|  |  |
| --- | --- |
|  | **TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  **KHOA CÔNG NGHỆ THÔNG TIN** |

**TỔNG HỢP CÁC BÀI LAB MÔN ĐIỆN TOÁN ĐÁM MÂY**

1. **THÔNG TIN CHUNG**

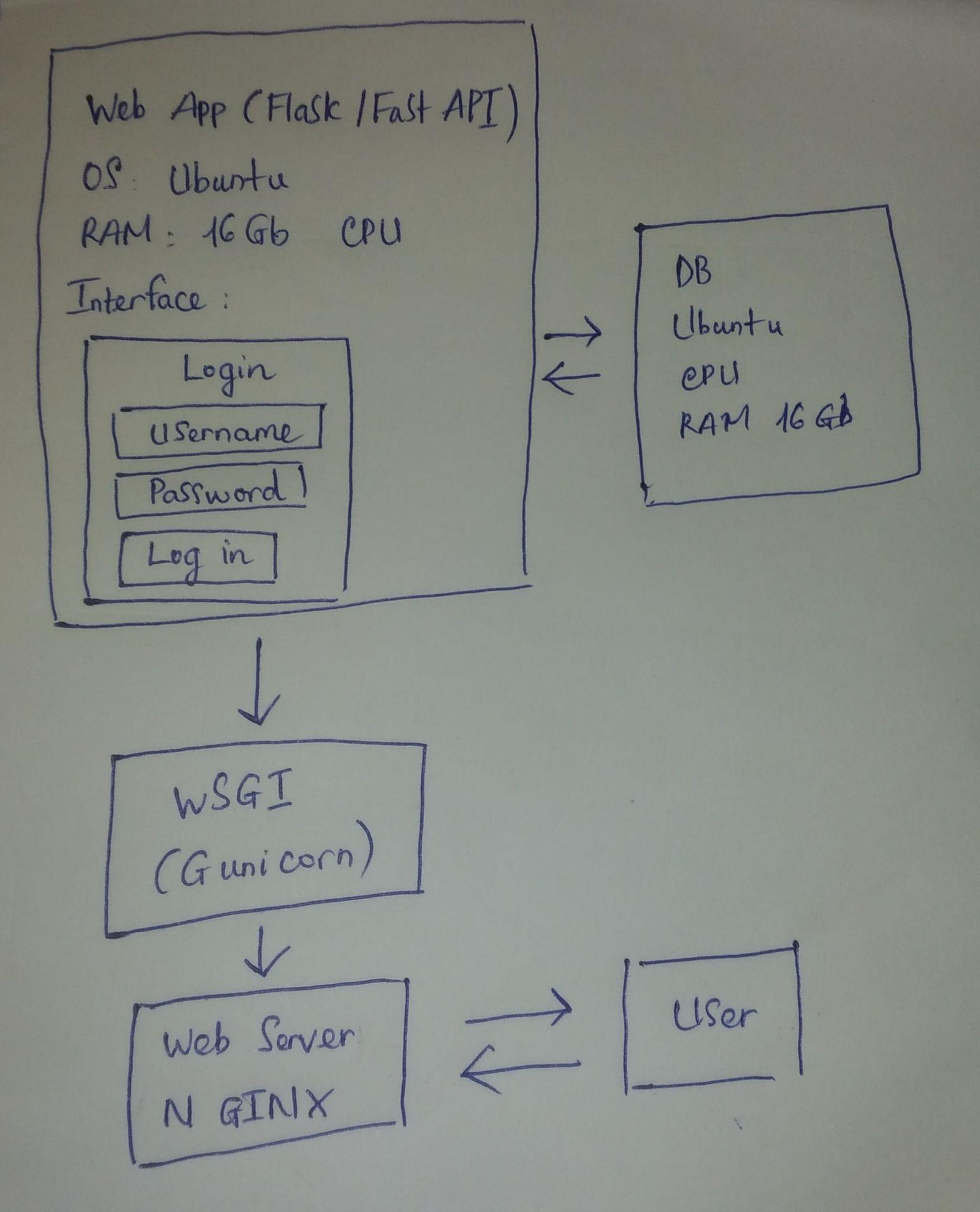
Học viên thực hiện: Nguyễn Thị Ngọc Trâm

Mã số học viên: 21C11036

Khóa: 31 Ngành: Khoa học máy tính

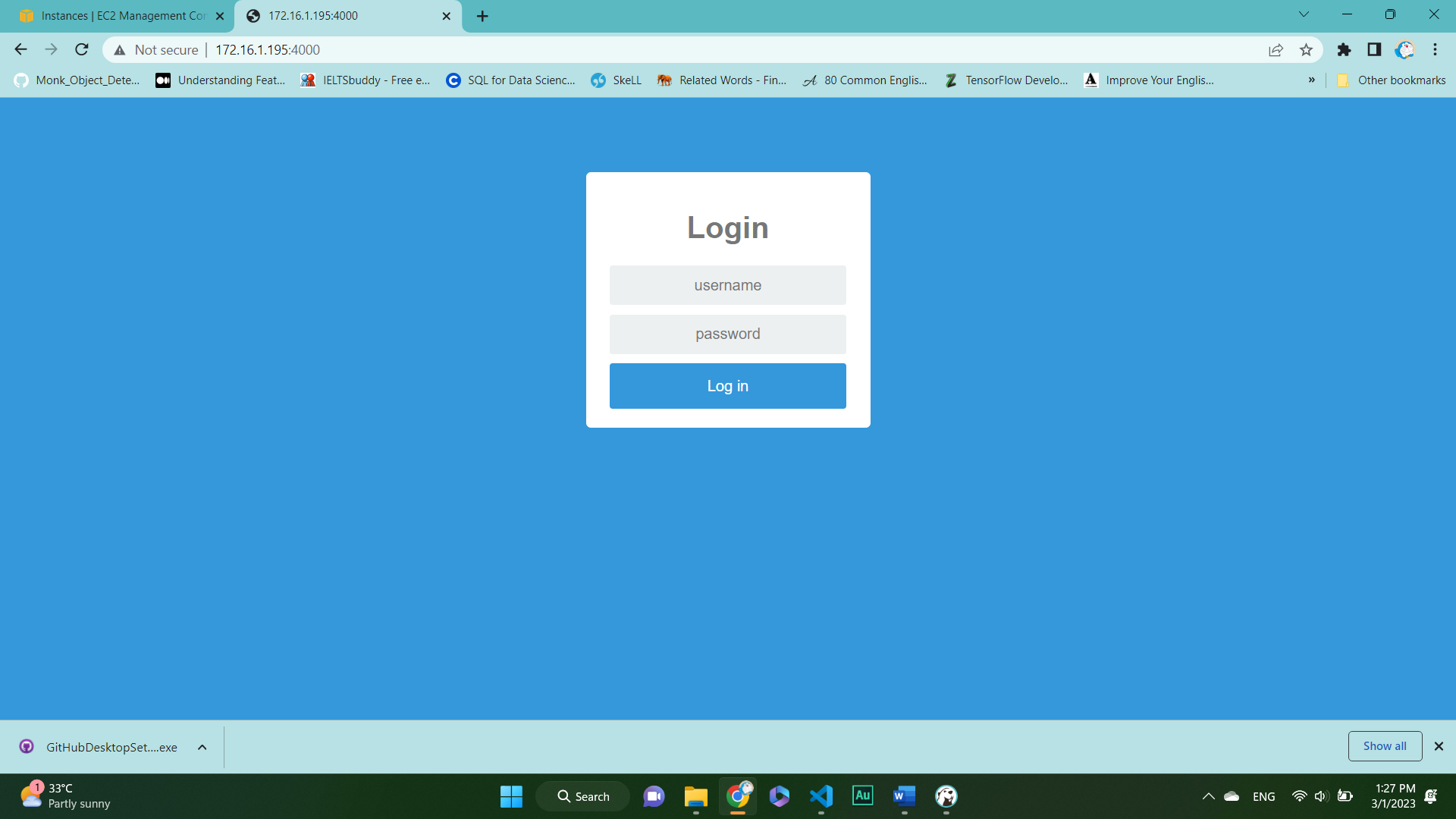
1. **NỘI DUNG BÁO CÁO**
2. **LAB 1: Amazon Web Services Amazon EC2**

