f16647362a3: Pull complete
5d56cdf9ab3b: Pull complete
2207a50ca41d: Pull complete
a51d14a628f3: Pull complete
24dcb11335d0: Extracting  20.05MB/73.03MB
54cc759cb0bb: Download complete
debc11d66570: Download complete
3ffd0589b5fc: Download complete
490b7ee49751: Download complete
Digest: sha256:ec7cfff29672a2f676c11cc53ae7dafe63a57ccefc2b06ea423493227da29b9c
Status: Downloaded newer image for postgres:latest
fa2417fe21be363f297d1e1368fef16a97ed864ff3e2d81d69bbf763b9fec4a0
```

Confirm your PostgreSQL container is now up by prompting Docker to list all running containers with:
Remember the container ID: e64143c099c7

```
sslee777@us2004lts:~$ docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS         PORTS      NAMES
e64143c099c7   postgres:latest  "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   5432/tcp   mypostgresqlname
sslee777@us2004lts:~$
```

Let's check the log output with:

```
sslee777@us2004lts:~$ docker logs mypostgresqlname
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale "en_US.utf8".
The default database encoding has accordingly been set to "UTF8".
The default text search configuration will be set to "english".

Data page checksums are disabled.

fixing permissions on existing directory /var/lib/postgresql/data ... ok
creating subdirectories ... ok
selecting dynamic shared memory implementation ... posix
selecting default max_connections ... 100
selecting default shared_buffers ... 128MB
selecting default time zone ... Etc/UTC
creating configuration files ... ok
running bootstrap script ... ok
performing post-bootstrap initialization ... ok
syncing data to disk ... ok
Success. You can now start the database server using:
    pg_ctl -D /var/lib/postgresql/data -l logfile start

waiting for server to start....2020-05-01 03:57:24.094 UTC [47] LOG:  starting PostgreSQL 12.3
(Debian 12.3-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-
bit
2020-05-01 03:57:24.096 UTC [47] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2020-05-01 03:57:24.117 UTC [48] LOG:  database system was shut down at 2020-05-01 03:57:23
UTC
2020-05-01 03:57:24.122 UTC [47] LOG:  database system is ready to accept connections
 done
server started

/usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*

2020-05-01 03:57:24.183 UTC [47] LOG:  received fast shutdown request
waiting for server to shut down....2020-05-01 03:57:24.187 UTC [47] LOG:  aborting any active
transactions
2020-05-01 03:57:24.188 UTC [47] LOG:  background worker "logical replication launcher" (PID
54) exited with exit code 1
2020-05-01 03:57:24.189 UTC [49] LOG:  shutting down
2020-05-01 03:57:24.209 UTC [47] LOG:  database system is shut down
 done
server stopped

PostgreSQL init process complete; ready for start up.

2020-05-01 03:57:24.309 UTC [1] LOG:  starting PostgreSQL 12.3 (Debian 12.3-1.pgdg100+1) on
x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-6) 8.3.0, 64-bit
2020-05-01 03:57:24.309 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
2020-05-01 03:57:24.309 UTC [1] LOG:  listening on IPv6 address "::", port 5432
2020-05-01 03:57:24.317 UTC [1] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
2020-05-01 03:57:24.352 UTC [56] LOG:  database system was shut down at 2020-05-01 03:57:24
UTC
2020-05-01 03:57:24.361 UTC [1] LOG:  database system is ready to accept connections
sslee777@us2004lts:~$
```

3.4 Create a database in the container

Your PostgreSQL container is running. Now we'll create a database to load a dataset by using 'docker exec' to launch an interactive shell running inside our mypostgresqlname container, which has the PostgreSQL CLI tools installed. This saves us from needing to have any tools to connect to and manage PostgreSQL databases installed locally.

```
sslee777@us2004lts:~$ docker exec -it mypostgresqlname bash
root@e64143c099c7:/#
```

Recall your memory! Your container ID: e64143c099c7. You see the container ID as a hostname of your container.

Inside that shell we can ask it to create a new database with the name 'mydb'.