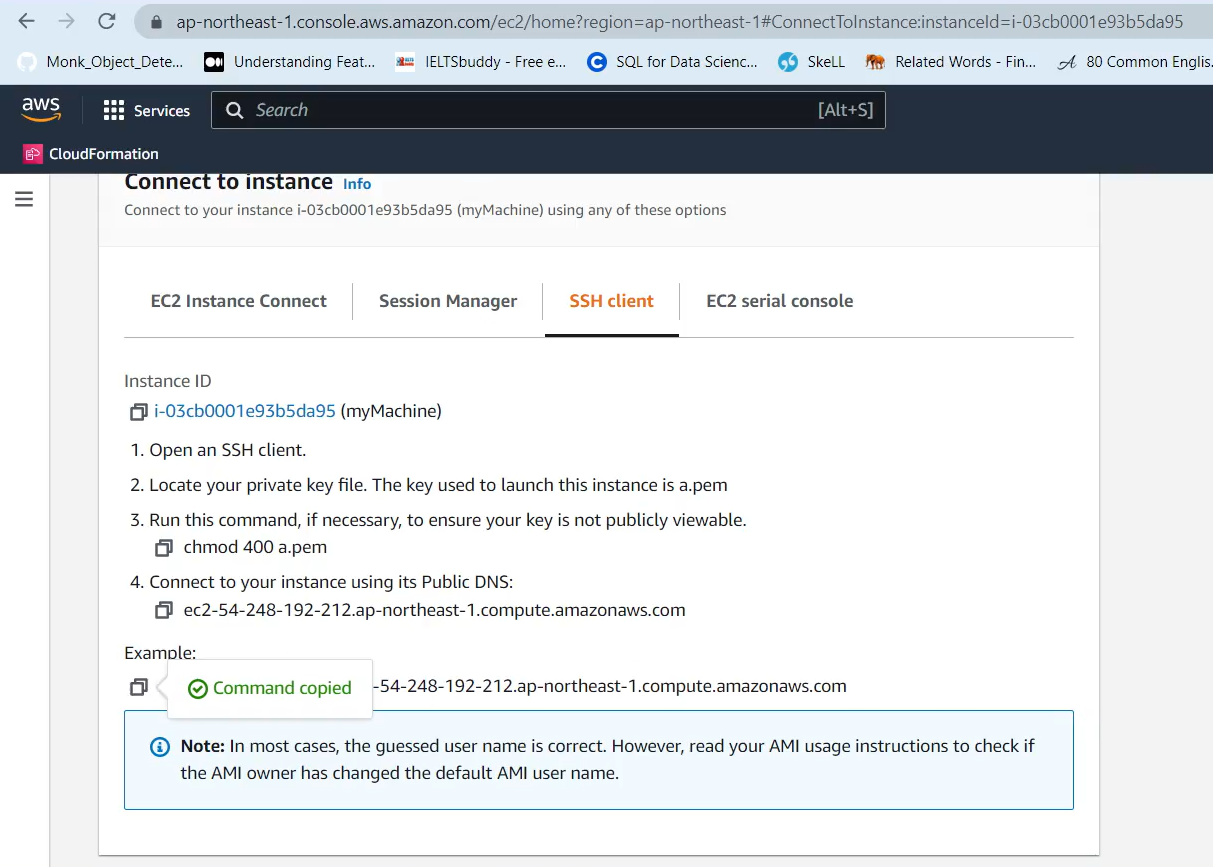
Deployment architecture

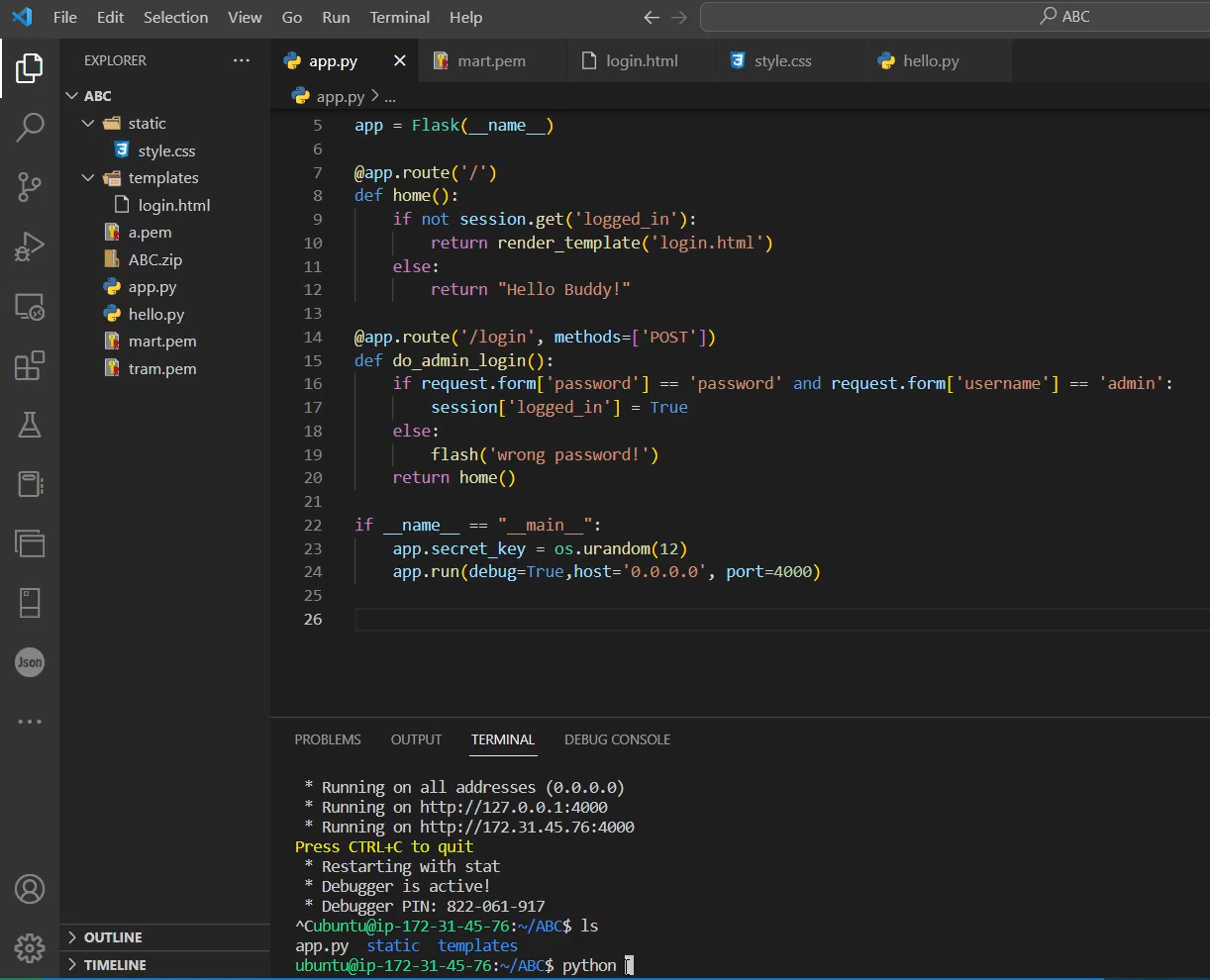


Demo video: <https://drive.google.com/file/d/1Onogbb95GKzzyUTXMHONt4jtKuBp6D3u/view?usp=sharing>

Cloud provider URL

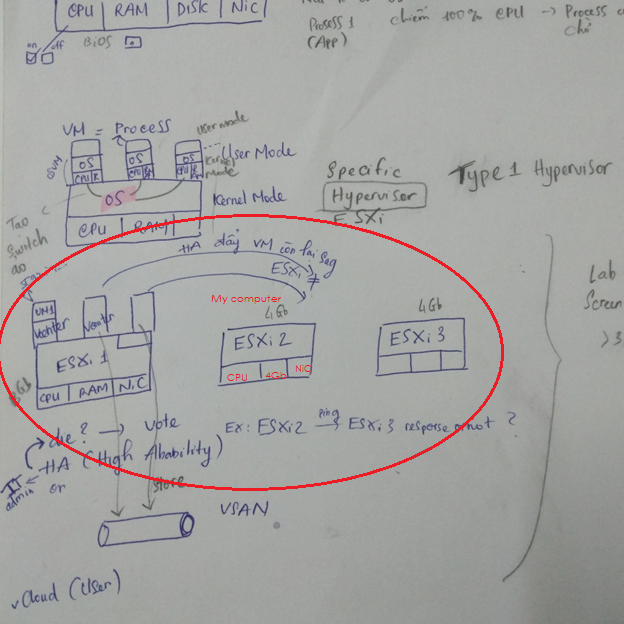




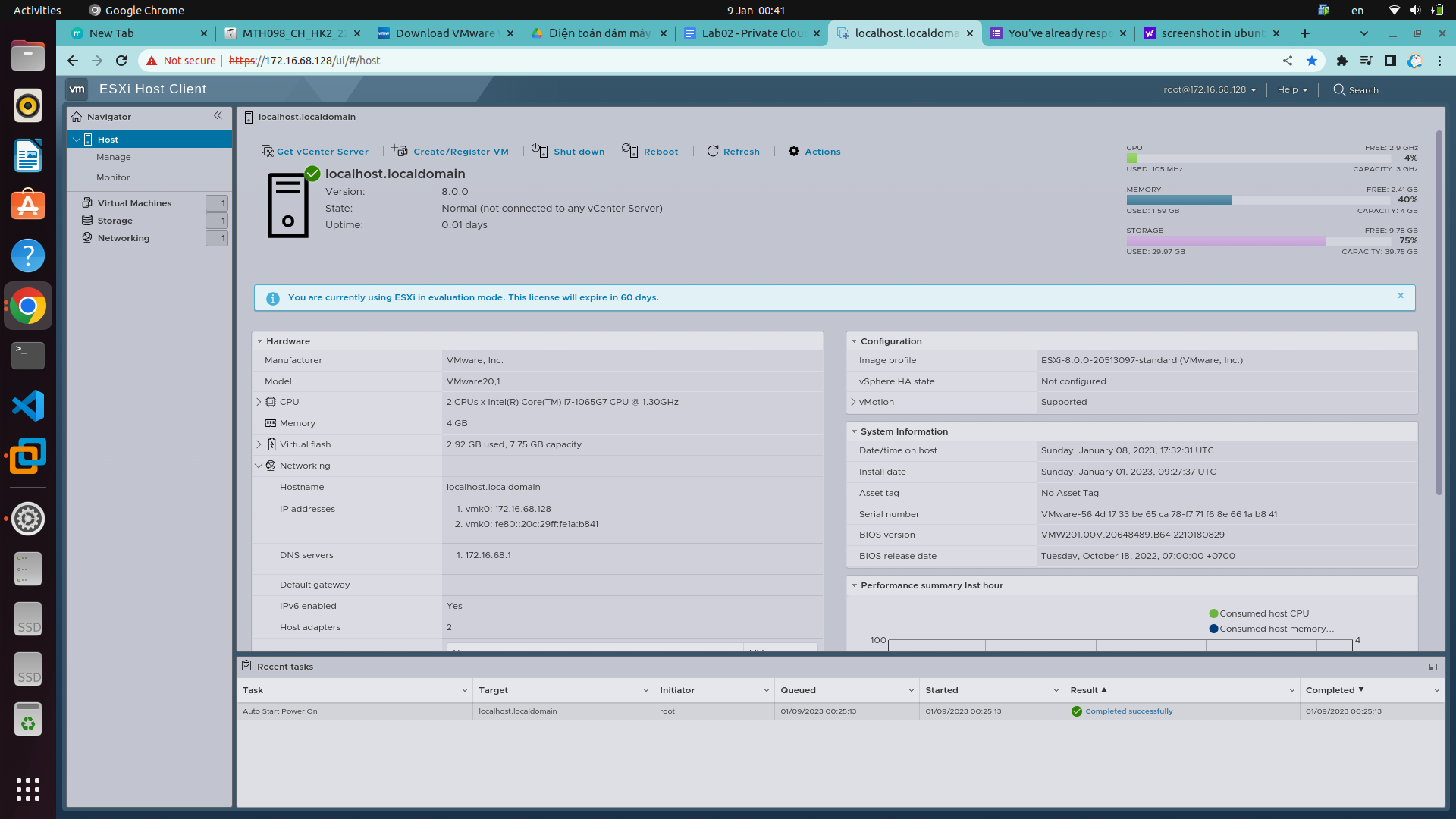
****

1. **LAB 2: Private Cloud**

Datacenter Architecture ESXi, Vcenter



Screenshot



1. **LAB 3: Platform as a Service Lab**

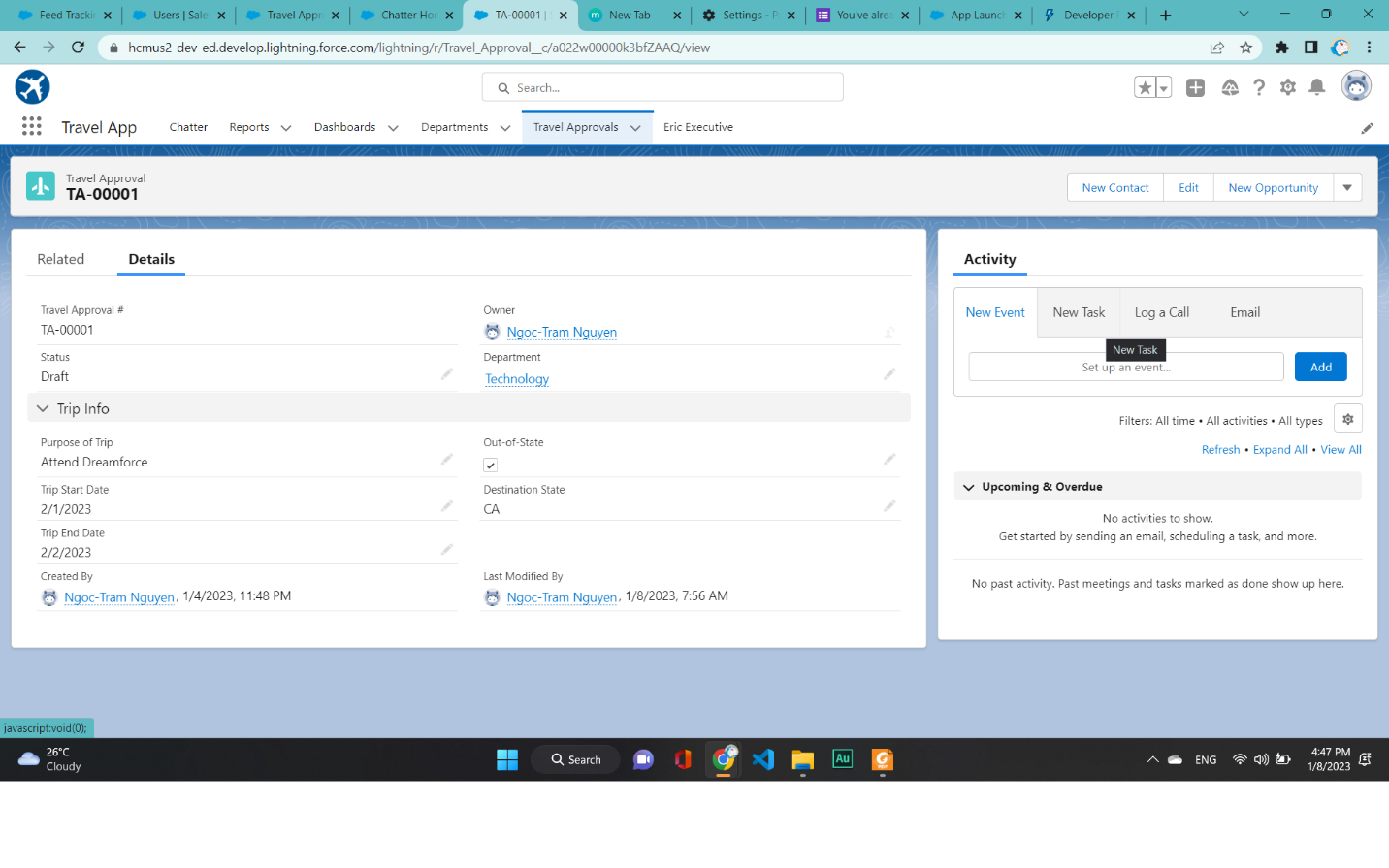
PaaS provider URL: <https://developer.salesforce.com/>

Application URL for admin: <https://hcmus2-dev-ed.develop.lightning.force.com/lightning/r/User/0052w00000Bx24iAAB/view>

Admin email or username: [tramnnt.139@gmail.com](mailto:tramnnt.139@gmail.com)

Admin password: ngocTram1309$

Screenshot of admin UI:



Application URL for user: <https://hcmus2-dev-ed.develop.lightning.force.com/lightning/r/Travel_Approval__c/a022w00000k3bfZAAQ/view>

User Email or username:

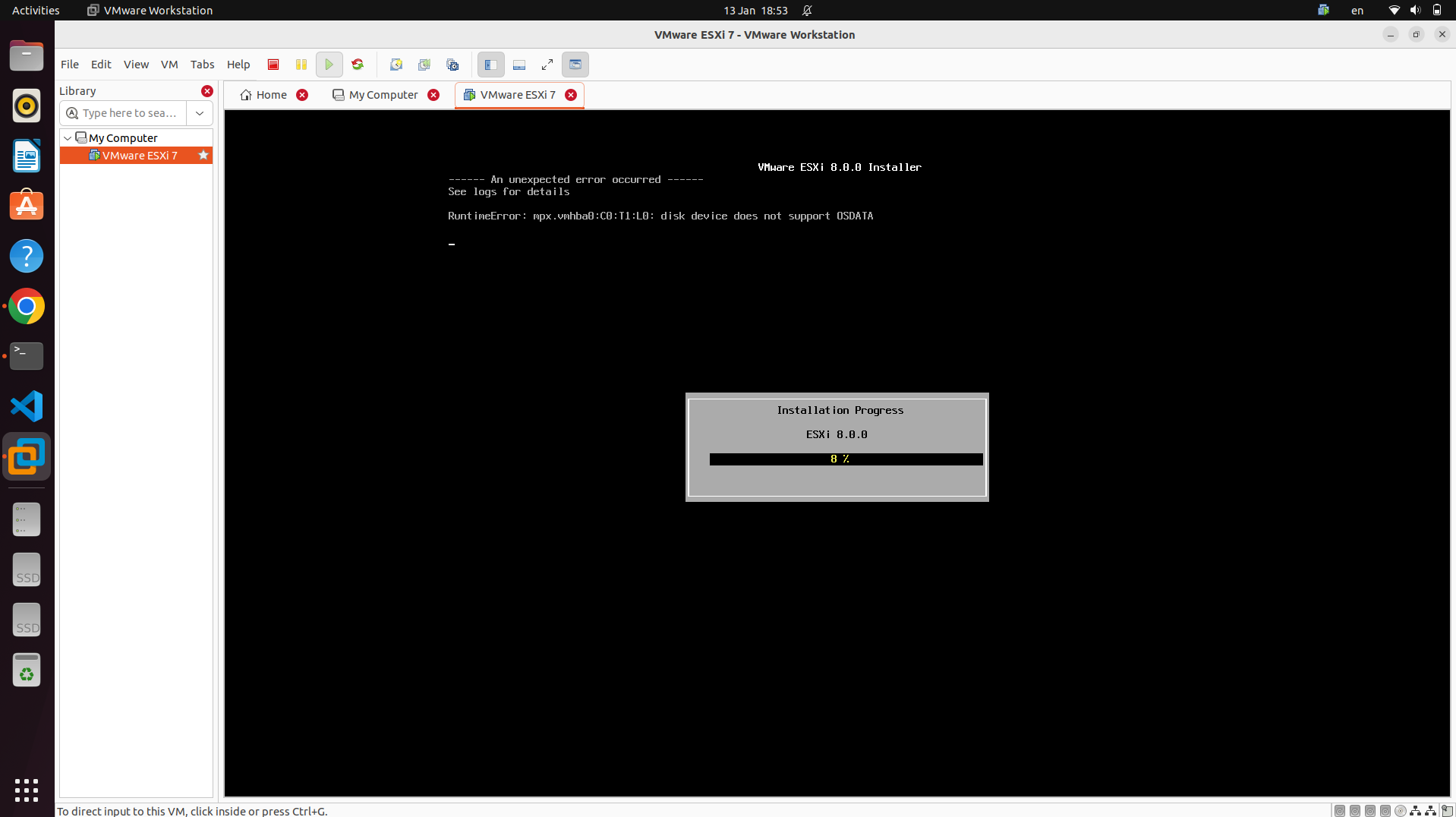
[21c11036@student.hcmus.edu.vn](mailto:21c11036@student.hcmus.edu.vn)