```
root@e64143c099c7:/# createdb -U postgres mydb
root@e64143c099c7:/#
```

And then let's launch the psql utility which is a CLI tool for PostgreSQL, connected to our mydb database:

```
root@e64143c099c7:/# psql -U postgres mydb
psql (12.3 (Debian 12.3-1.pgdg100+1))
Type "help" for help.

mydb=#             <== Prompt
```

'postgres' is a default superuser or administrator like 'root' of your PostgreSQL DB.

Now inside psql, let's run some basic commands. Lists the databases. We'll also ask for the database version, and the current date:

'\l' lists the databases:

```
mydb=# \l
                                 List of databases
   Name    |  Owner   | Encoding |  Collate   |   Ctype    |   Access privileges
-----------+----------+----------+------------+------------+-----------------------
 mydb      | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres  | postgres | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
           |          |          |            |            | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_US.utf8 | en_US.utf8 | =c/postgres          +
           |          |          |            |            | postgres=CTc/postgres
(4 rows)

mydb=#
```

You see your database 'mydb' in the list.

Show database version:

```
mydb=# select version();
                                    version
------------------------------------------------------------------------------------
PostgreSQL 12.3 (Debian 12.3-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc
(Debian 8.3.0-6) 8.3.0, 64-bit
(1 row)

mydb=#
```

It shows you are using **PostgreSQL 12.3**

Show the current date:

```
mydb=# select current_date;
 current_date
--------------
 2020-05-01
(1 row)

mydb=#
```

To quit from the **psql**
Psql **-->** bash shell (container) --> your virtual machine (VM)

```
mydb=# \q
root@e64143c099c7:/#
```

After that, you can exit from the shell in the container.
Psql --> bash shell (container) **-->** your virtual machine (VM)

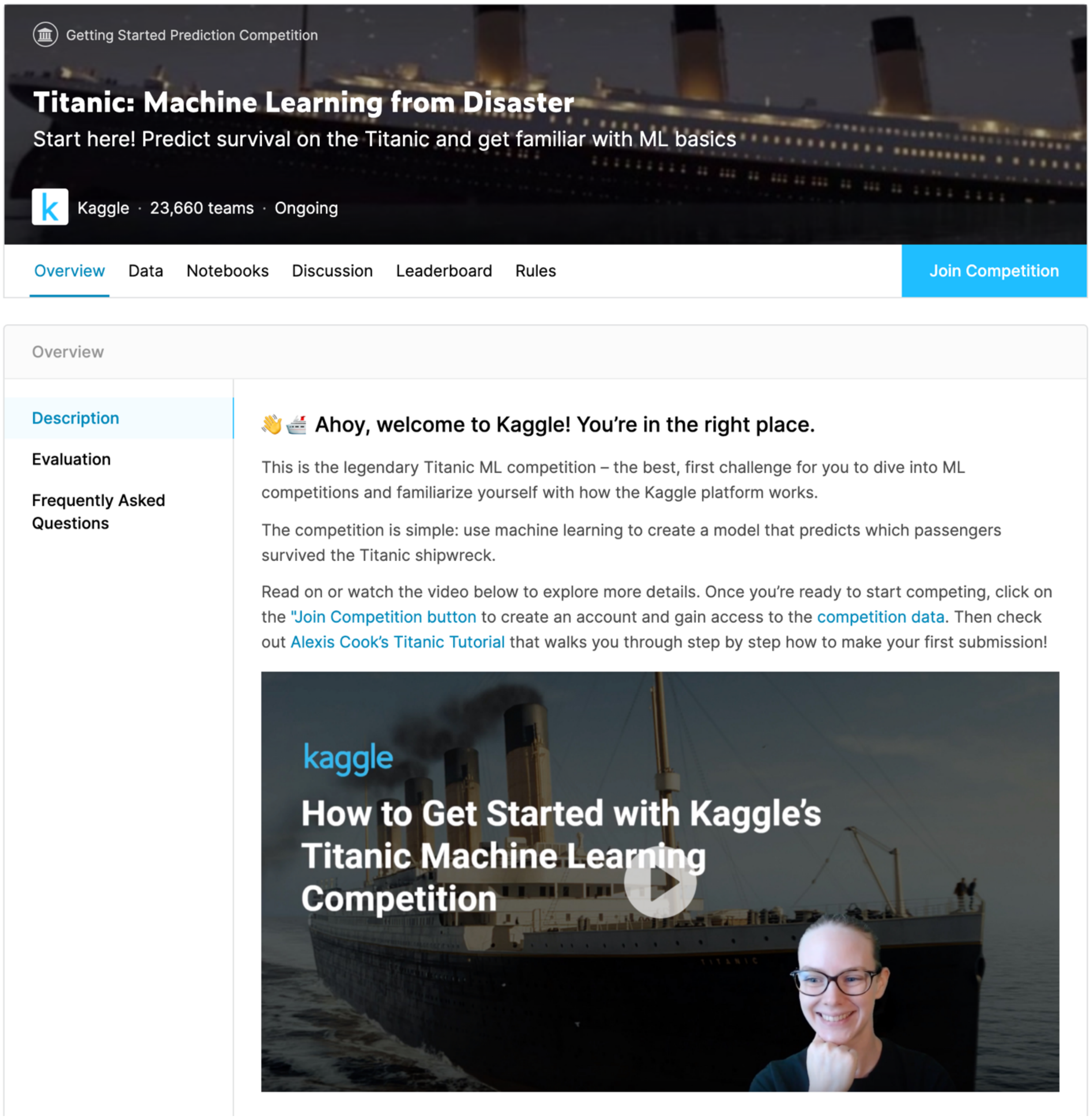
```
root@e64143c099c7:/# exit
exit
sslee777@us2004lts:~$
```

Now you are in your VM.

3.5 Populate a table in the PostgreSQL container

Let's do something a bit more interesting. We'll download a dataset, create a table and populate the table with the downloaded dataset.

The dataset we will use is a titanic dataset from Kaggle. It is the legendary Titanic ML competition – the best, first challenge for someone who are interested in diving into ML competitions and familiarize yourself with how the Kaggle platform works. See https://www.kaggle.com/c/titanic/overview.


Getting Started Prediction Competition
Titanic: Machine Learning from Disaster
Start here! Predict survival on the Titanic and get familiar with ML basics
Kaggle · 23,660 teams · Ongoing
Overview
Data
Notebooks
Discussion
Leaderboard
Rules
Join Competition
Overview
Description
Evaluation
Frequently Asked Questions
kaggle
How to Get Started with Kaggle's Titanic Machine Learning Competition

👋🚢 Ahoy, welcome to Kaggle! You're in the right place.

This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works.

The competition is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

Read on or watch the video below to explore more details. Once you're ready to start competing, click on the "Join Competition button to create an account and gain access to the competition data. Then check out Alexis Cook's Titanic Tutorial that walks you through step by step how to make your first submission!

We will use same, but simplified dataset from Stanford university. Run 'wget' command with following URL to download the file.