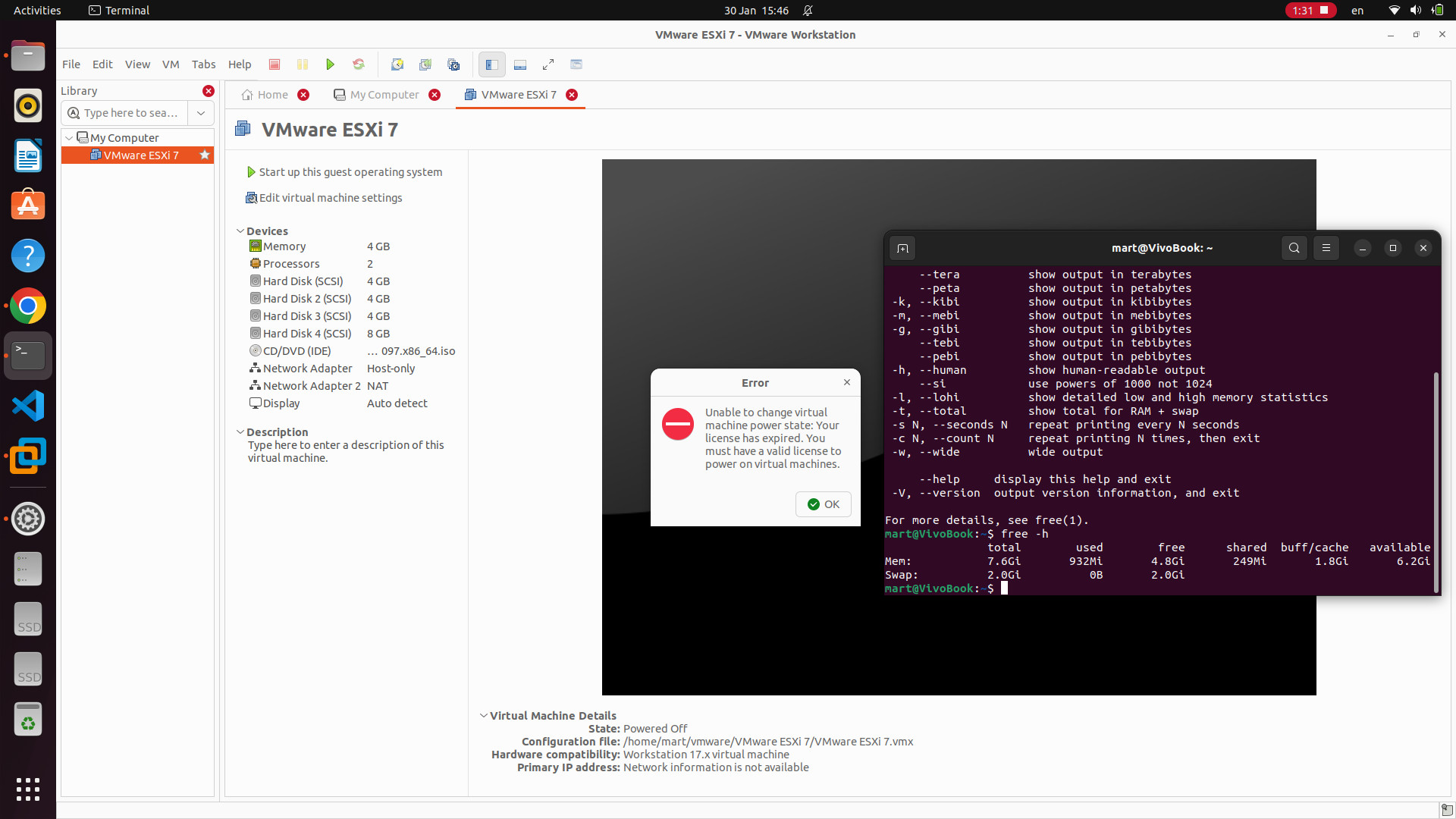
User password: ngocTram13

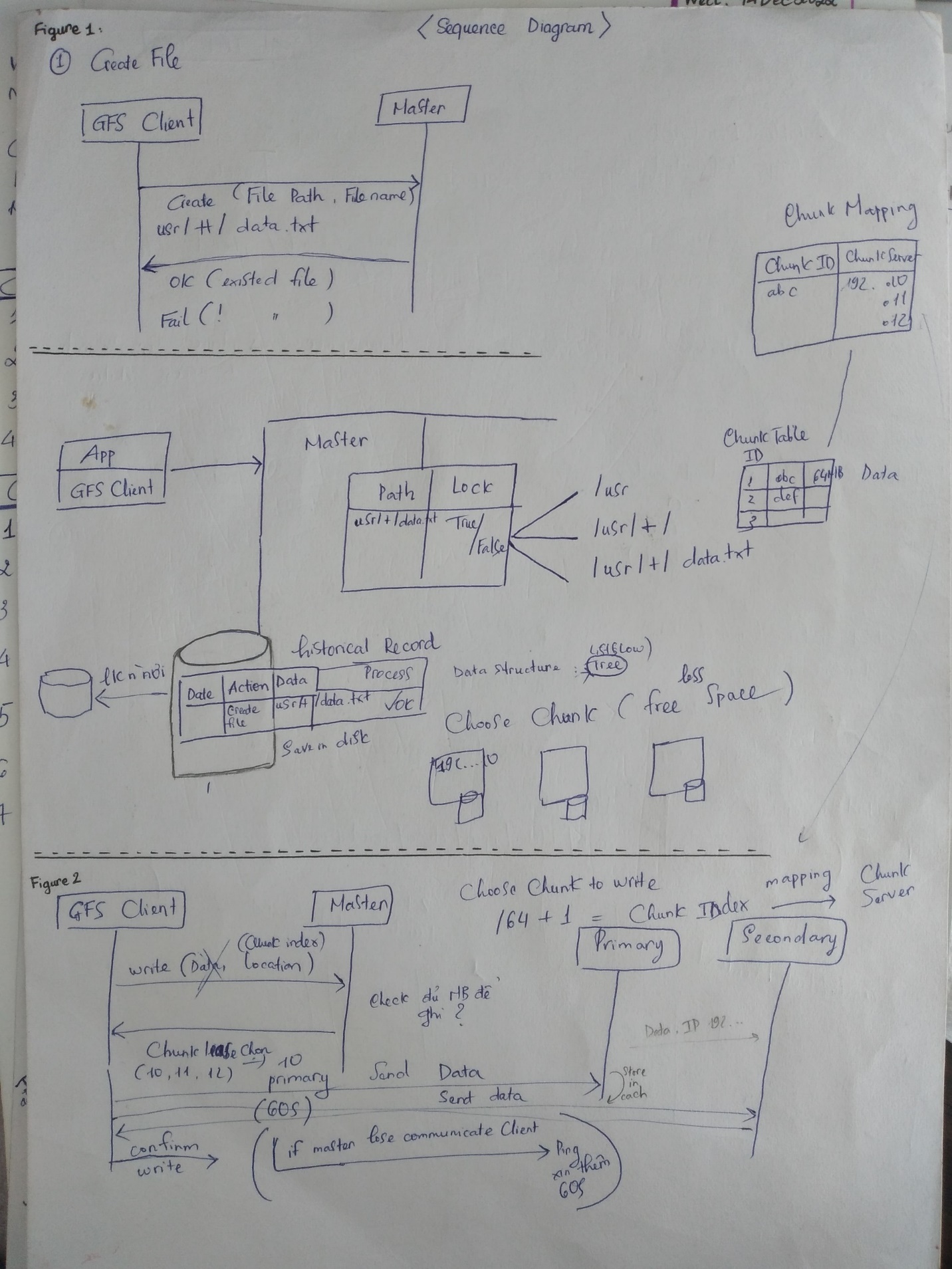
1. **LAB 4: Cloud Computing - Storage in the Cloud: NAS + GFS**

Install NAS

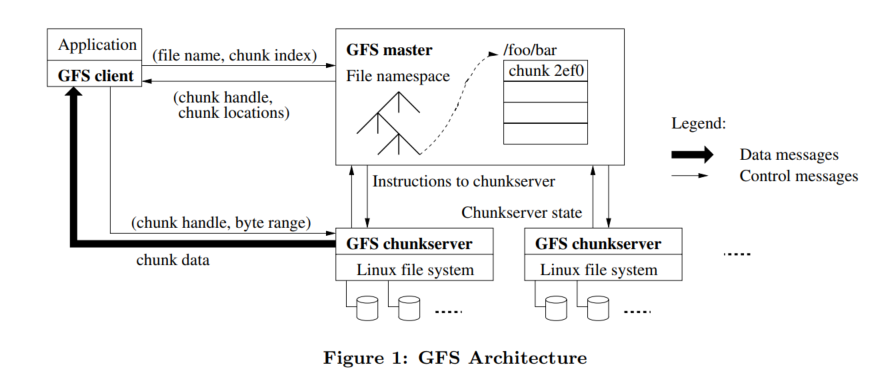
RAM available: 4Gb







Describe a ***GFS read*** using a sequence diagram:



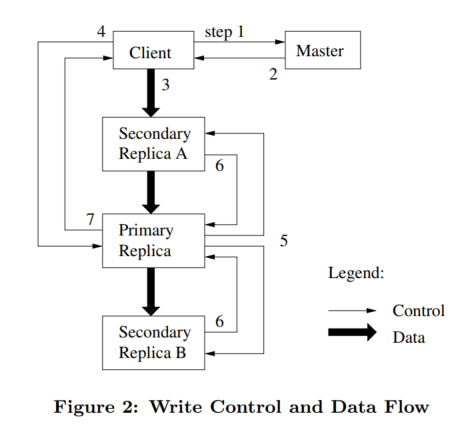
1. The client first communicates with the master, sending it a request containing ***the file name and the chunk index*.** The client derives the chunk index from a combination of the file name and the byte offset the application wants to read from.

2. The master replies with the corresponding ***chunk handle and the location*** of its replicas.

3. The client caches this information using the file name and chunk index as the key.

4. The client then sends a request to one of the replicas specified by the client, usually the one closest to it, specifying the ***chunk handle and the byte range*** for the requested data.

5. By caching the information from the master, further reads to the same chunk do not require any more client-master interactions until the cached information expires or the file is reopened.

Describe a ***GFS write*** using a sequence diagram

1. The client asks the master for all chunk servers.

2. The master grants a new lease to a replica (if none exist), increases the chunk version number, and tells all replicas to do the same after the mutation has been applied. It then replies to the client. After this, the client no longer has to talk to the master.

3. The client pushes the data to all the chunkservers, not necessarily to the primary first. The servers will initially store this data in an internal LRU buffer cache until the data is used.

4. Once the client receives the acknowledgement that this data has been pushed successfully, it sends the write request to the primary chunkserver. The primary decides what serial order to apply the mutations in and applies them to the chunk.

5. After applying the mutations, the primary forwards the write request and the serial number order to all the secondaries for them to apply in the same order.

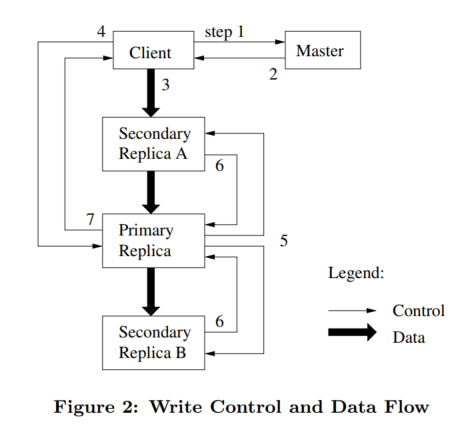
6. All secondaries reply to the primary once they have completed the operation.

7. The primary replies to the client, indicate whether the operation was a success or an error. Note:

+ If the write succeeds at the primary but fails at any of the secondaries, we'll have an inconsistent state and an error is returned to the client.

+ The client can retry steps 3 through 7.

Describe a ***GFS append*** using a sequence diagram



1. The system interactions for record appends are largely the same as discussed for writes, with the following exceptions:

2. In step 4, the primary first checks to see if appending the record to the current chunk would exceed the maximum size of 64MB. If so, the primary pads the chunk, notifies the secondaries to the same, and then tells the client to retry the request on the next chunk.

3. If the record append fails on any of the replicas, the client must retry the operation. As discussed in the Consistency section, this means that replicas of the same chunk may contain duplicates.

4. A record append is successful only when the data has been written at the same offset on all the replicas of a chunk.

<https://timilearning.com/posts/mit-6.824/lecture-3-gfs/#single-master>

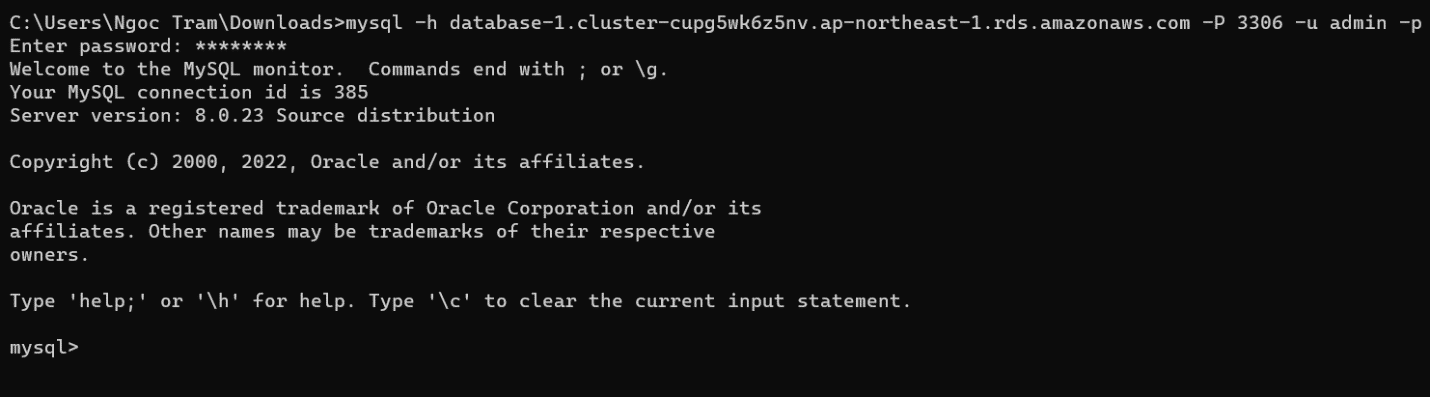
<https://www.linkedin.com/pulse/gfs-google-file-system-amit-kumar/>

<https://computer.howstuffworks.com/internet/basics/google-file-system.htm>

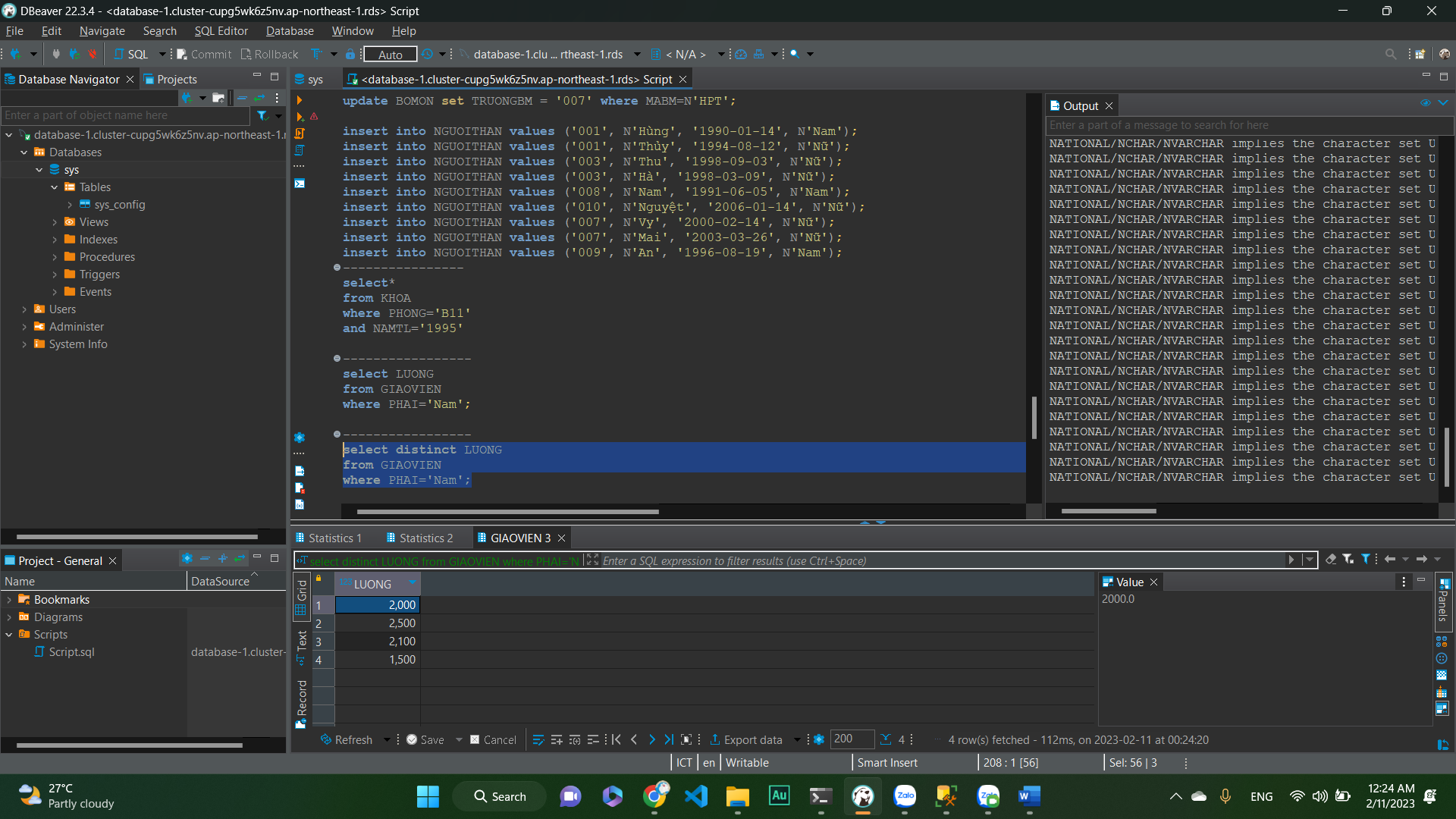
<https://cs.stanford.edu/~matei/courses/2015/6.S897/slides/gfs.pdf>