```
sslee777@us2004lts:~$ wget
http://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv

--2020-05-01 21:34:53--
http://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv
Resolving web.stanford.edu (web.stanford.edu)... 171.67.215.200
Connecting to web.stanford.edu (web.stanford.edu)|171.67.215.200|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 44225 (43K) [text/csv]
Saving to: ‘titanic.csv’

titanic.csv           100%[=========================>]  43.19K  --.-KB/s    in 0.1s

2020-05-01 21:34:53 (350 KB/s) - ‘titanic.csv’ saved [44225/44225]
sslee777@us2004lts:~$
```

Double check your current directory

```
sslee777@us2004lts:~$ ls
titanic.csv

sslee777@us2004lts:~$
```

Then copy the file into root directory in your container. Run below command

```
sslee777@us2004lts:~$ docker cp titanic.csv mypostgresqlname:/.
sslee777@us2004lts:~$
```

After you connect your PostgreSQL container, run ‘ls’ command to check the file.
You can see the file, titanic.csv in the list.

```
sslee777@us2004lts:~$ docker exec -it mypostgresqlname bash
root@e64143c099c7:/# ls
bin   docker-entrypoint-initdb.d  home   media proc  sbin  titanic.csv  var
boot  docker-entrypoint.sh  lib   mnt   root  srv   tmp
dev   etc                   lib64  opt  run   sys   usr

root@e64143c099c7:/#
```

Let’s connect your DB using ‘psql’ to create a table to store the data.

```
root@e64143c099c7:/# psql -U postgres mydb
psql (12.3 (Debian 12.3-1.pgdg100+1))
Type "help" for help.

mydb=#
```

After you connect your DB, run following CREATE TABLE SQL command by copy & paste below:

```
CREATE TABLE passengers (
    id SERIAL NOT NULL PRIMARY KEY,
    survived INT,
    pclass INT,
    name VARCHAR(255),
    sex TEXT,
    age FLOAT8,
    siblings_spouses INT,
    parents_children INT,
    fare FLOAT8
);
```

You will have a passengers table to store the titanic dataset.