1. **LAB 5: Cloud Computing - Amazon Aurora**

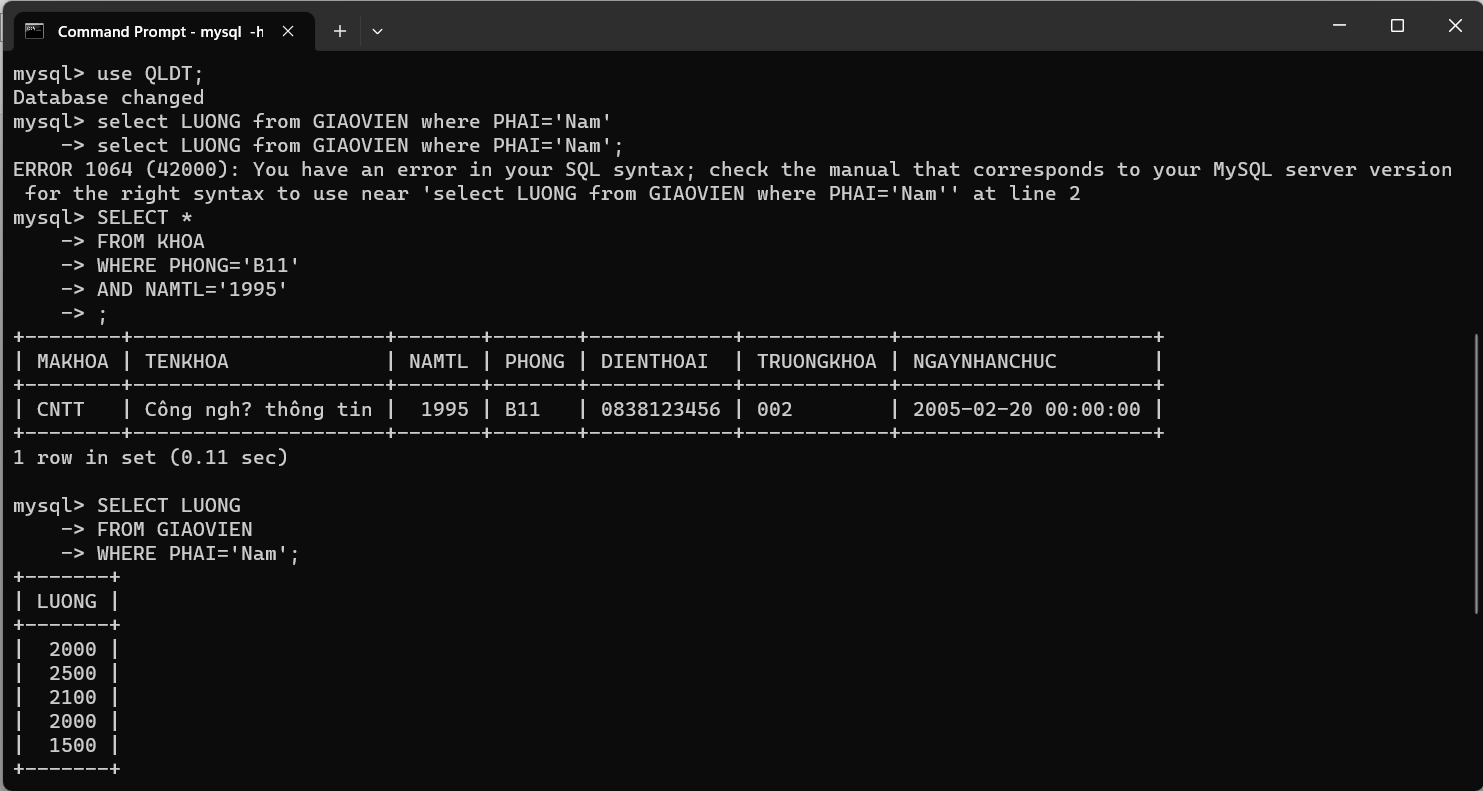
Connect to Aurora DB

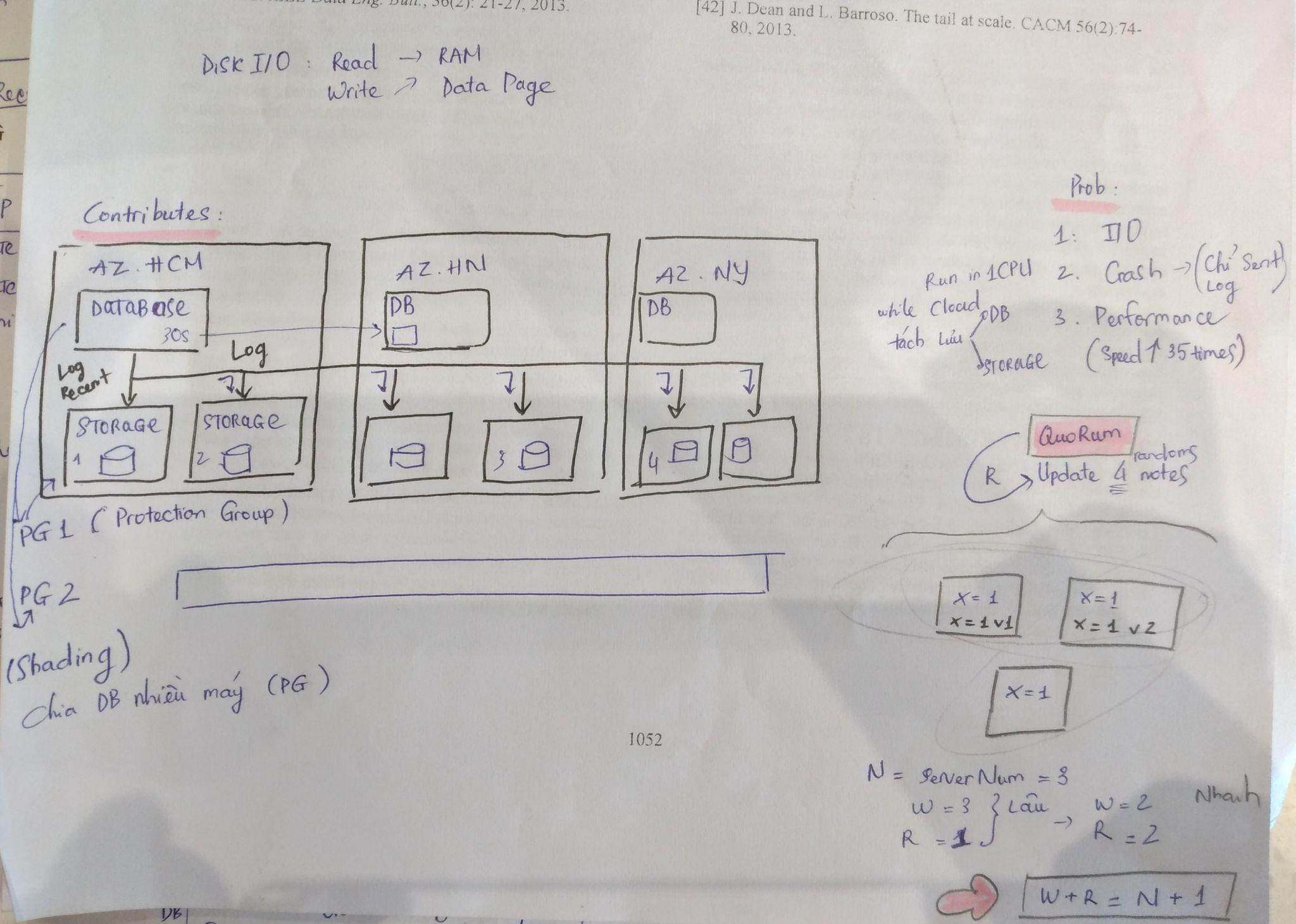


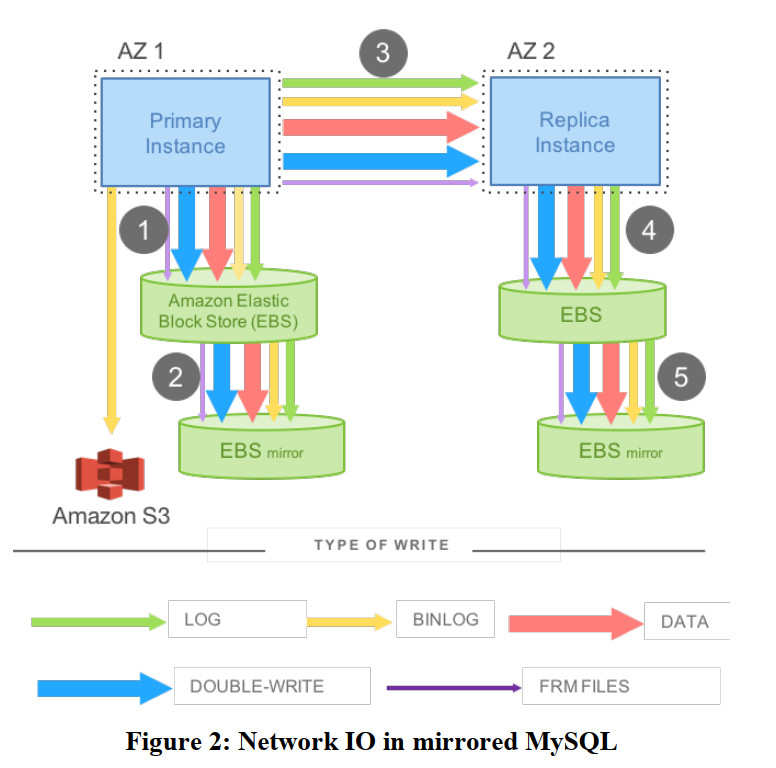
Use DBeaver to connect and create DB in Aurora DB



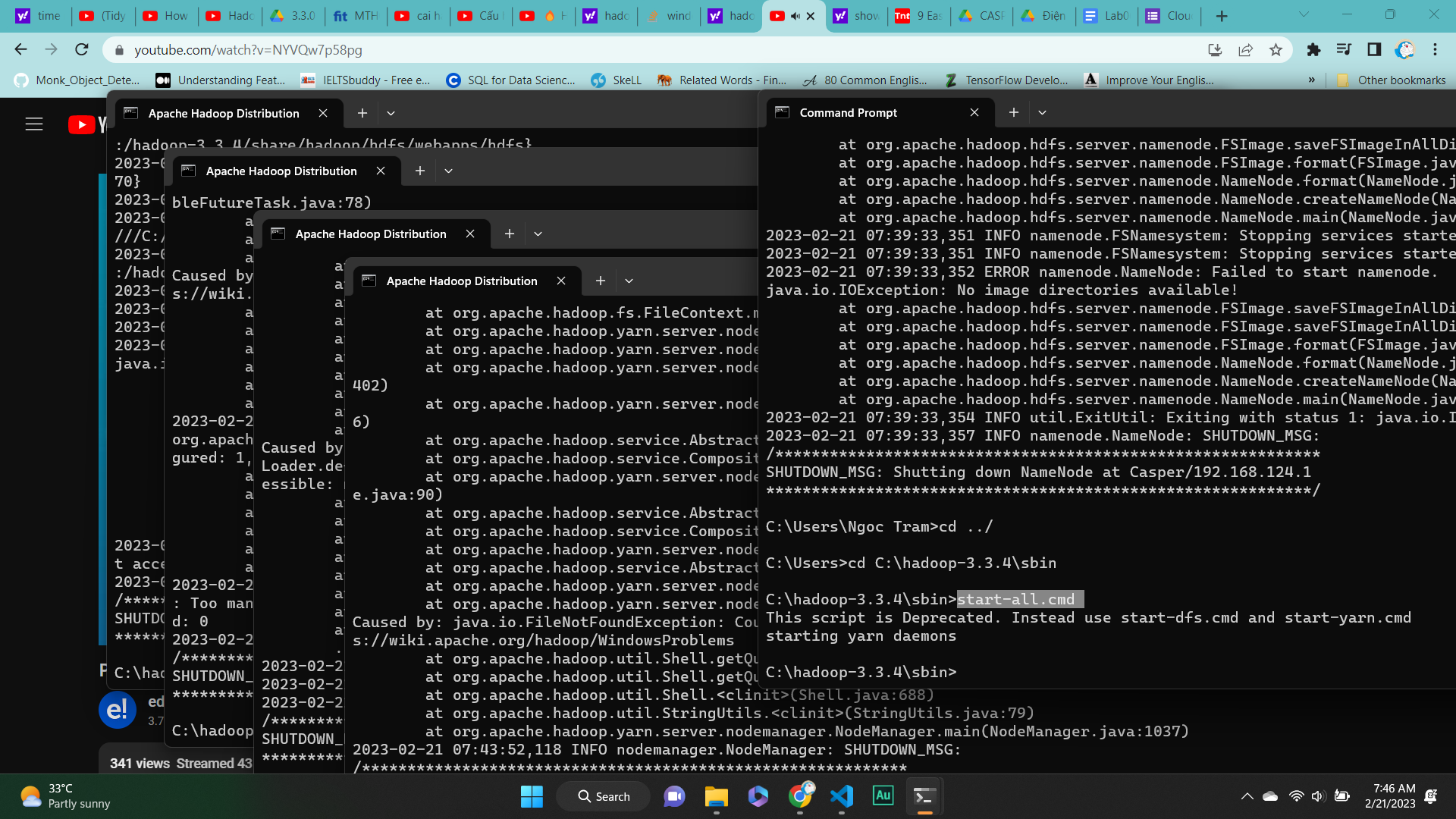
Query DB

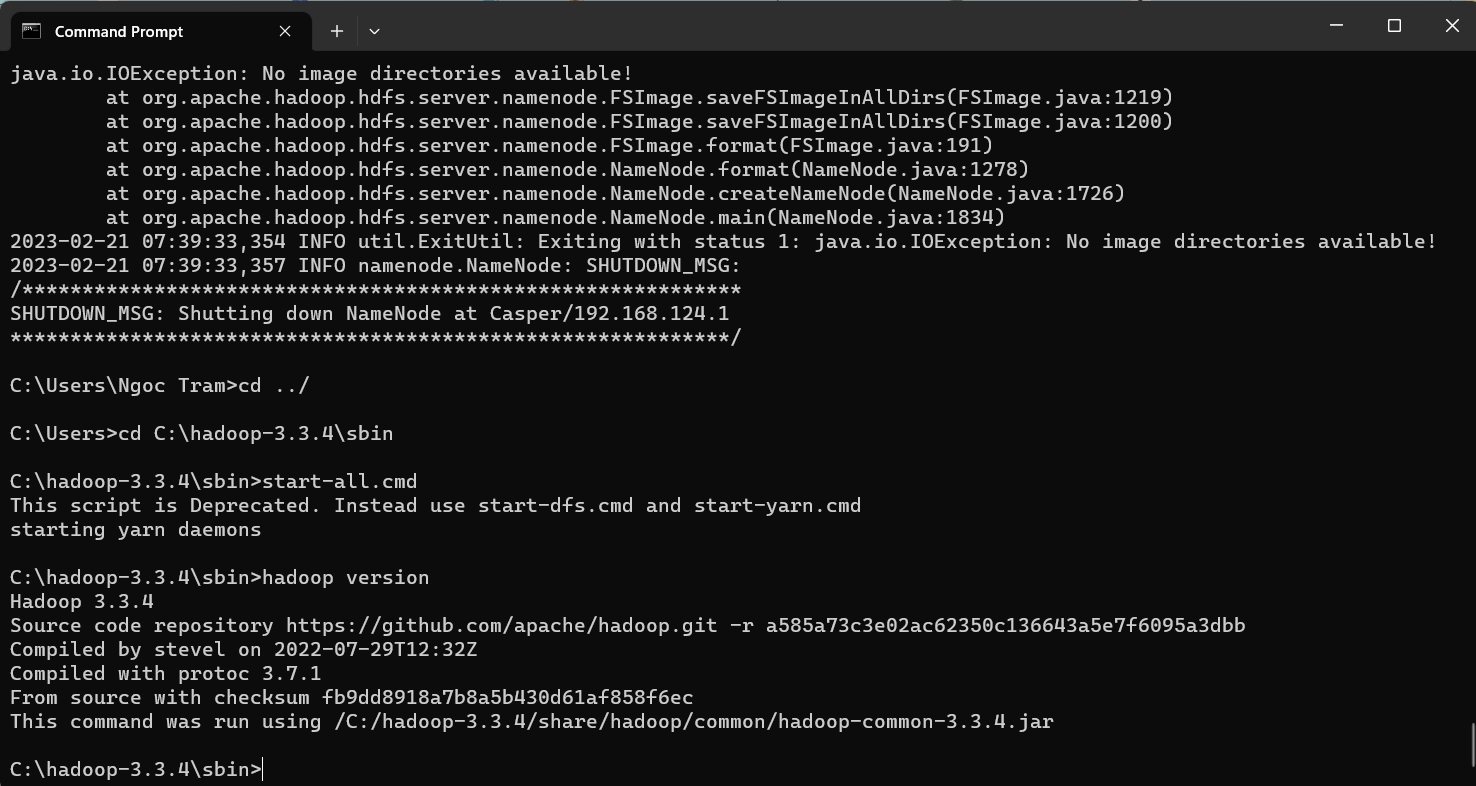


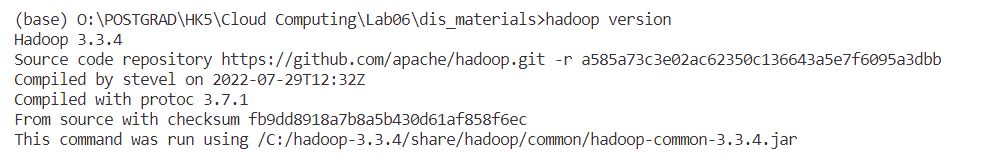
Explain how quorums work.  