```
mydb=# CREATE TABLE passengers (
mydb(#     id SERIAL NOT NULL PRIMARY KEY,
mydb(#     survived INT,
mydb(#     pclass INT,
mydb(#     name VARCHAR(255),
mydb(#     sex TEXT,
mydb(#     age FLOAT8,
mydb(#     siblings_spouses INT,
mydb(#     parents_children INT,
mydb(#     fare FLOAT8
mydb(# );
CREATE TABLE
mydb=#
```

Verify the passengers table by following command

```
mydb=# \dt
           List of relations
 Schema |    Name    | Type  |  Owner
--------+------------+-------+----------
 public | passengers | table | postgres
(1 row)
```

You have a table for titanic dataset.

Once the table is created, we'll import the CSV data over to PostgreSQL using psql's '\copy' command. Substitute the /path/to/titanic.csv with the path of your CSV file.

```
\copy passengers (survived, pclass, name, sex, age, siblings_spouses,
parents_children, fare) from '/path/to/titanic.csv' CSV HEADER;
```

```
mydb=# \copy passengers (survived, pclass, name, sex, age, siblings_spouses,
parents_children, fare) from '/titanic.csv' CSV HEADER;
COPY 887
mydb=#
```

That inserted 887 passengers into your database that we can work with.

3.6 Querying

Before we querying, let's check if we will lose data if we stop the container?

Stop and restart the container by running following commands

```
sslee777@us2004lts:~$ docker stop mypostgresqlname
mypostgresqlname
sslee777@us2004lts:~$ docker start mypostgresqlname
mypostgresqlname
sslee777@us2004lts:~$
```

Reconnect it

```
sslee777@us2004lts:~$ docker exec -it mypostgresqlname bash
root@e64143c099c7:/#
```

Launch 'psql' to query

```
root@e64143c099c7:/# psql -U postgres mydb
psql (12.3 (Debian 12.3-1.pgdg100+1))
Type "help" for help.

mydb=#
```

Now we can access data in the database stored in the PostgreSQL container:

Let's query to give us the average age of passengers who survived the Titanic disaster by ticket class. Pclass is a ticket class (1 = 1st, 2 = 2nd, 3 = 3rd).

```
mydb=# SELECT avg(age), pclass FROM passengers WHERE survived = 1 GROUP BY pclass;
        avg        | pclass
-------------------+--------
 35.96264705882353 |      1
 21.41109243697479 |      3
 26.17045977011494 |      2
(3 rows)

mydb=#
```

We are able to run a query and data is still there after restart the container.

4 Submit

Submit a word document (docx, doc, or PDF) as a hands-on exercise report to Canvas.
Assignments

- Take screenshots and attach them in your report
    a. Pulling a PostgreSQL image

b. Running your PostgreSQL container
c. Creating a database and table for titanic dataset in your PostgreSQL DBMS
d. Running an analytic query “average age of passengers who survived the Titanic disaster by ticket class” and results

If you have any problems or questions regarding this exercise, post messages in the ‘Discussions’ in Canvas.

5 Reference

- PostgreSQL: The World's Most Advanced Open Source Relational Database, https://www.postgresql.org/
- PostgreSQL 12.3 Documentation, https://www.postgresql.org/docs/12/index.html
- Setting up Docker & PostgreSQL — Connecting Locally — Using Advanced Functions, https://medium.com/@rrfd/setting-up-docker-postgresql-connecting-locally-using-advanced-functions-d8fe3bd58de6
- Exploring PostgreSQL with Docker, https://markheath.net/post/exploring-postgresql-with-docker
- Rapid API development tutorial with Docker + PostgreSQL + PostGraphile, https://medium.com/coderbunker/rapid-api-development-tutorial-with-docker-postgresql-postgraphile-e387c1c73dd8
- Postgres Docker Official Images - start a postgres instance https://hub.docker.com/_/postgres/
- Titanic: Machine Learning from Disaster - Start here! Predict survival on the Titanic and get familiar with ML basics, https://www.kaggle.com/c/titanic/overview
- Getting Started with Compose PostgreSQL and Jupyter Notebooks, https://www.compose.com/articles/getting-started-with-compose-postgresql-and-jupyter-notebooks/