Describe how replication writes work in a mirrored MySQL database using a sequence diagram.  


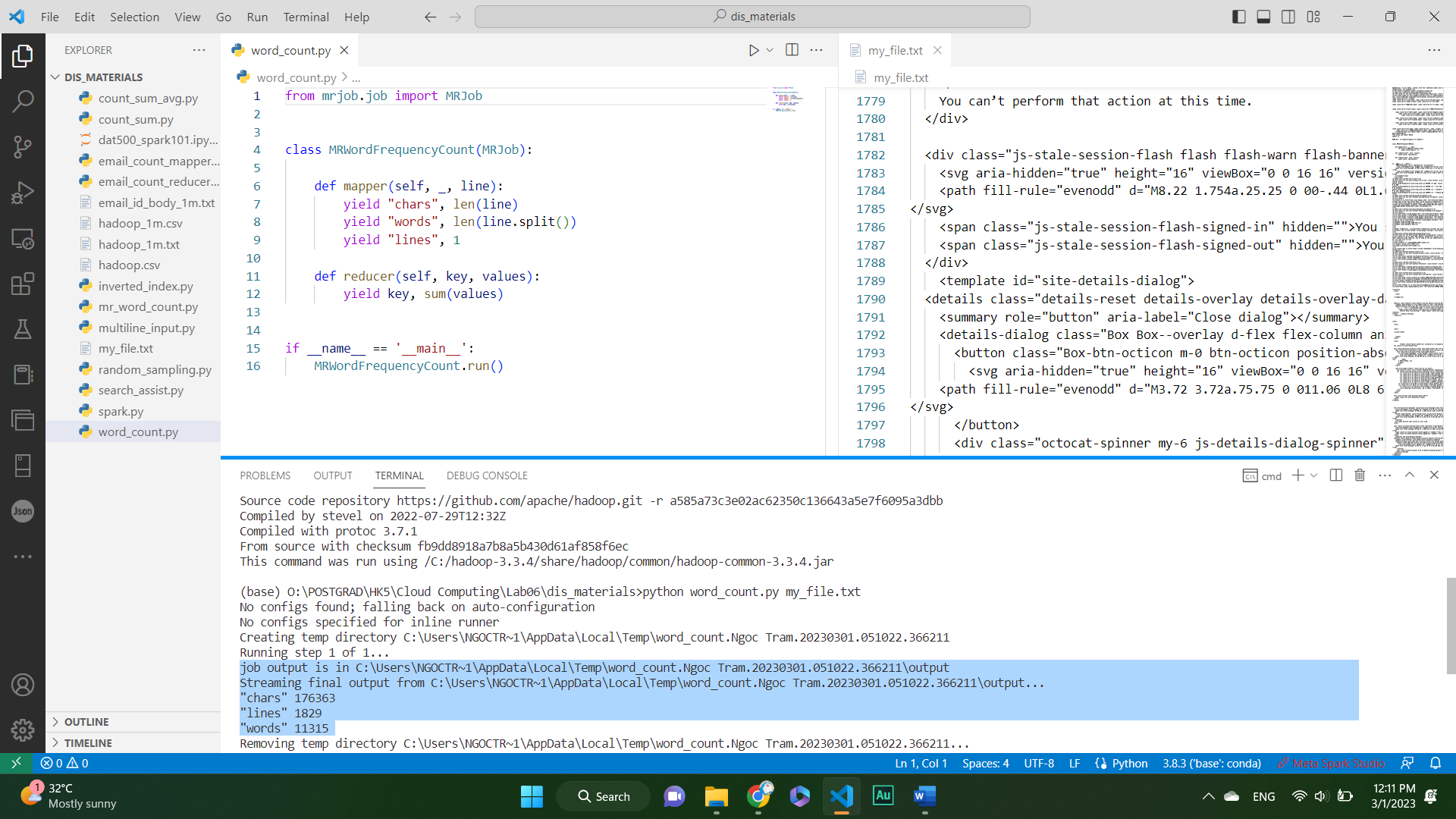
1. **LAB 6: MapReduce**



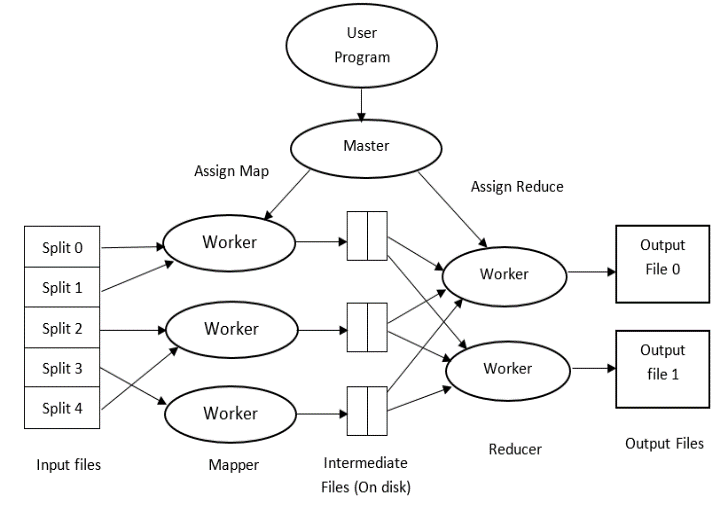


WordCount + myFile

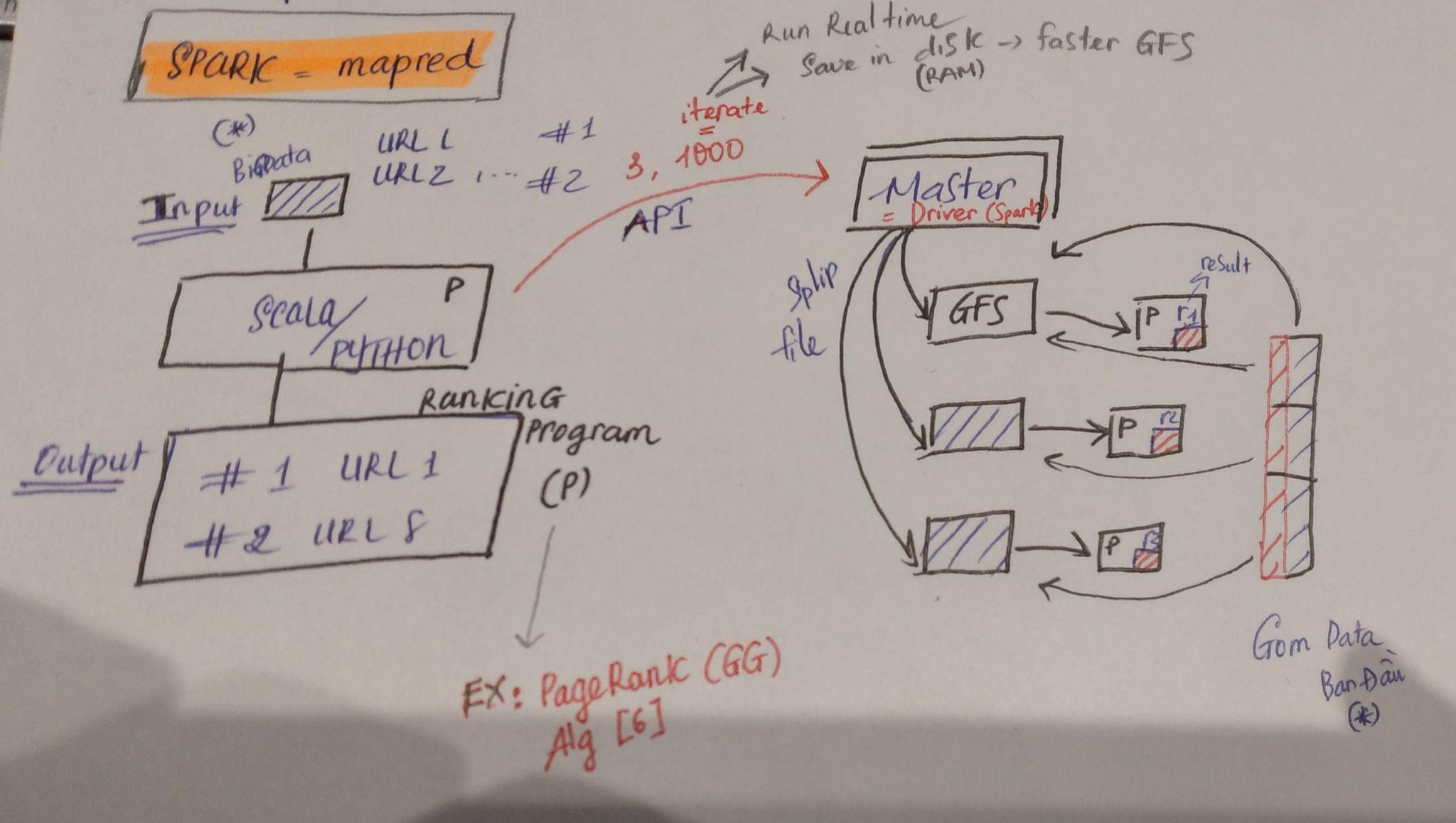
Screenshot captured after running your code.



Describe how a MapReduce program runs on 6 machines (1 Master, 3 Mappers, 2 Reducers) using a sequence diagram.



1. **LAB 7: Cloud Computing (Spark)**



Run SparkPageRank:

