

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

BỘ MÔN HỆ THỐNG THÔNG TIN



ĐỀ TÀI CUỐI KỲ MÔN NHẬP MÔN DỮ LIỆU LỚN

TÊN ĐỀ TÀI:

Kết Nối Cụm Hadoop Nhiều Node, Tìm hiểu CSDL NoSQL. Tìm hiểu Và Cài Đặt Riak, Demo Ứng Dụng Của Riak.

GVHD: *ThS. Lê Thị Minh Châu*

Lớp HP: BDES333877_23_2_02

Nhóm thực hiện: Nhóm 14

Học kỳ: 2

Nă̄m h̄oc: 2023 – 2024

Thành phố Hồ Chí Minh, tháng 5, năm 2024

DANH SÁCH SINH VIÊN NHÓM THỰC HIỆN

HỌC KÌ 2 NĂM HỌC 2023 - 2024

Nhóm 2

Đề tài: Tìm hiểu CSDL NoSQL. Tìm hiểu và cài đặt Riak, demo ứng dụng của Riak.

Bảng Phân Công Nhiệm Vụ:

MSSV	Tên Sinh Viên	Nhiệm Vụ
21110942	Phùn Khoản Võ	Tìm Hiểu, Cấu hình Hadoop, viết báo cáo ưu và nhược điểm của NOSQL,Các mô hình dữ liệu trong Riak, cấu hình Riak, Làm thao tác update dữ liệu trong thao tác cơ bản Riak, Viết code Python Sửa Value tương ứng với 1 key trong Bucket.
21110873	Nguyễn Anh Dũng	Tìm Hiểu, viết báo cáo về So sánh NOSQL và SQL, Ứng Dụng của Riak, Cài đặt Riak , tham gia làm thao tác Add database, kết nối cluster 2 máy riak, code python thêm dữ liệu vào riak, ShowData trong 1 bucket trong riak.
21110731	Phạm Quốc Vương	Tìm Hiểu cấu hình SSH, viết báo cáo khái niệm NOSQL,Kiến trúc và các chế độ thực thi của Riak, làm Thao tác cơ bản Riak Insert một Key-Value, Viết code Python Riak Tìm Kiếm 1 Value tương ứng với 1 bucket trong demo ứng dụng nâng cao
21110528	Nguyễn Đình Liệu	Tìm Hiểu, Viết Báo cáo về Phân loại, cách hoạt động NoSQL, Giới Thiệu về Riak, tham gia delete key, tìm kiếm 1 Key-Value trong thao tác cơ bản với Riak, viết code Python thêm 1 Key-Value vào 1 bucket.

Mục lục

Bảng Phân Công Nhiệm Vụ:	1
Chương 1. Kết Nối Cụm Hadoop Nhiều Node	4
1.1. Cài Đặt và Cấu Hình SSH.....	4
1.2. Cài Hadoop trên Windows.....	9
Chương 2. GIỚI THIỆU NOSQL.....	23
2.1. Khái niệm:	23
2.2. Cách hoạt động.....	24
2.3. Phân loại NoSQL.....	24
2.4. Ưu điểm và nhược điểm của NoSQL.....	25
2.4.1. Ưu điểm:	25
2.4.2. Nhược điểm:.....	26
2.5. So sánh Nosql với SQL:	27
Chương 3. CƠ SỞ DỮ LIỆU RIAK.....	28
3.1. Giới thiệu:	28
3.1.1. Đặc điểm chính của Riak trong NoSQL:	29
3.1.2. Ưu điểm	29
3.1.3. Nhược điểm:.....	30
3.2. Các mô hình dữ liệu trong Riak:	31
3.2.1. Key-Value (Khóa-Giá trị):	31
3.2.2. Document (Tài liệu):	31
3.2.3. Counter (Bộ đếm):	31
3.2.4. Set (Tập hợp) và Map (Bản đồ):	31
3.2.5. HyperLogLog (HLL):	32
3.3. Kiến trúc của Riak.....	32
3.3.1. Cụm Riak (Riak Cluster):.....	32
3.3.2. Nút Riak (Riak Node):	32
3.3.3. Dynamo-style Ring	32
3.3.4. Phân phối dữ liệu và Replication:	32
3.3.5. Quorum-based Consensus:.....	33
3.3.6. Riak Core:.....	33
3.4. Chế độ thực thi của riak.....	33
3.4.1. Single-Node Mode (Chế độ Một nút):	33
3.4.2. Multi-Node Mode (Chế độ Nhiều nút):	33
3.4.3. Active-Anti-Entropy (AAE):	33
3.4.4. Hinted Handoff:	33
3.4.5. Read Repair và Anti-Entropy Repair:	34

3.5. Ứng dụng của Riak	34
3.5.1. Ứng dụng Web và Mobile:	34
3.5.2. Phân tích dữ liệu lớn (Big Data):	34
3.5.3. Dịch vụ Lưu trữ Tập:	34
3.5.4. Game trực tuyến và ứng dụng Real-Time:.....	34
3.5.5. Ứng dụng IoT (Internet of Things):	35
Chương 4. Cài Đặt, Cấu Hình và Thao Tác Cơ bản Riak	35
4.1. Cài Đặt Riak.....	35
4.2. Cấu Hình Riak	37
4.3. Thao tác cơ bản với Riak	39
4.3.1. Add database.....	39
4.3.2. Insert 1 key -value	41
4.3.3. Tìm kiếm 1 key value trong 1 bucket	41
4.3.4. Update dữ liệu	42
4.3.5. Delete key:.....	42
Chương 5. Demo Ứng Dụng Riak Nâng Cao	43
5.1. Kết nối Cluster-Riak 2 máy:	43
5.2. Demo xử lý dữ liệu 2 máy	44
5.2.1. Ghi dữ liệu.....	44
5.2.2. ShowData trong 1 bucket	47
5.2.3. Tìm kiếm 1 value tương ứng với 1 key trong 1 bucket	49
5.2.4. Thêm key value vào 1 bucket.....	50
5.2.5. Sửa value tương ứng với 1 key trong 1 bucket.....	52
Tài Liệu Tham Khảo	54

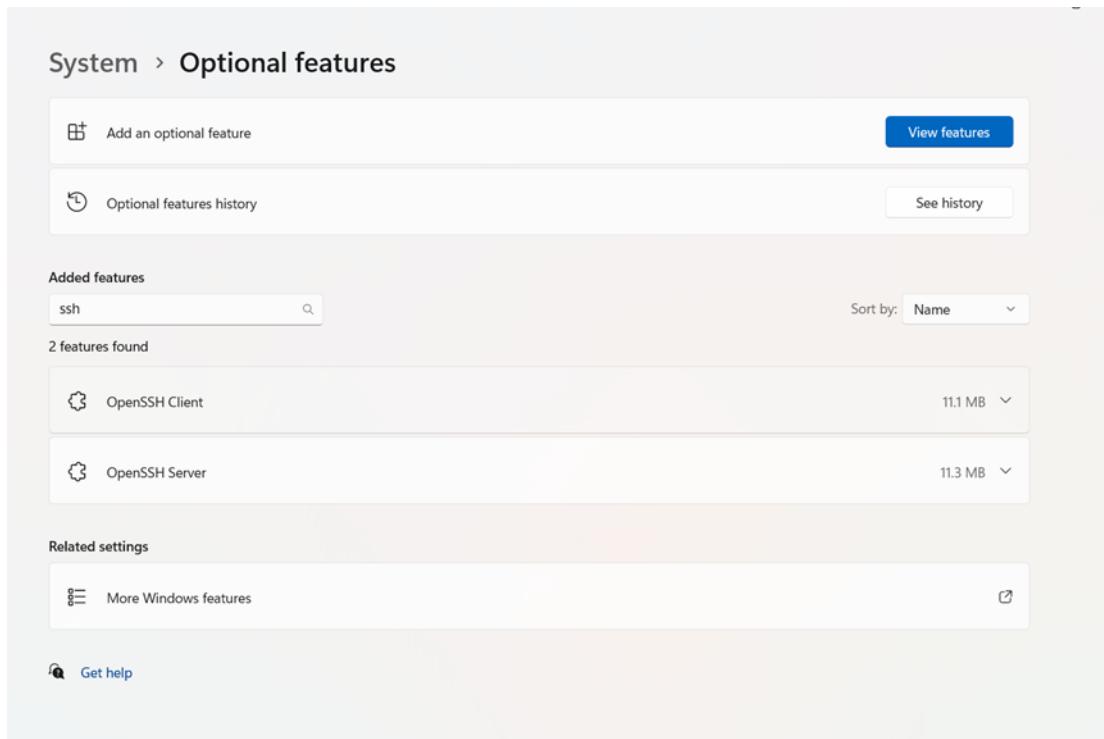
Đề Tài: Kết Nối Cụm Hadoop Nhiều Node, Tìm hiểu CSDL NoSQL. Tìm hiểu Và Cài Đặt Riak, Demo Ứng Dụng Của Riak.

Chương 1. Kết Nối Cụm Hadoop Nhiều Node

1.1. Cài Đặt và Cấu Hình SSH

Cài đặt SSH Client và SSH Server

Cài đặt trên tất cả các máy bằng cách mở “Optional Feature” rồi tìm và kích hoạt



Tạo key-gen

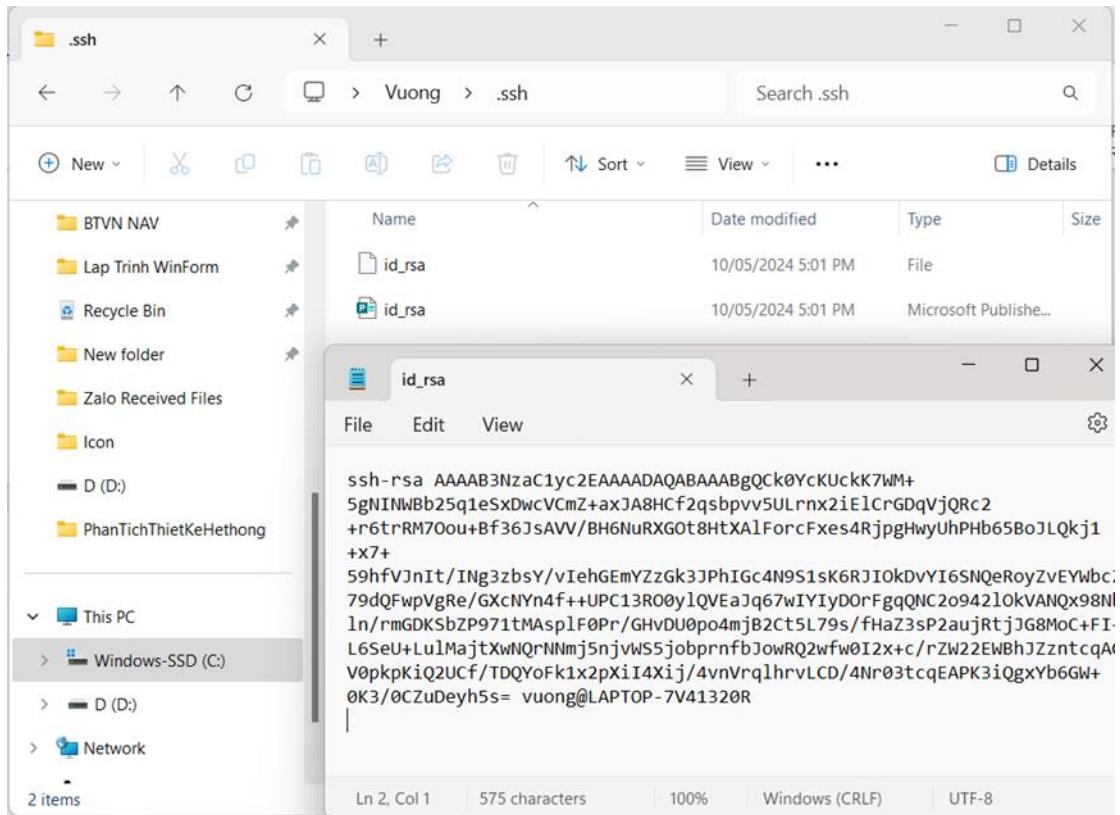
```

Command Prompt
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Vuong>ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\Vuong\.ssh/id_rsa):
C:\Users\Vuong\.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Vuong\.ssh/id_rsa
Your public key has been saved in C:\Users\Vuong\.ssh/id_rsa.pub
The key fingerprint is:
SHA256:BanECPT0hnIJCrljimI5L+MllID5QyQsZ0frHZuacLfc vuong@LAPTOP-7V41320R
The key's randomart image is:
+---[RSA 3072]---+
|.. .
|o+ . ..
|+.+
|..+
|..oo o oS
|*.o + .
|.* Oo* +
|**o** B o
|O++ = oE
+---[SHA256]---+
C:\Users\Vuong>

```

Copy id_rsa qua authorized_keys



Đặt file authorized_keys qua các máy slave tại đường dẫn tương tự

Cấu hình file hosts

```
C: > Windows > System32 > drivers > etc > hosts
11  # Additionally, comments (such as these) may be inserted on individual
12  # lines or following the machine name denoted by a '#' symbol.
13  #
14  # For example:
15  #
16  #      102.54.94.97      rhino.acme.com      # source server
17  #      38.25.63.10      x.acme.com          # x client host
18
19  # localhost name resolution is handled within DNS itself.
20  #      127.0.0.1      localhost
21  #      ::1            localhost
22  # Added by Docker Desktop
23  192.168.1.7 host.docker.internal
24  192.168.1.7 gateway.docker.internal
25  # To allow the same kube context to work on the host and the container:
26  127.0.0.1 kubernetes.docker.internal
27  # End of section
28  192.168.1.127 vuong-master
29  192.168.1.229 vo-slave
30  192.168.1.184 dung-slave
31  192.168.1.185 lieu-slave
```

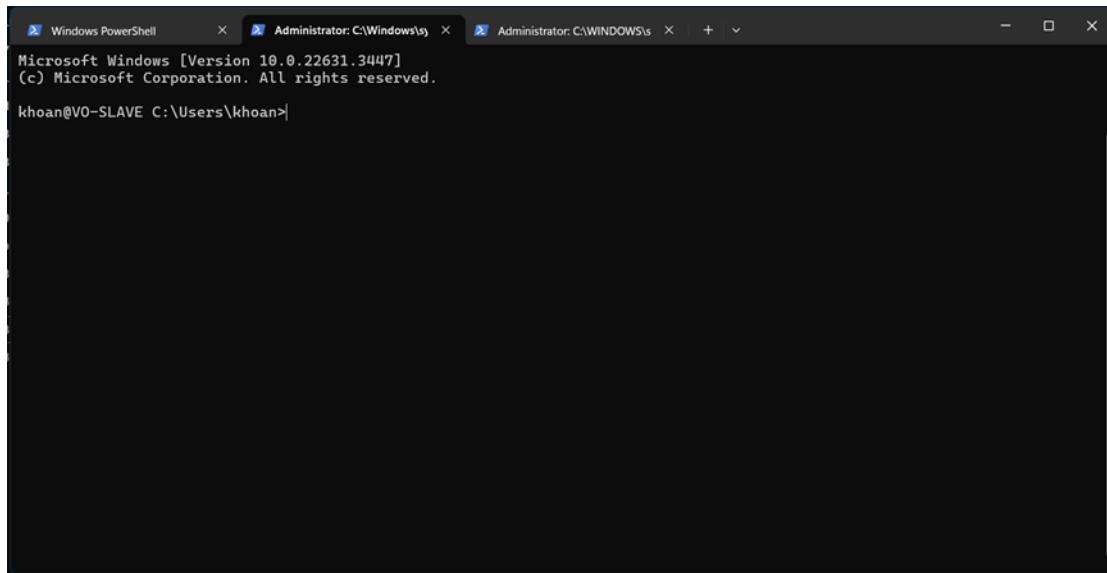
Cấu hình file sshd_config

```
C:\>ProgramData>ssh> $ sshd_config
29    #PermitRootLogin prohibit-password
30    #StrictModes yes
31    #MaxAuthTries 6
32    #MaxSessions 10
33
34    PubkeyAuthentication yes
35
36    # The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
37    # but this is overridden so installations will only check .ssh/authorized_keys
38    AuthorizedKeysFile .ssh/authorized_keys
39
40    #AuthorizedPrincipalsFile none
41
42    # For this to work you will also need host keys in %programData%/ssh/ssh_known_hosts
43    #HostbasedAuthentication no
44    # Change to yes if you don't trust ~/.ssh/known_hosts for
45    # HostbasedAuthentication
46    #IgnoreUserKnownHosts no
47    # Don't read the user's ~/.rhosts and ~/.shosts files
48    #IgnoreRhosts yes
49
50    # To disable tunneled clear text passwords, change to no here!
51    PasswordAuthentication no
52    #PermitEmptyPasswords no
```

```
69    #PidFile /var/run/sshd.pid
70    #MaxStartups 10:30:100
71    #PermitTunnel no
72    #ChrootDirectory none
73    #VersionAddendum none
74
75    # no default banner path
76    #Banner none
77
78    # override default of no subsystems
79    Subsystem sftp sftp-server.exe
80
81    # Example of overriding settings on a per-user basis
82    #Match User anoncvs
83    #    AllowTcpForwarding no
84    #    PermitTTY no
85    #    ForceCommand cvs server
86
87    Match Group administrators
88    #        AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authorized_keys
89
```

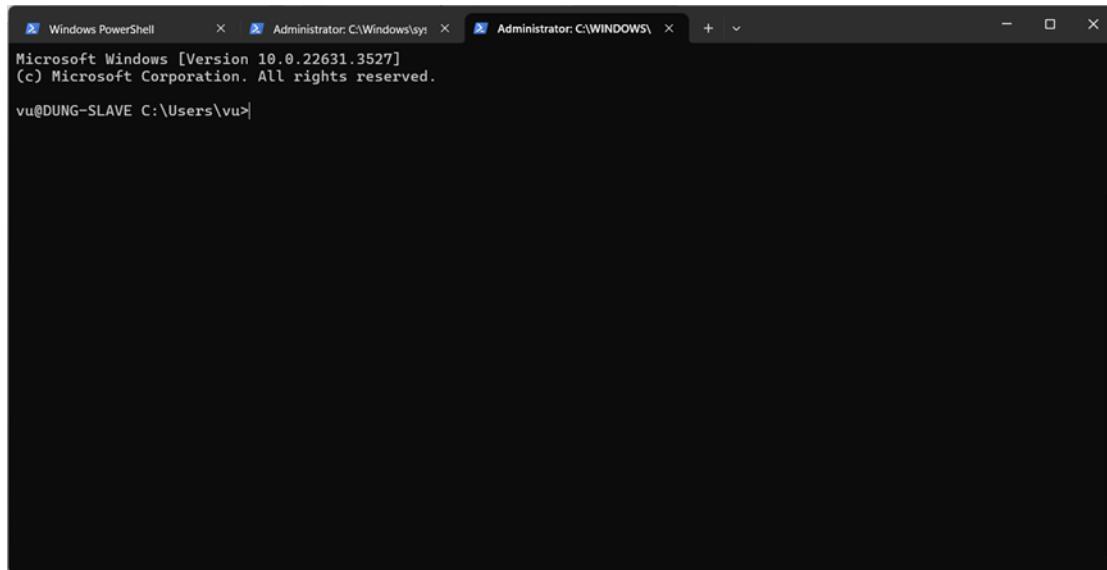
Thực hiện kết nối bằng lệnh: ssh khoan@192.168.1.229

Kết nối hành công sẽ có giao diện như bên dưới



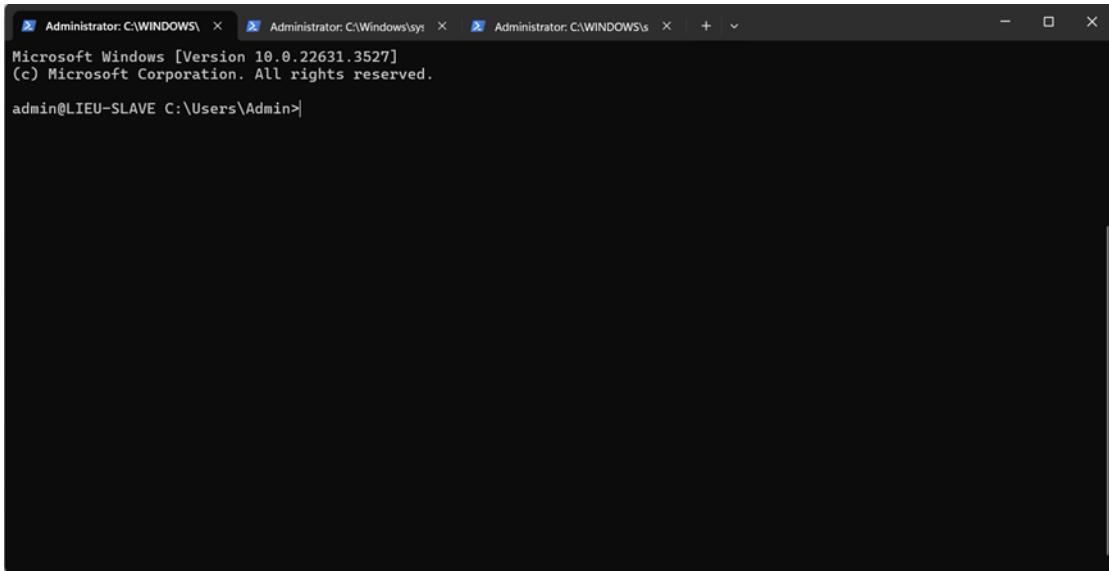
```
Windows PowerShell x Administrator: C:\Windows\sys x Administrator: C:\WINDOWS\sys + 
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

khoan@VO-SLAVE C:\Users\khoan>
```



```
Windows PowerShell x Administrator: C:\Windows\sys x Administrator: C:\WINDOWS\sys + 
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

vu@DUNG-SLAVE C:\Users\vu>
```



```
Administrator: C:\WINDOWS\ x Administrator: C:\Windows\sys x Administrator: C:\WINDOWS\sys x + v
Microsoft Windows [Version 10.0.22631.3527]
(c) Microsoft Corporation. All rights reserved.

admin@LIEU-SLAVE C:\Users\Admin>
```

1.2. Cài Hadoop trên Windows

Cài đặt JDK bản 1.8

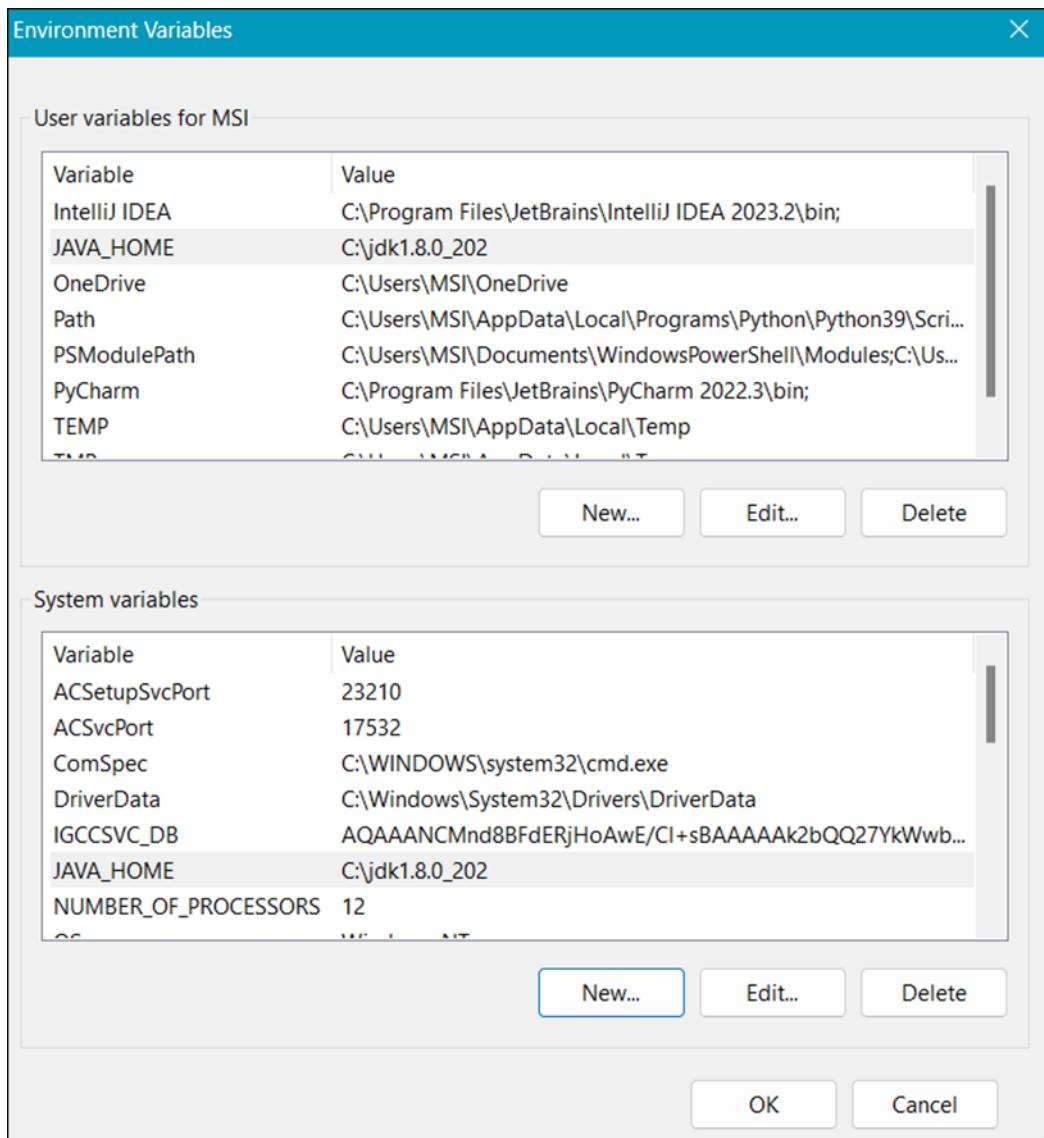
Link: <https://www.oracle.com/java/technologies/javase/javase8-archive-downloads.html#license-lightbox>

Trong mục user và system variables ta cấu hình JAVA_HOME trỏ tới nơi cài đặt JDK (bằng cách nhấn vào nút New...)

Variable name: JAVA_HOME

Variable value: C:\jdk1.8.0_202

Kết quả:

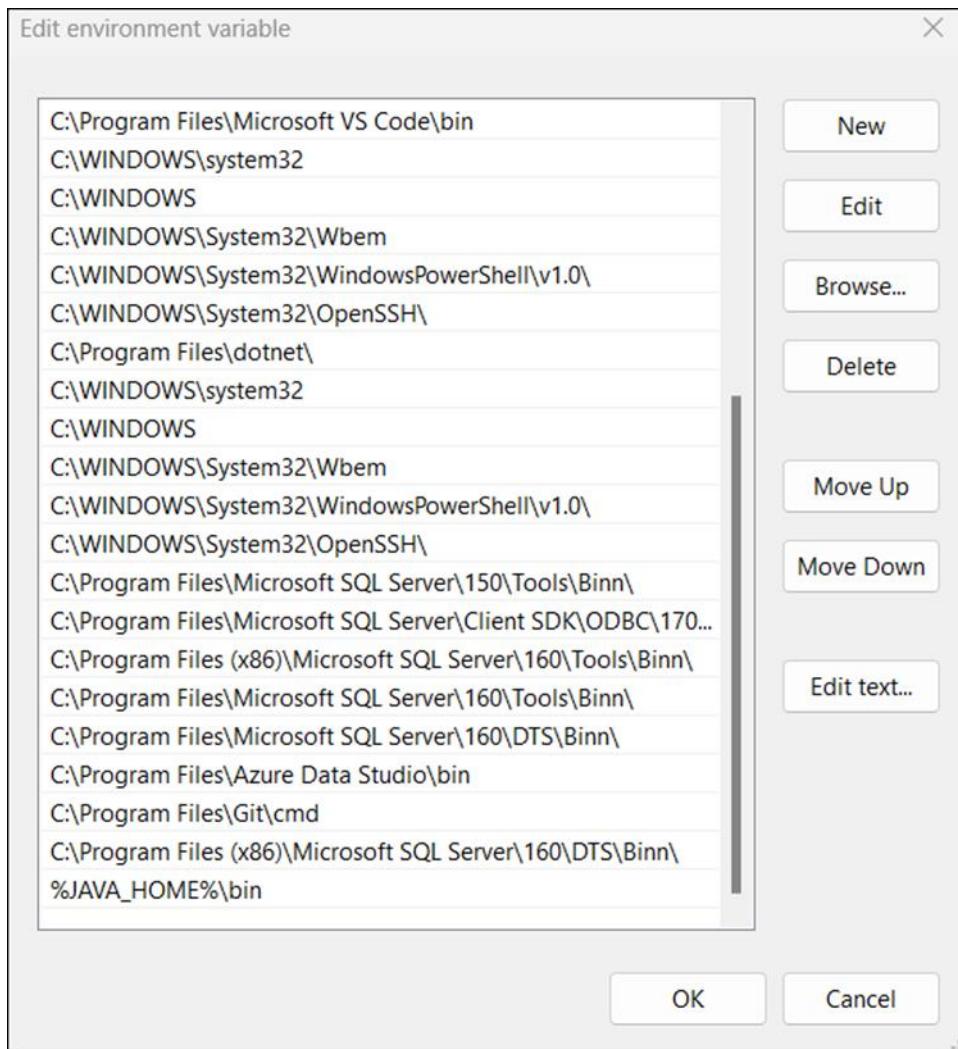


Sau đó bấm OK liên tục để đóng các cửa sổ cũng như xác nhận sự thay đổi

Tiếp theo cấu hình Path (cho cả user và system variable). Tìm tới biến Path, nhấn Edit:

Edit environment variable

C:\Users\MSI\AppData\Local\Programs\Python\Python39\Scripts\	New
C:\Users\MSI\AppData\Local\Programs\Python\Python39\	Edit
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps	Browse...
C:\Users\MSI\AppData\Local\Programs\MiKTeX\miktex\bin\x64\	Delete
%PyCharm%	Move Up
C:\Program Files\Azure Data Studio\bin	Move Down
%IntelliJ IDEA%	Edit text...
%USERPROFILE%.dotnet\tools	
C:\Users\MSI\AppData\Local\Google\Cloud SDK\google-cloud-...	
C:\Users\MSI\AppData\Local\Programs\mongosh\	
C:\Users\MSI\AppData\Local\Programs\Microsoft VS Code\bin	
%JAVA_HOME%\bin	

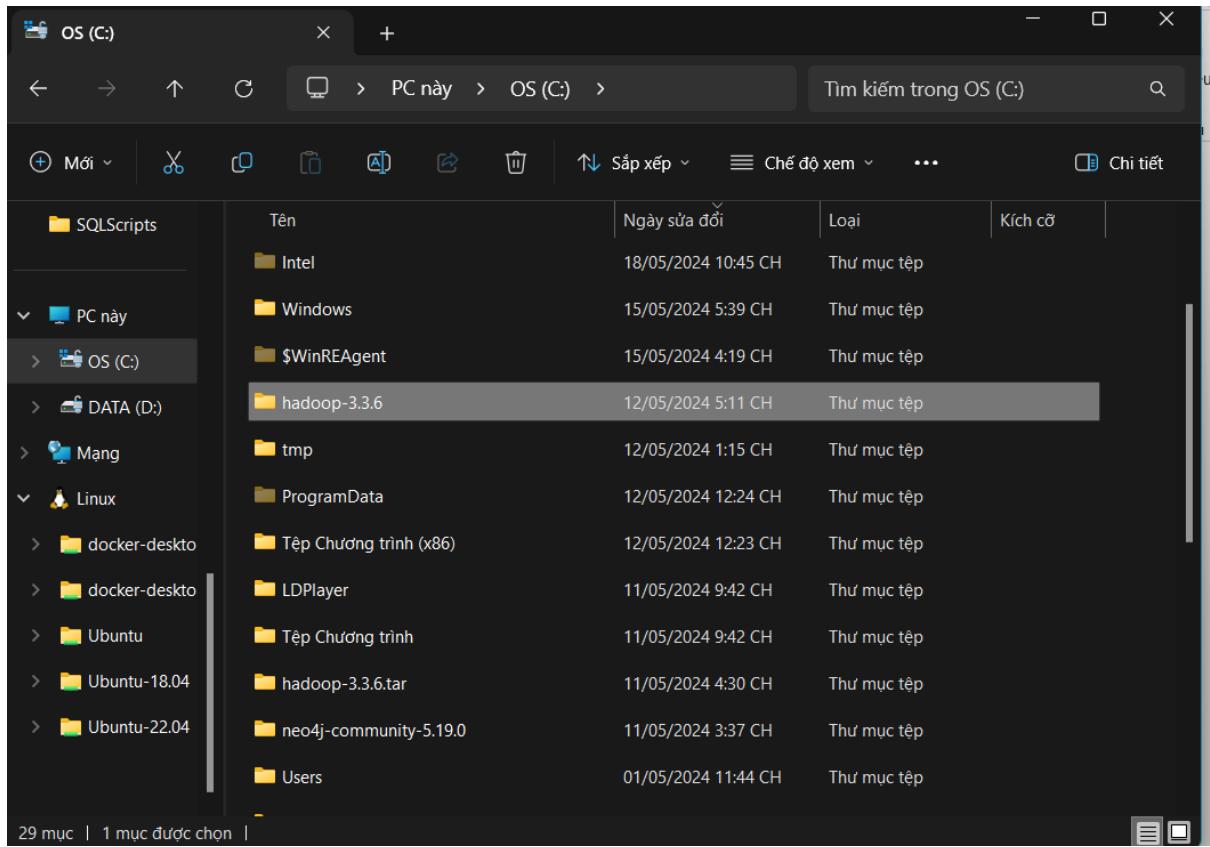


Cài đặt Hadoop và giải nén vào ổ đĩa C:

Cài đặt hadoop 3.3.6 ở link sau:

<https://archive.apache.org/dist/hadoop/common/hadoop-3.3.6/hadoop-3.3.6.tar.gz>

Giải nén vào ổ C:

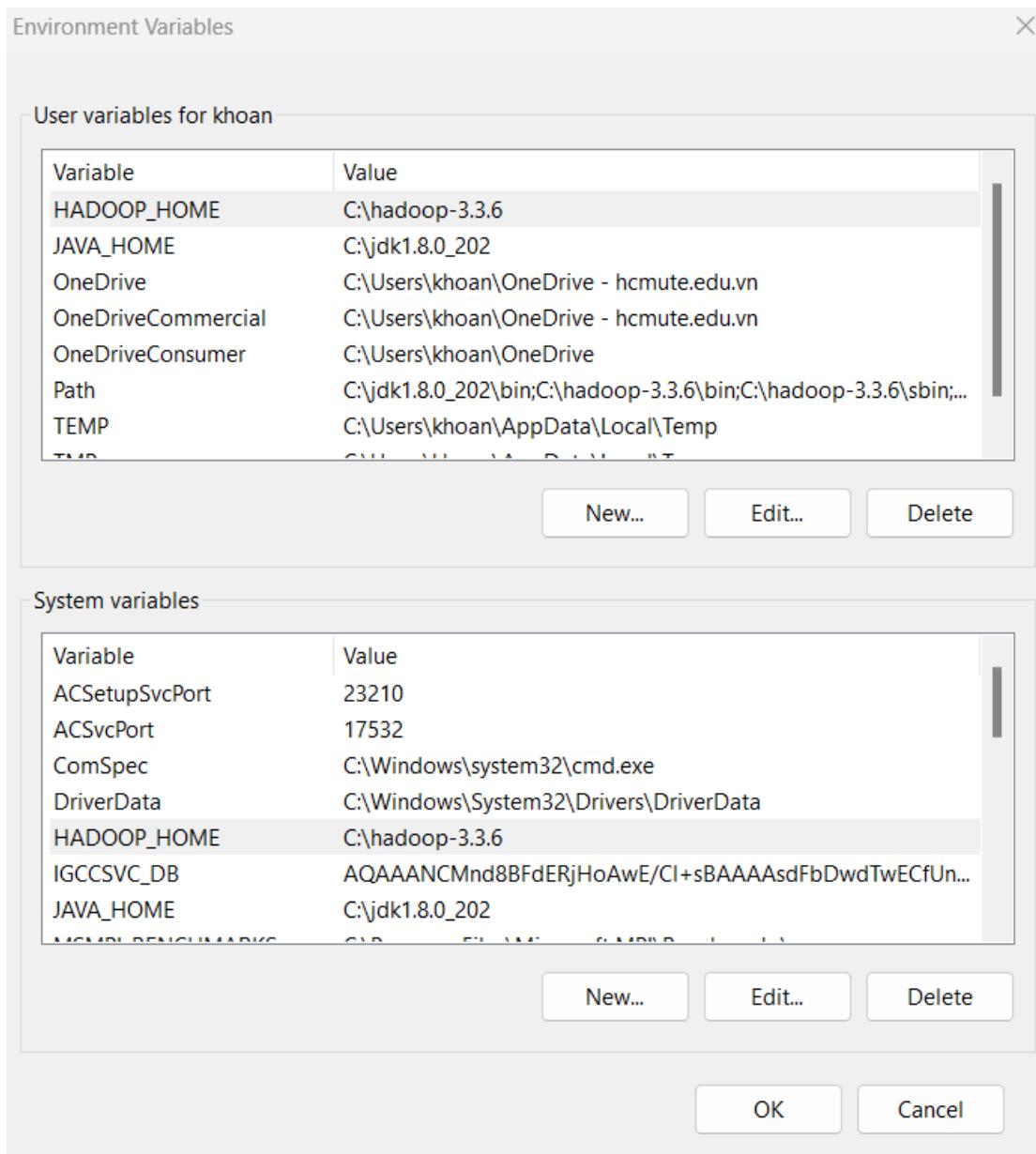


Thiết lập biến môi trường cho Hadoop

Lần lượt trong user và system variable thêm biến HADOOP_HOME có giá trị là C:\hadoop-3.3.6 mà ta giải nén ở trên.

Tương tự như JAVA JDK, ta cần cấu hình biến môi trường cho Hadoop (**HADOOP_HOME**)

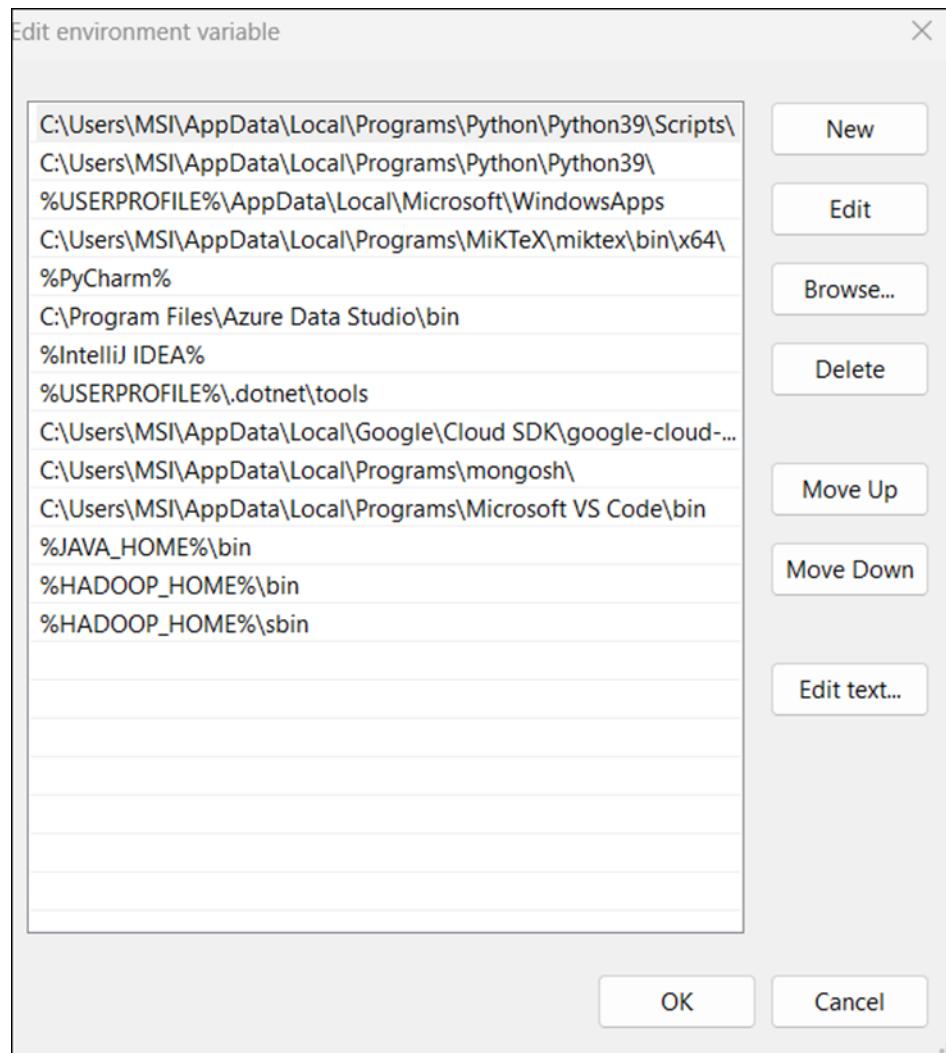
Lần lượt trong user và system variable thêm biến HADOOP_HOME có giá trị là C:\hadoop-3.3.6 mà ta giải nén ở trên.

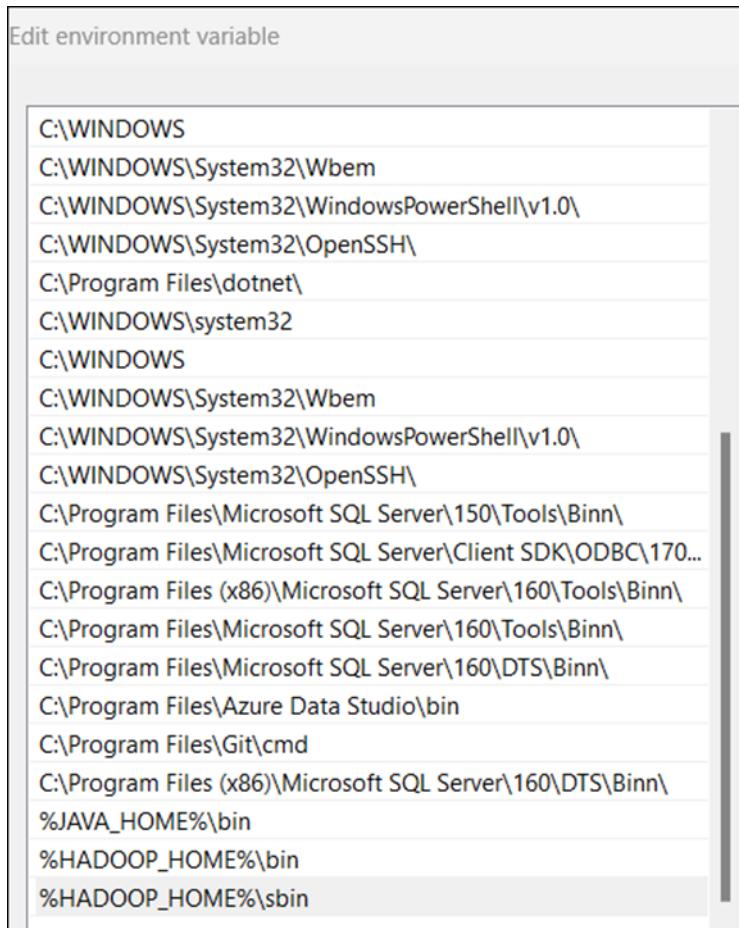


Sau đó chỉnh sửa biến path cho cả user và system variable. Bổ sung thêm:

%HADOOP_HOME%\bin

%HADOOP_HOME%\sbin





Nhấn Ok để đóng tất cả các cửa sổ

Mở CMD để test lại:

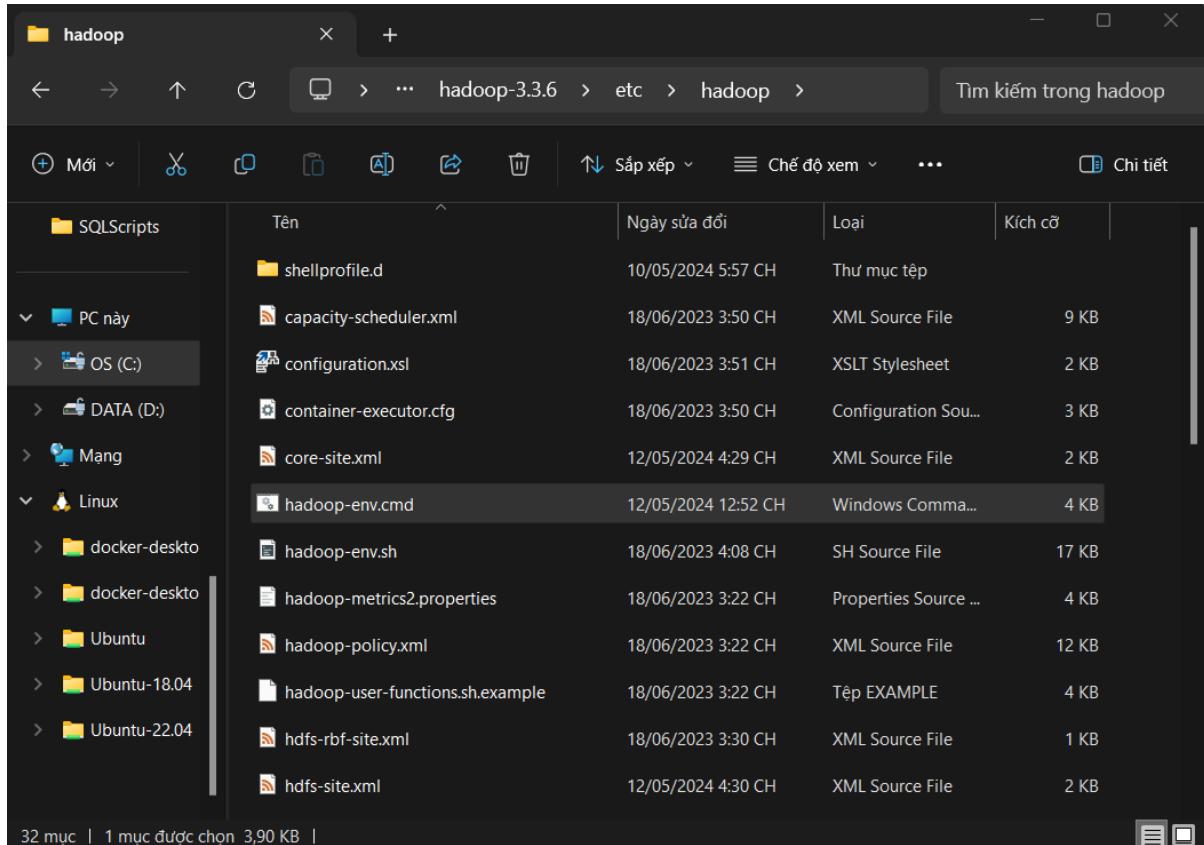
```
Dấu nhắc Lệnh
Microsoft Windows [Version 10.0.22631.3593]
(c) Microsoft Corporation. All rights reserved.

C:\Users\khoan>hadoop version
Hadoop 3.3.6
Source code repository https://github.com/apache/hadoop.git -r 1be78238728da9266a4f88195058f08fd012bf9c
Compiled by ubuntu on 2023-06-18T08:22Z
Compiled on platform linux-x86_64
Compiled with protoc 3.7.1
From source with checksum 5652179ad55f76cb287d9c633bb53bbd
This command was run using /C:/hadoop-3.3.6/share/hadoop/common/hadoop-common-3.3.6.jar

C:\Users\khoan>
```

Cấu hình hadoop tại tất cả máy slave và master như sau:

Hadoop-env.cmd



Đổi đường dẫn JAVA_HOME:

```
@rem The only required environment variable is JAVA_HOME. All others are  
@rem optional. When running a distributed configuration it is best to  
@rem set JAVA_HOME in this file, so that it is correctly defined on  
@rem remote nodes.  
  
@rem The java implementation to use. Required.  
set JAVA_HOME=C:/jdk1.8.0_202  
  
@rem The jsvc implementation to use. Jsvc is required to run secure datanodes.  
@rem set JSVC_HOME=%JSVC_HOME%  
  
@rem set HADOOP_CONF_DIR=  
  
@rem Extra Java CLASSPATH elements. Automatically insert capacity-scheduler.  
if exist %HADOOP_HOME%\contrib\capacity-scheduler (  
    if not defined HADOOP_CLASSPATH (  
        set HADOOP_CLASSPATH=%HADOOP_HOME%\contrib\capacity-scheduler\*.jar  
    ) else (  
        set HADOOP_CLASSPATH=%HADOOP_CLASSPATH%;%HADOOP_HOME%\contrib\capacity-scheduler\*.jar  
    )  
)  
  
@rem The maximum amount of heap to use, in MB. Default is 1000.  
@rem set HADOOP_HEAPSIZE=  
@rem set HADOOP_NAMENODE_INIT_HEAPSIZE=""
```

Core-site.xml

```
C: > hadoop-3.3.6 > etc > hadoop > core-site.xml
3  <!--
15   -->
16
17  <!-- Put site-specific property overrides in this file. -->
18
19  <configuration>
20      <property>
21          <name>hadoop.tmp.dir</name>
22          <value>/hadoop-3.3.6/tmp</value>
23      </property>
24      <property>
25          <name>dfs.permissions</name>
26          <value>false</value>
27      </property>
28      <property>
29          <name>fs.defaultFS</name>
30          <value>hdfs://vuong-master:9000</value>
31      </property>
32  </configuration>
33
```

Hdfs-site.xml

```
C:\> hadoop-3.3.6\etc\hadoop> hdfs-site.xml
19  <configuration>
20      <property>
21          <name>dfs.replication</name>
22          <value>1</value>
23      </property>
24      <property>
25          <name>dfs.namenode.name.dir</name>
26          <value>C:\hadoop-3.3.6\data\namenode</value>
27      </property>
28      <property>
29          <name>dfs.permissions</name>
30          <value>false</value>
31      </property>
32      <property>
33          <name>dfs.datanode.use.datanode.hostname</name>
34          <value>false</value>
35      </property>
36      <property>
37          <name>dfs.namenode.datanode.registration.ip-hostname-check</name>
38          <value>false</value>
39      </property>
40      <property>
41          <name>dfs.datanode.data.dir</name>
42          <value>/hadoop-3.3.6/data/datanode</value>
43      </property>
44      <property>
45          <name>dfs.namenode.http-address</name>
46          <value>vuong-master:50070</value>
47      </property>
48      <property>
49          <name>dfs.namenode.secondary.http-address</name>
50          <value>vuong-master:50090</value>
51      </property>
52  </configuration>
```

Mapred-site.xml

```
C: > hadoop-3.3.6 > etc > hadoop > mapred-site.xml

3   <!--
4     limitations under the License. See accompanying LICENSE file.
5   -->
6
7   <!-- Put site-specific property overrides in this file. -->
8
9   <configuration>
10    <property>
11      <name>mapreduce.framework.name</name>
12      <value>yarn</value>
13    </property>
14    <property>
15      <name>mapred.job.tracker</name>
16      <value>vuong-master:9001</value>
17    </property>
18  </configuration>
19
20
```

Yarn-site.xml

```

C: > hadoop-3.3.6 > etc > hadoop > yarn-site.xml
15  <configuration>
16      <property>
17          <name>yarn.nodemanager.aux-services</name>
18          <value>mapreduce_shuffle</value>
19      </property>
20      <property>
21          <name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
22          <value>org.apache.hadoop.mapred.ShuffleHandler</value>
23      </property>
24      <property>
25          <name>yarn.log-aggregation-enable</name>
26          <value>true</value>
27      </property>
28      <property>
29          <name>yarn.resourcemanager.scheduler.address</name>
30          <value>vuong-master:8030</value>
31      </property>
32      <property>
33          <name>yarn.resourcemanager.resoure-tracker.address</name>
34          <value>vuong-master:8031</value>
35      </property>
36      <property>
37          <name>yarn.resourcemanager.address</name>
38          <value>vuong-master:8032</value>
39      </property>
40      <property>
41          <name>yarn.resourcemanager.admin.address</name>
42          <value>vuong-master:8033</value>
43      </property>
44      <property>
45          <name>yarn.resourcemanager.webapp.address</name>
46          <value>vuong-master:8088</value>
47      </property>
48  </configuration>

```

Sau khi cấu hình hoàn tất, tắt cả các máy master và slave mở command line và di chuyển tới thư mục hadoop/sbin sau đó gõ lệnh: Start-all.cmd

Tắt cả máy slave,master truy cập vào: vuong-master:8088

Có thể kiểm tra kết nối ở: vuong-maser:50070

Ta

← → ⌂ Không bảo mật vuong-master:50070/dfshealth.html#tab-datanode

Disk usage of each DataNode (%)

In operation

DataNode State	All	Show	25	entries	Search:				
Node	Http Address	Last contact	Last Block Report	Used	Non DFS Used	Capacity	Blocks	Block pool used	Version
✓ default-rack(LAPTOP-TV41320R.mshome.net 9866 (192.168.1.127:9866))	http://LAPTOP-TV41320R.mshome.net:9866	1s	13m	55.73 KB	248 GB	251.08 GB	<div style="width: 55.73 KB;"></div>	38	55.73 KB (0%) 3.3.6
✓ default-rack(feu-slave 9866 (192.168.1.185:9866))	http://feu-slave:9866	145s	4m	150 B	153.87 GB	173.09 GB	<div style="width: 150 B;"></div>	0	150 B (0%) 3.3.6
✓ default-rack(dung-slave 9866 (192.168.1.184:9866))	http://dung-slave:9866	1s	16m	323 B	211.36 GB	225.43 GB	<div style="width: 323 B;"></div>	0	323 B (0%) 3.3.6
✓ default-rack(voslave.mshome.net 9866 (192.168.1.229:9866))	http://voslave.mshome.net:9866	1s	16m	323 B	141.72 GB	329.31 GB	<div style="width: 323 B;"></div>	0	323 B (0%) 3.3.6

Showing 1 to 4 of 4 entries

Previous 1 Next

Entering Maintenance

No nodes are entering maintenance.

Có thể kiểm tra bằng lệnh hdfs dfsadmin -report trên command line

```
Administrator: C:\Windows\sys Administrator: Command Pro + 
-----
Live datanodes (4):
Name: 192.168.1.127:9866 (vuong-master)
Hostname: LAPTOP-TV41320R.mshome.net
Decommission Status: Normal
Configured Capacity: 251.08322688 (251.08 GB)
DFS Used: 55.73 (55.73 MB)
Non DFS Used: 262989331921 (248.00 GB)
DFS Remaining: 33879333696 (3.08 GB)
DFS Used%: 9.00%
DFS Remaining%: 1.23%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 0
Last contact: Sun May 12 17:30:10 ICT 2024
Last Block Report: Sun May 12 17:14:49 ICT 2024
Num of Blocks: 38

Name: 192.168.1.184:9866 (dung-slave)
Hostname: dung-slave
Decommission Status: Normal
Configured Capacity: 225.4322624 (225.43 GB)
DFS Used: 323 (323 B)
Non DFS Used: 226941738685 (211.36 GB)
DFS Remaining: 15116783616 (14.08 GB)
DFS Used%: 0.00%
DFS Remaining%: 6.25%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 0
Last contact: Sun May 12 17:30:08 ICT 2024
Last Block Report: Sun May 12 17:11:24 ICT 2024
```

```

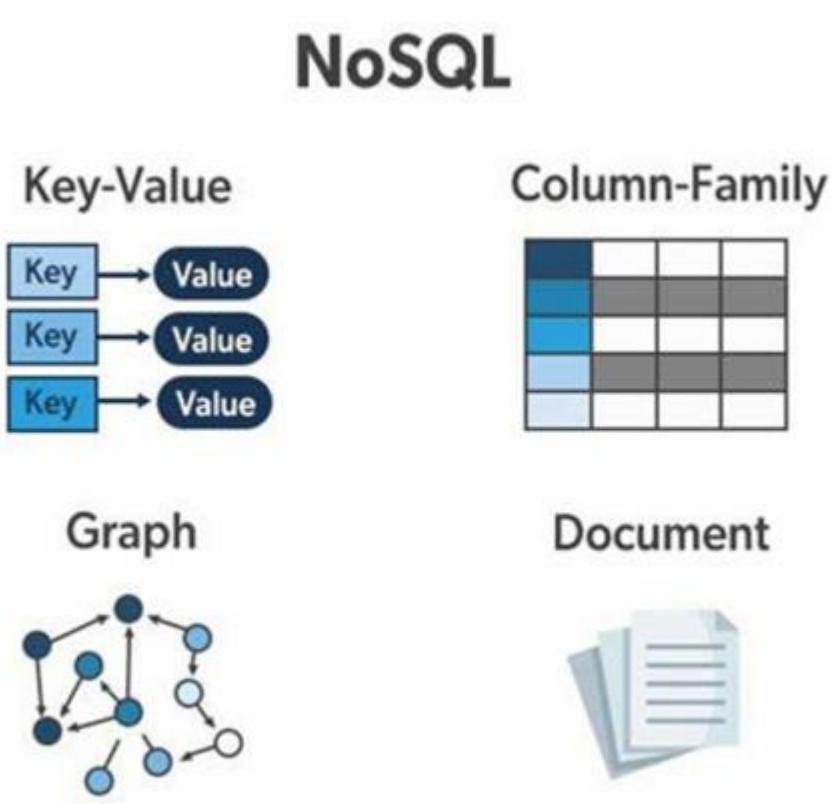
Administrator: C:\Windows\sys | Administrator: Command Pro + x
Hostname: lieu-slave
Decommission Status : Normal
Configured Capacity: 185854849024 (173.09 GB)
DFS Used: 158 (158 B)
Non DFS Used: 165211684714 (153.87 GB)
DFS Remaining: 20643164168 (19.23 GB)
DFS Used%: 0.00%
DFS Remaining%: 11.11%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 0
Last contact: Sun May 12 17:38:09 ICT 2024
Last Block Report: Sun May 12 17:28:42 ICT 2024
Num of Blocks: 0

Name: 192.168.1.229:9866 (vo-slave)
Hostname: Vo-slave.mshome.net
Decommission Status : Normal
Configured Capacity: 353590644736 (329.31 GB)
DFS Used: 323 (323 B)
Non DFS Used: 152178587965 (141.72 GB)
DFS Remaining: 2014201364448 (187.59 GB)
DFS Used%: 0.00%
DFS Remaining%: 56.96%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Xceivers: 0
Last contact: Sun May 12 17:38:11 ICT 2024
Last Block Report: Sun May 12 17:12:01 ICT 2024
Num of Blocks: 0

C:\hadoop-3.3.6\sbin>

```

Chương 2. GIỚI THIỆU NOSQL



2.1. Khái niệm:

Cơ sở dữ liệu NoSQL (Not Only SQL) là một loại cơ sở dữ liệu phi quan hệ, không sử dụng cấu trúc bảng quan hệ như trong cơ sở dữ liệu quan hệ truyền thống. Thay vào

đó, NoSQL cung cấp các cơ chế lưu trữ và truy xuất dữ liệu khác nhau, không giới hạn bởi một giản đồ cố định. NoSQL thường được phát triển với mục tiêu hỗ trợ các ứng dụng hiện đại, có khả năng lưu trữ lượng dữ liệu lớn và truy vấn dữ liệu với tốc độ cao mà không đòi hỏi quá nhiều tài nguyên phần cứng và hệ thống.

2.2. Cách hoạt động

NoSQL có các mô hình dữ liệu đa dạng như key-value, document, graph, wide column, ... dễ dàng mở rộng ngang, hỗ trợ sao chép dữ liệu, API đơn giản. NoSQL phù hợp cho hệ thống phân tán, lưu trữ lượng lớn dữ liệu phi cấu trúc, truy vấn hiệu quả. Tuy nhiên, NoSQL có thể có hạn chế về tính nhất quán dữ liệu và truy vấn phức tạp.

Lựa chọn NoSQL khi cần lưu trữ linh hoạt, mở rộng dễ dàng, truy vấn hiệu quả, phát triển ứng dụng nhanh chóng, tiết kiệm chi phí. NoSQL mang đến giải pháp lưu trữ dữ liệu linh hoạt, hiệu quả cho ứng dụng hiện đại.

2.3. Phân loại NoSQL

Cơ sở dữ liệu NoSQL được phân loại theo mô hình dữ liệu chính mà chúng sử dụng để lưu trữ và truy xuất dữ liệu. Dưới đây là bốn loại NoSQL phổ biến nhất:

- Key-value:

- Lưu trữ dữ liệu dưới dạng cặp khóa-giá trị.
- Mỗi khóa là một định danh duy nhất để truy cập giá trị tương ứng.
- Dễ dàng mở rộng và truy cập dữ liệu nhanh chóng.
- Ví dụ: Riak, Cassandra, DynamoDB.

- Document:

- Lưu trữ dữ liệu dưới dạng các tài liệu JSON hoặc BSON.
- Mỗi tài liệu là một tập hợp các cặp khóa-giá trị, có thể lồng nhau và có cấu trúc linh hoạt.
- Phù hợp cho lưu trữ dữ liệu phi cấu trúc và truy vấn dữ liệu theo thuộc tính.

- Ví dụ: MongoDB, CouchDB.

- **Graph:**

- Lưu trữ dữ liệu dưới dạng đồ thị, với các nút và cạnh biểu thị mối quan hệ giữa các thực thể.
- Hiệu quả cho việc lưu trữ và truy vấn dữ liệu có mối quan hệ phức tạp.
- Phù hợp cho các ứng dụng mạng xã hội, bản đồ tri thức,...
- Ví dụ: Neo4j, OrientDB.

- **Wide column:**

- Lưu trữ dữ liệu trong các bảng có nhiều cột, mỗi cột có thể có nhiều giá trị.
- Dễ dàng mở rộng theo chiều ngang và truy vấn dữ liệu hiệu quả cho các tập dữ liệu lớn.
- Phù hợp cho lưu trữ dữ liệu chuỗi thời gian, nhật ký,...
- Ví dụ: Cassandra, HBase

2.4. Ưu điểm và nhược điểm của NoSQL

2.4.1. Ưu điểm:

- **Linh hoạt:**

- Lưu trữ đa dạng: NoSQL có thể lưu trữ nhiều loại dữ liệu khác nhau, bao gồm dữ liệu phi cấu trúc (như JSON, XML), dữ liệu bán cấu trúc (như CSV) và dữ liệu có cấu trúc (như bảng).
- Dễ dàng thay đổi: Cấu trúc dữ liệu NoSQL không cần được định nghĩa trước, do đó bạn có thể dễ dàng thêm, sửa đổi hoặc xóa các trường dữ liệu mà không cần thay đổi toàn bộ schema của cơ sở dữ liệu.
- Hỗ trợ nhiều mô hình dữ liệu: NoSQL cung cấp nhiều mô hình dữ liệu khác nhau, bao gồm Key-Value, Document, Graph và Columnar. Mỗi mô hình dữ liệu phù hợp với những kiểu dữ liệu và truy vấn khác nhau.

- **Khả năng mở rộng:**

- Mở rộng theo chiều ngang: NoSQL có thể dễ dàng mở rộng theo chiều ngang bằng cách thêm nhiều máy chủ vào hệ thống.
 - Phân tán dữ liệu: Dữ liệu NoSQL được phân tán trên nhiều máy chủ, giúp tăng khả năng sẵn sàng và giảm thiểu nguy cơ mất dữ liệu.
- **Hiệu suất cao:**
- Tối ưu hóa truy vấn: NoSQL được tối ưu hóa cho truy vấn dữ liệu với tốc độ cao, đặc biệt là với lượng dữ liệu lớn.
 - Truy vấn phi quan hệ: NoSQL hỗ trợ các truy vấn phi quan hệ hiệu quả, ví dụ như truy vấn theo ngữ cảnh, truy vấn theo vị trí địa lý, truy vấn theo mối quan hệ.
- **Dễ sử dụng:**
- Cấu trúc đơn giản: Cấu trúc dữ liệu NoSQL thường đơn giản và dễ hiểu hơn so với cấu trúc dữ liệu quan hệ.
 - Nhiều ngôn ngữ lập trình: NoSQL cung cấp nhiều API và thư viện cho các ngôn ngữ lập trình phổ biến, giúp các nhà phát triển dễ dàng tích hợp NoSQL vào ứng dụng của họ.
- **Chi phí thấp:**
- Phần cứng: NoSQL thường có thể chạy trên phần cứng rẻ hơn so với phần cứng cần thiết cho cơ sở dữ liệu quan hệ.
 - Giấy phép: Nhiều cơ sở dữ liệu NoSQL là mã nguồn mở hoặc miễn phí, giúp giảm chi phí giấy phép.
- 2.4.2. Nhược điểm:**
- **Khó truy vấn phức tạp:**
- Việc truy vấn dữ liệu phức tạp có thể khó khăn hơn so với cơ sở dữ liệu quan hệ.

- NoSQL thường thiếu các tính năng truy vấn nâng cao như join, subqueries và aggregation.

- **Ít công cụ hỗ trợ:**

- Số lượng công cụ hỗ trợ cho NoSQL ít hơn so với cơ sở dữ liệu quan hệ.
- Việc quản lý và giám sát cơ sở dữ liệu NoSQL có thể khó khăn hơn do thiếu các công cụ hỗ trợ chuyên dụng.

- **Yêu cầu chuyên môn cao:**

- Việc triển khai và quản lý NoSQL có thể đòi hỏi chuyên môn cao hơn so với cơ sở dữ liệu quan hệ.
- Các nhà phát triển cần có kiến thức về các mô hình dữ liệu NoSQL và các kỹ thuật truy vấn NoSQL để sử dụng NoSQL hiệu quả.

- **Ít tính nhất quán:**

- NoSQL thường ít nhất quán hơn so với cơ sở dữ liệu quan hệ.
- Điều này có thể dẫn đến một số vấn đề về tính toàn vẹn dữ liệu, đặc biệt là trong các ứng dụng đòi hỏi tính nhất quán cao.

2.5. So sánh Nosql với SQL:

Tính năng	SQL	NoSQL
Mô hình dữ liệu	Quan hệ (bảng, cột)	Phi quan hệ (key-value, document, graph, wide column)
Khả năng mở rộng	Khó mở rộng theo chiều ngang	Dễ dàng mở rộng theo chiều ngang
Tính nhất quán dữ liệu	Strongly Consistent (nhất quán mạnh)	Eventually Consistent (nhất quán cuối cùng) hoặc Strongly Consistent

Cấu trúc dữ liệu	Cố định	Linh hoạt
Truy vấn dữ liệu	SQL	Ngôn ngữ truy vấn khác nhau tùy thuộc vào mô hình dữ liệu
Lợi ích	Dễ sử dụng, truy vấn phức tạp mạnh mẽ, tính nhất quán dữ liệu cao	Khả năng mở rộng cao, linh hoạt, hiệu suất cao
Nhược điểm	Khó mở rộng cho khối lượng dữ liệu lớn, không phù hợp cho dữ liệu phi cấu trúc	Có thể có hạn chế về tính nhất quán dữ liệu, truy vấn phức tạp hạn chế hơn SQL
Chi phí	Cả giải pháp phần mềm nguồn mở và thương mại đều có sẵn	Phần mềm nguồn mở phổ biến hơn
Cộng đồng	Cộng đồng người dùng và nhà phát triển lớn	Cộng đồng đang phát triển nhanh chóng
Độ trưởng thành	Công nghệ đã được phát triển và sử dụng trong nhiều thập kỷ	Công nghệ mới hơn
Bảo mật	Cung cấp các tính năng bảo mật mạnh mẽ	Một số cơ sở dữ liệu NoSQL có thể có lỗ hổng bảo mật
Hỗ trợ ACID	Hỗ trợ đầy đủ mô hình ACID	Một số cơ sở dữ liệu NoSQL hỗ trợ ACID đầy đủ, trong khi một số khác có thể cung cấp các mức độ nhất quán dữ liệu khác nhau

Chương 3. CƠ SỞ DỮ LIỆU RIAK

3.1. Giới thiệu:

Riak là một loại cơ sở dữ liệu phân tán được xây dựng bởi Basho Technology để có thể tối đa hóa quá trình chuyển dữ liệu tới người dùng qua nhiều server. Riak được chia

làm 2 phần chính: Riak server và Riak client. Khi Riak client có thể kết nối tới một server là có thể truy cập dữ liệu.

3.1.1. Đặc điểm chính của Riak trong NoSQL:

- **Khả năng Mở Rộng Linh Hoạt:** Riak được thiết kế để có thể mở rộng theo chiều ngang một cách linh hoạt bằng cách thêm các nút vào cụm. Điều này cho phép hệ thống mở rộng dễ dàng để đáp ứng với sự tăng trưởng của dữ liệu và lưu lượng truy cập mà không gặp vấn đề về hiệu suất.
- **Tính Nhất Quán Cao:** Riak đảm bảo tính nhất quán của dữ liệu thông qua các cơ chế như vector clocks và quorum-based replication. Điều này đảm bảo rằng các thay đổi dữ liệu được thực hiện trên các nút sẽ được đồng bộ và không có sự xung đột dữ liệu.
- **Độ Bền Bỉ và Tin Cậy:** Riak được thiết kế để chịu được sự cố phần cứng và phần mềm một cách linh hoạt. Dữ liệu được phân tán trên nhiều nút và sao lưu, giúp đảm bảo rằng dữ liệu vẫn có thể truy cập được ngay cả khi một hoặc nhiều nút gặp sự cố.
- **Hỗ Trợ CRDTs:** Riak hỗ trợ sử dụng CRDTs để giải quyết xung đột dữ liệu một cách tự động và không cần sự can thiệp từ phía người dùng. Điều này giúp giải quyết các vấn đề về tính nhất quán của dữ liệu một cách hiệu quả trong môi trường phân tán.
- **Tính Linh Hoạt và Đa Dạng của Dữ Liệu:** Riak hỗ trợ lưu trữ và truy vấn dữ liệu đa dạng, từ key-value đến dữ liệu không cấu trúc và dữ liệu phân tán. Điều này cho phép nó được sử dụng trong một loạt các ứng dụng và kịch bản sử dụng.
- **Hiệu Suất và Tốc Độ Truy Cập:** Riak được thiết kế để cung cấp hiệu suất và tốc độ truy cập cao, đặc biệt là trong việc xử lý các tải công việc lớn và lưu lượng truy cập đồng thời từ nhiều nguồn.

3.1.2. Ưu điểm

- **Khả năng phục hồi cao:** Riak KV đảm bảo việc đọc và ghi dữ liệu ngay cả khi có sự cố về phần cứng hoặc mạng.

- Quy mô lớn: Có khả năng mở rộng gần như tuyến tính, cho phép bạn thêm dung lượng một cách dễ dàng bằng cách sử dụng phần cứng thông thường.
- Đơn giản vận hành: Riak KV tự động phân phối dữ liệu qua cluster để đạt hiệu suất cao và duy trì hoạt động kinh doanh mạnh mẽ với kiến trúc không có master.
- Hỗ trợ truy vấn phức tạp: Riak KV cung cấp ba cách để truy vấn dữ liệu bằng cách sử dụng tìm kiếm văn bản đầy đủ Solr, chỉ mục phụ và Map Reduce.
- Hết hạn đối tượng toàn cầu: Cho phép bạn xác định khi nào dữ liệu cũ được loại bỏ khỏi cơ sở dữ liệu.
- Vector phiên bản chấm (DVs): Giải quyết xung đột tự động bằng cách theo dõi thời gian logic thay vì thời gian thực tế.
- Kiểu dữ liệu phân tán của Riak: Cung cấp các kiểu dữ liệu được xây dựng sẵn cho các cấu trúc dữ liệu phổ biến nhất cần thiết cho các tải công việc phân tán, hoạt động.
- API và thư viện khách hàng mạnh mẽ: Hỗ trợ nhiều ngôn ngữ lập trình như Java, Ruby, Python, C#, Erlang, Node.js và .NET.
- Kết nối Spark: Tích hợp với Apache Spark để cung cấp phân tích thời gian thực của Spark với khả năng sẵn có và khả năng mở rộng của Riak.

3.1.3. Nhược điểm:

- **Khó khăn trong quản lý:** Riak có thể đòi hỏi một lượng lớn kiến thức và kỹ năng để triển khai và quản lý hiệu quả, đặc biệt khi cần phải xử lý các vấn đề như tối ưu hóa hiệu suất, cấu hình đồng bộ hoặc xử lý sự cố.
- **Có thể phức tạp với các truy vấn phức tạp:** Riak thường được tối ưu hóa cho các truy vấn đơn giản và cơ bản. Việc thực hiện các truy vấn phức tạp hoặc phức tạp có thể đòi hỏi kiến thức sâu rộng về cách hoạt động của hệ thống và các cơ chế đồng thuận.
- **Yêu cầu kiến thức về phân phối dữ liệu:** Sử dụng Riak hiệu quả đòi hỏi hiểu biết sâu rộng về cách phân phối dữ liệu trên các nút, cùng với khả năng đánh giá và quản lý các vấn đề liên quan đến tính nhất quán và đồng bộ.

- **Hiệu suất có thể giảm khi quá tải:** Mặc dù Riak được thiết kế để xử lý tải công việc lớn, nhưng nếu hệ thống bị quá tải, hiệu suất có thể giảm đáng kể và gây ra các vấn đề về độ trễ.
- **Hạn chế về công cụ và tài liệu:** So với một số hệ thống NoSQL khác, Riak có thể có ít tài liệu và công cụ hỗ trợ, điều này có thể làm tăng độ phức tạp trong việc triển khai và quản lý.
- **Chi phí:** Mặc dù Riak là một giải pháp mã nguồn mở, nhưng triển khai và quản lý một hệ thống Riak có thể đòi hỏi chi phí lớn, đặc biệt là khi cần phải xử lý các yêu cầu về hiệu suất và tính nhất quán.

3.2. Các mô hình dữ liệu trong Riak:

Riak cung cấp nhiều mô hình dữ liệu khác nhau, mỗi mô hình phù hợp cho các trường hợp sử dụng cụ thể. Dưới đây là các mô hình dữ liệu chính trong Riak:

3.2.1. Key-Value (Khóa-Giá trị):

- Mô hình key-value là mô hình lưu trữ cơ bản nhất trong Riak, trong đó mỗi dữ liệu được lưu trữ dưới dạng một cặp khóa-giá trị.
- Key định danh duy nhất cho mỗi giá trị, và giá trị có thể là bất kỳ loại dữ liệu nào, từ một chuỗi đến một đối tượng JSON hoặc một tập hợp các byte.

3.2.2. Document (Tài liệu):

- Riak hỗ trợ lưu trữ dữ liệu dưới dạng các tài liệu JSON hoặc XML, cho phép lưu trữ các tài liệu có cấu trúc phức tạp hơn so với mô hình key-value đơn giản.
- Mỗi tài liệu được lưu trữ dưới một key duy nhất và có thể được truy vấn sử dụng các truy vấn phức tạp hơn.

3.2.3. Counter (Bộ đếm):

- Riak cung cấp một kiểu dữ liệu counter để lưu trữ các bộ đếm tăng giảm theo thời gian.
- Bộ đếm được lưu trữ dưới dạng key-value với giá trị là một con số nguyên, và có thể được tăng hoặc giảm một cách an toàn từ nhiều nút khác nhau mà không gây ra xung đột dữ liệu.

3.2.4. Set (Tập hợp) và Map (Bản đồ):

- Riak hỗ trợ lưu trữ dữ liệu dưới dạng các tập hợp và bản đồ, cho phép lưu trữ và truy vấn dữ liệu theo các cấu trúc dữ liệu phức tạp hơn.
- Tập hợp lưu trữ một tập hợp các giá trị duy nhất không có thứ tự, trong khi bản đồ lưu trữ các cặp khóa-giá trị có thứ tự.

3.2.5. HyperLogLog (HLL):

- Riak cung cấp hỗ trợ cho cấu trúc HyperLogLog để ước lượng số lượng phần tử duy nhất trong một tập hợp lớn với độ chính xác cao.
- Cấu trúc HyperLogLog được sử dụng để tính toán ước lượng số lượng phần tử duy nhất mà không cần lưu trữ tất cả các phần tử.

3.3. Kiến trúc của Riak

3.3.1. Cụm Riak (Riak Cluster):

Riak hoạt động trong một cụm (cluster) gồm nhiều nút (nodes). Cụm Riak có thể chứa hàng trăm hoặc thậm chí hàng nghìn nút, tùy thuộc vào quy mô và yêu cầu của hệ thống.

3.3.2. Nút Riak (Riak Node):

- Mỗi nút Riak là một máy chủ độc lập trong cụm, có khả năng lưu trữ dữ liệu và thực hiện các hoạt động như đọc, ghi và xóa dữ liệu.
- Mỗi nút có trạng thái riêng và thường giao tiếp với các nút khác trong cụm để đồng bộ hóa dữ liệu và quản lý trạng thái hệ thống.

3.3.3. Dynamo-style Ring

- Riak sử dụng một hệ thống phân phối dữ liệu được gọi là "Dynamo-style ring" để quản lý việc phân tán dữ liệu trên các nút.
- Mỗi nút trong cụm được gán một vị trí duy nhất trong vòng (ring) dựa trên một hàm băm, cho phép việc phân phối dữ liệu hiệu quả.

3.3.4. Phân phối dữ liệu và Replication:

- Dữ liệu trong Riak được phân tán và sao lưu trên nhiều nút trong cụm để đảm bảo tính nhất quán và độ bền bỉ.

- Dữ liệu thường được sao lưu trên nhiều nút để đảm bảo khả năng phục hồi dữ liệu trong trường hợp một hoặc nhiều nút gặp sự cố.

3.3.5. Quorum-based Consensus:

- Riak sử dụng cơ chế đồng thuận dựa trên quorum để đảm bảo tính nhất quán của dữ liệu.
- Khi thực hiện các hoạt động như đọc, ghi hoặc xoá dữ liệu, Riak yêu cầu một số lượng quorum của các nút tham gia để xác nhận hoạt động trước khi nó được coi là thành công.

3.3.6. Riak Core:

- Riak Core là một framework mở và đơn giản cho việc xây dựng các hệ thống phân tán có tính nhất quán.
- Nó cung cấp các khái niệm và công cụ cần thiết để quản lý việc phân phối và lưu trữ dữ liệu trên các nút Riak.

3.4. Chế độ thực thi của riak

3.4.1. Single-Node Mode (Chế độ Một nút):

- Trong chế độ này, Riak hoạt động trên một nút duy nhất, không có sự phân tán. Đây thường là chế độ được sử dụng cho môi trường phát triển hoặc kiểm thử.

3.4.2. Multi-Node Mode (Chế độ Nhiều nút):

- Trong chế độ này, Riak hoạt động trên nhiều nút được kết nối với nhau thành một cluster. Dữ liệu được phân tán trên các nút để đảm bảo sự phân tán và sẵn sàng cao.

3.4.3. Active-Anti-Entropy (AAE):

AAE là một chế độ thực thi trong Riak được sử dụng để đồng bộ hóa dữ liệu giữa các nút trong cluster. Nó đảm bảo rằng các bản sao của dữ liệu trên các nút là nhất quán và không mất mát.

3.4.4. Hinted Handoff:

Đây là một chế độ chịu lỗi trong Riak. Khi một nút không thể giao tiếp với một nút khác để sao chép dữ liệu, nút đó sẽ lưu trữ dữ liệu vào bản sao của chính nó và giữ gợi ý (hint) cho nút đích. Khi nút đích trở lại trạng thái hoạt động, nó sẽ lấy dữ liệu từ nút khác và xóa gợi ý.

3.4.5. Read Repair và Anti-Entropy Repair:

Đây là các quá trình tự động trong Riak để sửa chữa dữ liệu khi phát hiện ra lỗi hoặc không nhất quán. Read Repair được kích hoạt khi một yêu cầu đọc dữ liệu được gửi và phát hiện ra rằng có sự không nhất quán. Anti-Entropy Repair thực hiện việc sửa chữa lỗi dữ liệu giữa các nút trong cluster.

3.5. Ứng dụng của Riak

Redis là một nền tảng lưu trữ dữ liệu trong bộ nhớ mã nguồn mở, được sử dụng rộng rãi cho nhiều trường hợp sử dụng khác nhau. Dưới đây là một số ứng dụng phổ biến của Redis:

3.5.1. Ứng dụng Web và Mobile:

Riak thích hợp cho các ứng dụng web và di động với số lượng lớn người dùng. Các ứng dụng này thường cần khả năng mở rộng mạnh mẽ để xử lý tải trọng cao và Riak có thể được sử dụng để lưu trữ dữ liệu người dùng, phiên và nội dung.

3.5.2. Phân tích dữ liệu lớn (Big Data):

Riak có thể được sử dụng làm một phần của hệ thống xử lý dữ liệu lớn để lưu trữ và xử lý dữ liệu có cấu trúc hoặc không cấu trúc. Nó có thể tích hợp với các công cụ phân tích như Apache Hadoop hoặc Apache Spark để cung cấp khả năng lưu trữ dữ liệu phân tán và tính toán song song.

3.5.3. Dịch vụ Lưu trữ Tệp:

Riak có thể được sử dụng làm nền tảng lưu trữ tệp phân tán cho các ứng dụng yêu cầu lưu trữ dữ liệu lớn, chẳng hạn như dịch vụ lưu trữ đám mây hoặc lưu trữ dữ liệu đa phương tiện như hình ảnh, video và tệp âm thanh.

3.5.4. Game trực tuyến và ứng dụng Real-Time:

Riak có thể được sử dụng cho các trò chơi trực tuyến và ứng dụng Real-Time để lưu trữ dữ liệu trò chơi, dữ liệu người dùng và trạng thái game. Khả năng mở rộng và sẵn

sang cao của Riak giúp đảm bảo rằng các trò chơi có thể mở rộng để chịu được lượng người chơi tăng cao.

3.5.5. Ứng dụng IoT (Internet of Things):

Riak cung cấp khả năng lưu trữ và xử lý dữ liệu từ các thiết bị IoT. Các thiết bị này thường tạo ra lượng lớn dữ liệu cần được lưu trữ và phân tích, và Riak có thể giúp xử lý dữ liệu này một cách hiệu quả.

Chương 4. Cài Đặt, Cấu Hình và Thao Tác Cơ bản Riak

4.1. Cài Đặt Riak

Sử Dụng người dùng **root** để cài đặt:

Đầu tiên ta phải update các gói trong ubuntu bằng 2 câu lệnh:

```
# sudo apt update
```

```
# sudo apt upgrade
```

```
anhdung3@ubuntu:~$ sudo apt update
[sudo] password for anhdung3:
Hit:1 http://security.ubuntu.com/ubuntu xenial-security InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu xenial InRelease
Hit:3 http://us.archive.ubuntu.com/ubuntu xenial-updates InRelease
Hit:4 http://us.archive.ubuntu.com/ubuntu xenial-backports InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
191 packages can be upgraded. Run 'apt list --upgradable' to see them.
anhdung3@ubuntu:~$ sudo apt upgrade
Reading package lists... Done
```

Tiếp theo ta tải libpam0g-dev bằng câu lệnh:

```
# sudo apt-get install libpam0g-dev
```

```
done
anhdung3@ubuntu:~$ sudo apt-get install libpam0g-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  libpam0g-dev
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 109 kB of archives.
After this operation, 381 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu xenial-updates/main amd64 libpam0g-dev
  amd64 1.1.8-3.2ubuntu2.3 [109 kB]
Fetched 109 kB in 2s (38.0 kB/s)
Selecting previously unselected package libpam0g-dev:amd64.
(Reading database ... 213119 files and directories currently installed.)
Preparing to unpack .../libpam0g-dev_1.1.8-3.2ubuntu2.3_amd64.deb ...
  Ubuntu Software 0g-dev:amd64 (1.1.8-3.2ubuntu2.3) ...
Setting up libpam0g-dev:amd64 (1.1.8-3.2ubuntu2.3) ...
anhdung3@ubuntu:~$
```

Tiếp theo ta tải libssl-dev bằng lệnh:

```
# sudo apt install libssl-dev
```

```
anhdung3@ubuntu:~$ sudo apt install libssl-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libssl-doc zlib1g-dev
The following NEW packages will be installed:
  libssl-dev libssl-doc zlib1g-dev
0 upgraded, 3 newly installed, 0 to remove and 0 not upgraded.
Need to get 2,591 kB of archives.
After this operation, 10.5 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Tiếp theo ta phải tải gói **riak_2.9.10-1_amd64.deb** theo câu lệnh:

```
# wget https://files-source.tiot.jp/riak/kv/2.9/2.9.10/ubuntu/xenial64/riak_2.9.10-1_amd64.deb
```

```
anhdung3@ubuntu:~$ wget https://files-source.tiot.jp/riak/kv/2.9/2.9.10/ubuntu/xenial64/riak_2.9.10-1_amd64.deb
--2024-05-11 21:33:58-- https://files-source.tiot.jp/riak/kv/2.9/2.9.10/ubuntu/xenial64/riak_2.9.10-1_amd64.deb
Resolving files-source.tiot.jp (files-source.tiot.jp)... 52.15.82.139
Connecting to files-source.tiot.jp (files-source.tiot.jp)|52.15.82.139|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 60642470 (58M) [application/x-debian-package]
Saving to: 'riak_2.9.10-1_amd64.deb'

riak_2.9.10-1_amd64 15%[==>] 9.01M 1.49MB/s eta 62s
```

Giải nén file **riak** vừa mới tải xong bằng lệnh:

```
# sudo dpkg -i riak_2.9.10-1_amd64.deb
```

```
anhdung3@ubuntu:~$ sudo su
root@ubuntu:/home/anhdung3# sudo dpkg -i riak_2.9.10-1_amd64.deb
(Reading database ... 217999 files and directories currently installed.)
Preparing to unpack riak_2.9.10-1_amd64.deb ...
Unpacking riak (2.9.10-1) over (2.9.10-1) ...
Setting up riak (2.9.10-1) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for systemd (229-4ubuntu21.31) ...
Processing triggers for ureadahead (0.100.0-19.1) ...
root@ubuntu:/home/anhdung3#
```

Kiểm tra riak đã được cài hay chưa bằng:

```
# riak start
```

```
# riak ping : nếu nó trả về kết quả pong thì đã cài đặt thành công.
```

```
root@ubuntu:/home/anhdung3
[anhdung3@ubuntu:~$ sudo su
[sudo] password for anhdung3:
root@ubuntu:/home/anhdung3# riak start
!!!!
!!!! WARNING: ulimit -n is 1024; 65536 is the recommended minimum.
!!!!
root@ubuntu:/home/anhdung3# riak ping
pong
root@ubuntu:/home/anhdung3#
```

LibreOffice Calc

4.2. Cấu Hình Riak

Lấy ip máy để cấu hình riak bằng: ip a

```
root@ubuntu:/home/anhdung3# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:7e:14:06 brd ff:ff:ff:ff:ff:ff
    inet 192.168.45.129/24 brd 192.168.45.255 scope global dynamic ens33
        Ubuntu Software 1731sec preferred_lft 1731sec
        brd 192.168.45.255 scope link
        valid_lft forever preferred_lft forever
root@ubuntu:/home/anhdung3#
```

Chỉnh sửa file riak.conf bằng lệnh: vim /etc/riak/riak.conf

```
valid_lft forever preferred_lft forever
root@ubuntu:/home/anhdung3# vim /etc/riak/riak.conf
```

Đổi IP cấu hình (máy của mình có ip là 192.168.45.129 nên máy bạn lấy đúng ip máy mình nhé)

username riak@127.0.0.1 thành riak@**192.168.45.129**

listener.http.internal = **192.168.45.129:8098**

listener.protobuf.internal = **192.168.45.129:8087**

```
##      - an IP/port pair, e.g. 127.0.0.1:10011
listener.http.internal = 192.168.45.129:8098

## listener.protobuf.<name> is an IP address and TCP port that the Riak
## Protocol Buffers interface will bind.
##
## Default: 127.0.0.1:8087
##
## Acceptable values:
##      - an IP/port pair, e.g. 127.0.0.1:10011
listener.protobuf.internal = 192.168.45.129:8087
-- INSERT --
```

294,44

Sửa file advanced.config bằng lệnh: vim /etc/riak/advanced.config

```
root@ubuntu:/home/anhdung# vim /etc/riak/advanced.config
```

Đoạn clusster_mgs, {"**172.0.0.1**"} trong ngoặc sẽ đổi thành ip của máy mình

```
[riak_core,
[
  %% The cluster manager will listen for connections from remote
  %% clusters on this ip and port. Every node runs one cluster
  %% manager, but only the cluster manager running on the
  %% cluster_leader will service requests. This can change as nodes
  %% enter and leave the cluster.
  {cluster_mgr, {"192.168.45.129", 9080} }
]],

[riak_cool]
```

Khởi tạo riak: riak start

```
root@ubuntu:/home/anhdung# vim /etc/riak/riak.conf
root@ubuntu:/home/anhdung# vim /etc/riak/advanced.config
root@ubuntu:/home/anhdung# riak start
!!!!
!!!! WARNING: ulimit -n is 1024; 65536 is the recommended minimum.
!!!!
riak failed to start within 15 seconds,
see the output of 'riak console' for more information.
If you want to wait longer, set the environment variable
WAIT_FOR_ERLANG to the number of seconds to wait.
root@ubuntu:/home/anhdung#
[1] + 0:00 pts/0 S+ 0.0s 0.0s /bin/bash
root@ubuntu:/home/anhdung# rm -rf /var/lib/riak/ring/*
[1]+ 0:00 pts/0 S+ 0.0s 0.0s /bin/bash
```

reboot máy(khởi động lại máy)

Tiếp theo ta khởi động: **riak start**

Kiểm tra cấu hình ip: **riak-admin status**

```
anhdung@ubuntu:~$ sudo su
[sudo] password for anhdung:
root@ubuntu:/home/anhdung# riak start
!!!!
!!!! WARNING: ulimit -n is 1024; 65536 is the recommended minimum.
!!!!
root@ubuntu:/home/anhdung# riak ping
pong
root@ubuntu:/home/anhdung# riak-admin status
1-minute stats for 'riak@192.168.45.129'
-----
cluster_ae_fsm_active : 0
[1] + 0:00 pts/0 S+ 0.0s 0.0s /bin/bash
```

Thành công !!!

4.3. Thao tác cơ bản với Riak

Theo đường dẫn mặc định của riak:

http://localhost:8098/buckets/name_bucket/keys/name_Key

Ví dụ:

<http://localhost:8098/buckets/nhom14/keys/user1>

=> buckets = nhom14 và keys = user1

4.3.1. Add database

addData.sh: thực hiện việc tạo 1000 dữ liệu rồi insert dữ liệu đó vào riak với:

bucket = nhom14

key = user_i(i là số từ 1 tới 1000)

```
value = {  
    name = user_i,  
    age = 20 + i  
}
```

The screenshot shows a terminal window in VS Code with the following content:

```
EXPLORER          ... $ addData.sh •
DATA [WSL: UBUNTU-18.04] $ addData.sh
$ addData.sh
  1
  2  ↵ add_user() {
  3    local key=$1
  4    local name=$2
  5    local age=$3
  6  ↵   curl -X PUT \
  7     -H "Content-Type: application/json" \
  8     -d "{\"name\": \"$name\", \"age\": $age}" \
  9     http://localhost:8008/buckets/nhom14/keys/$key
 10  }
 11
 12
 13 for i in {1..1000}
 14 do
 15   name="User_$i"
 16   age=$((20 + $i))
 17   key="user_$i"
 18   add_user $key $name $age
 19 done
 20
```

Thực hiện việc insert bằng cách chạy file đó

Kiểm tra theo đường dẫn ta thêm vào

Check key = user_1

```
localhost:8098/buckets/nhom14/keys/user_1
1 | {
2 |   "name": "User_1",
3 |   "age": 21
4 | }
```

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
root@dung-slave:/home/anhdung/data# ./addData.shj
bash: ./addData.shj: No such file or directory
root@dung-slave:/home/anhdung/data# ./addData.sh
bash: ./addData.sh: Permission denied
root@dung-slave:/home/anhdung/data# chmod +x addData.sh
root@dung-slave:/home/anhdung/data# ./addData.sh
root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data# curl -X GET http://localhost:8098/buckets/my_bucket/keys/user_1
{"name": "Alice", "age": 25}root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data#
```

So sánh 2 kết quả

=> add data thành công

4.3.2. Insert 1 key -value

Key = user1

value = {"name": "Alice", "age": 25}

Thực hiện chạy câu lệnh: curl -X PUT -H "Content-Type: application/json" -d

'{"name": "Alice", "age": 25}' <http://localhost:8098/buckets/nhom14/keys/user1>

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
Follow link (ctrl + click) bash - data + v ⌂ ⌂ ... ^ x
root@dung-slave:/home/anhdung/data# curl -X PUT -H "Content-Type: application/json" -d '{"name": "Alice", "age": 25}' http://localhost:8098/buckets/nhom14/keys/user1
root@dung-slave:/home/anhdung/data#
```

Không hiện lỗi và kiểm tra lại trên: <http://localhost:8098/buckets/nhom14/keys/user1>

Hiện ra kết quả

localhost:8098/buckets/nhom14/keys/user1

hack Duolingo - Cách họ... tài liệu tiếng anh dịch Drive của tôi - Goo... DBSY230184_22_2... Documentation for... icon 🔍

```
1 {  
2     "name": "Alice",  
3     "age": 25  
4 }
```

=> insert thành công

4.3.3. Tìm kiếm 1 key value trong 1 bucket

Ta sẽ tìm lại key = user1 trong việc insert ở trên

Để lấy 1 value thì ta sử dụng câu lệnh tương ứng với bucket và key tương ứng:

curl -X GET <http://localhost:8098/buckets/nhom14/keys/user1>

```
root@dung-slave:/home/anhdung/data# curl -X PUT -H "Content-Type: application/json" -d '{"name": "Alice", "age": 25}' http://localhost:8098/buckets/nhom14/keys/user1
root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data# curl -X GET http://localhost:8098/buckets/nhom14/keys/user1
{"name": "Alice", "age": 25}root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data#
```

value = {"name": "Alice", "age": 25}

=> thành công

4.3.4. Update dữ liệu

Việc update dữ liệu cũng giống như là việc insert dữ liệu

Ta cứ insert đúng key mà ta cần muốn update

Kiểm tra lại thì kết quả sẽ được thay đổi với key ta vừa mới làm

Ta sẽ insert lại value với key = user1 và value = {"name": "poppy", "age": 26}

```
root@dung-slave:/home/anhdung/data# curl -X GET http://localhost:8098/buckets/nhom14/keys/user1
{"name": "Alice", "age": 25}root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data# curl -X PUT -H "Content-Type: application/json" -d '{"name": "poppy", "age": 26}' http://localhost:8098/buckets/nhom14/keys/user1
root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data# curl -X GET http://localhost:8098/buckets/nhom14/keys/user1
{"name": "poppy", "age": 26}root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data#
```

Kiểm tra lại value của key = user1 thì ta thấy

value = {"name": "Alice", "age": 25}

đã chuyển thành:

value = {"name": "poppy", "age": 26}

=> update thành công

4.3.5. Delete key:

Thực hiện việc xóa key = user1 với bucket = nhom14 bằng lệnh:

curl -X DELETE http://localhost:8098/buckets/nhom14/keys/user1

```
root@dung-slave:/home/anhdung/data# curl -X DELETE http://localhost:8098/buckets/nhom14/keys/user1
root@dung-slave:/home/anhdung/data#
root@dung-slave:/home/anhdung/data# curl -X GET http://localhost:8098/buckets/nhom14/keys/user1
not found
root@dung-slave:/home/anhdung/data#
```

Thực hiện get data với key = user1 ta thấy riak trả về not found

=> Delete thành công

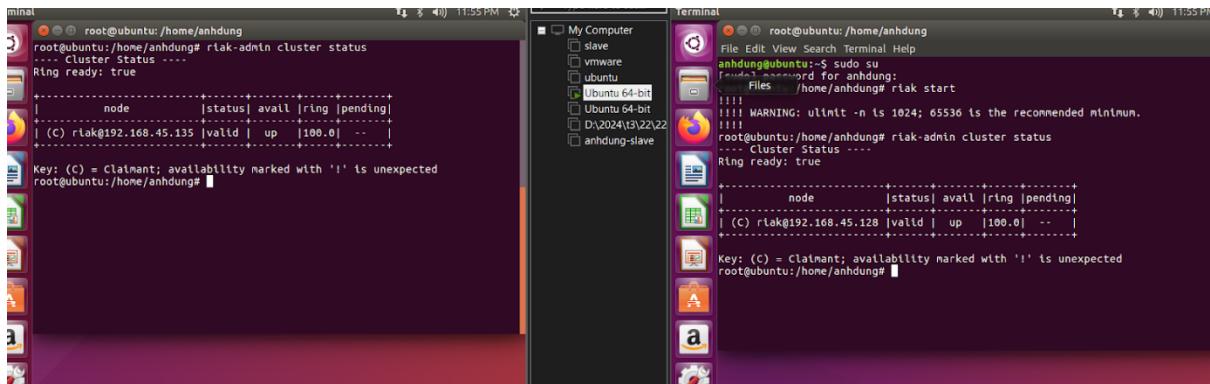
Chương 5. Demo Ứng Dụng Riak Nâng Cao

5.1. Kết nối Cluster-Riak 2 máy:

Riak-admin cluster status: được sử dụng để hiển thị trạng thái hiện tại của cluster Riak, bao gồm thông tin về các node trong cluster, trạng thái của mỗi node và thông tin về vòng riak_core.

riak@192.168.45.135: namenode của máy 1

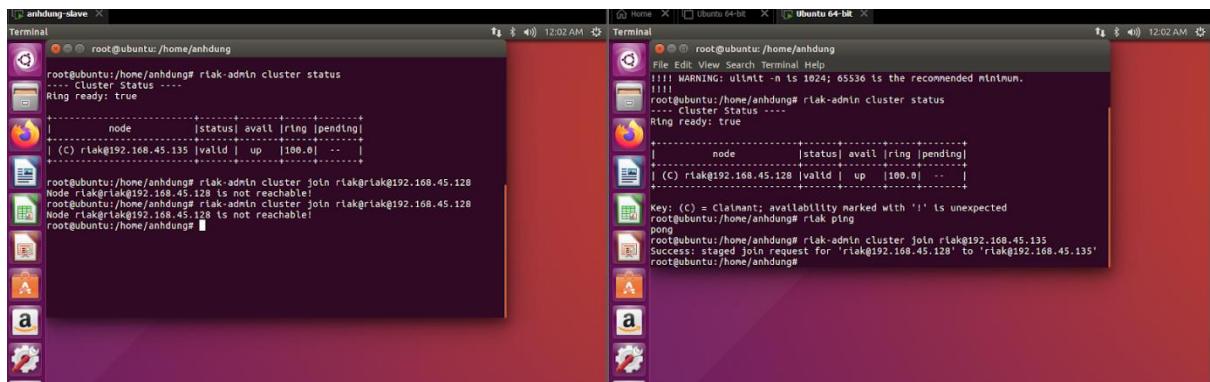
riak@192.168.45.128: namenode của máy 2



Tiến hành kết nối riak giữa máy 1 và máy 2 theo

riak-admin cluster join riak@192.168.45.135

Trong máy 2:



Trong máy 2: sẽ thực hiện câu lệnh “riak-admin cluster plan” để thiết lập kế hoạch kết nối được sử dụng để lập kế hoạch cho việc thay đổi cấu trúc của một cluster Riak. Khi bạn chạy lệnh này, Riak sẽ phân tích cấu trúc hiện tại của cluster và đề xuất các thay đổi cần thiết để cập nhật hoặc mở rộng cluster.

```

Terminal 1: root@ubuntu:/home/anhndung
root@ubuntu:/home/anhndung# riak-admin cluster status
---- Cluster Status ----
Ring ready: true
|-----+-----+-----+-----+
| node |status| avail |ring | pending|
| (C) riak@192.168.45.135 |valid | up | [100.0] ... |
|-----+-----+-----+-----+
root@ubuntu:/home/anhndung# riak-admin cluster join riak@riak@192.168.45.128
Node riak@riak@192.168.45.128 ls not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster join riak@riak@192.168.45.128
Node riak@riak@192.168.45.128 ls not reachable!
root@ubuntu:/home/anhndung#

```



```

Terminal 2: root@ubuntu:/home/anhndung
File Edit View Search Terminal Help
root@ubuntu:/home/anhndung# riak-admin cluster plan
=====
Action Details(s) Staged Changes
join 'riak@192.168.45.128'
=====
NOTE: Applying these changes will result in 1 cluster transition
#####
After cluster transition 1/1
#####
=====
Status Rng Pending Node Membership
valid 0.0% 50.0% 'riak@192.168.45.128'

```

Máy 2: thực hiện câu lệnh “riak-admin cluster commit” để thực hiện plan vừa mới tạo được sử dụng để thực hiện các thay đổi đã được lập kế hoạch trước đó bằng lệnh “riak-admin cluster plan” trên một cluster Riak

```

Terminal 1: root@ubuntu:/home/anhndung
root@ubuntu:/home/anhndung# riak-admin cluster status
---- Cluster Status ----
Ring ready: true
|-----+-----+-----+-----+
| node |status| avail |ring | pending|
| (C) riak@192.168.45.135 |valid | up | [100.0] ... |
|-----+-----+-----+-----+
root@ubuntu:/home/anhndung# riak-admin cluster join riak@riak@192.168.45.128
root@ubuntu:/home/anhndung# riak-admin cluster join riak@riak@192.168.45.128
Node riak@riak@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung#

```



```

Terminal 2: root@ubuntu:/home/anhndung
File Edit View Search Terminal Help
=====
Status Rng Pending Node Membership
valid 0.0% 50.0% 'riak@192.168.45.128'
valid 100.0% 50.0% 'riak@192.168.45.135'
=====
Valid:2 / Leaving:0 / Exiting:0 / Joining:0 / Down:0
WARNING: Not all replicas will be on distinct nodes
Transfers resulting from cluster changes: 32
32 transfers from 'riak@192.168.45.135' to 'riak@192.168.45.128'
root@ubuntu:/home/anhndung# riak-admin cluster commt
cluster changes committed
root@ubuntu:/home/anhndung#

```

Kiểm tra lại việc kết nối của 2 máy bằng “riak-admin cluster status”:

```

Terminal 1: root@ubuntu:/home/anhndung
root@ubuntu:/home/anhndung# riak-admin cluster join riak@riak@192.168.45.128
Node riak@riak@192.168.45.128 ls not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster join riak@riak@192.168.45.128
Node riak@riak@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster status
---- Cluster Status ----
Ring ready: true
|-----+-----+-----+-----+
| node |status| avail |ring | pending|
| riak@192.168.45.128 |valid | up | [35.9] 50.0 |
| (C) riak@192.168.45.135 |valid | up | [64.1] 50.0 |
|-----+-----+-----+-----+
Key: (C) = Clientname; availability marked with '!' is unexpected
root@ubuntu:/home/anhndung#

```



```

Terminal 2: root@ubuntu:/home/anhndung
File Edit View Search Terminal Help
Transfers resulting from cluster changes: 32
32 transfers from 'riak@192.168.45.135' to 'riak@192.168.45.128'
root@ubuntu:/home/anhndung# riak-admin cluster commt
cluster changes committed
root@ubuntu:/home/anhndung# riak-admin cluster status
---- Cluster Status ----
Ring ready: false
|-----+-----+-----+-----+
| node |status| avail |ring | pending|
| riak@192.168.45.128 |valid | up | [32.9] 50.0 |
| (C) riak@192.168.45.135 |valid | up | [67.2] 50.0 |
|-----+-----+-----+-----+
LibreOfficeImpress t; availability marked with '!' is unexpected
root@ubuntu:/home/anhndung#

```

=> Namenode của 2 máy được hiện trên 2 máy suy ra đã kết nối thành công

5.2. Demo xử lý dữ liệu 2 máy

5.2.1. Ghi dữ liệu

Máy 2: thực hiện import data vào với host = ‘192.168.45.128’, http_port = 8098

Đầu vào là file data: customers.csv với bucket sẽ là customers, orders.csv với bucket sẽ là orders\

Tập dữ liệu nói về việc bán xe đạp demo chỉ lấy 2 data để có thể thực hiện việc thao tác đó

file: customers.csv

customers.csv: bao gồm nhiều thông tin của các khách hàng mua đồ tại store

key = customer_id

value = {

first_name: tên

last_name: họ

phone: số điện thoại của khách hàng

email: email của khách hàng

street: địa chỉ nhận hàng

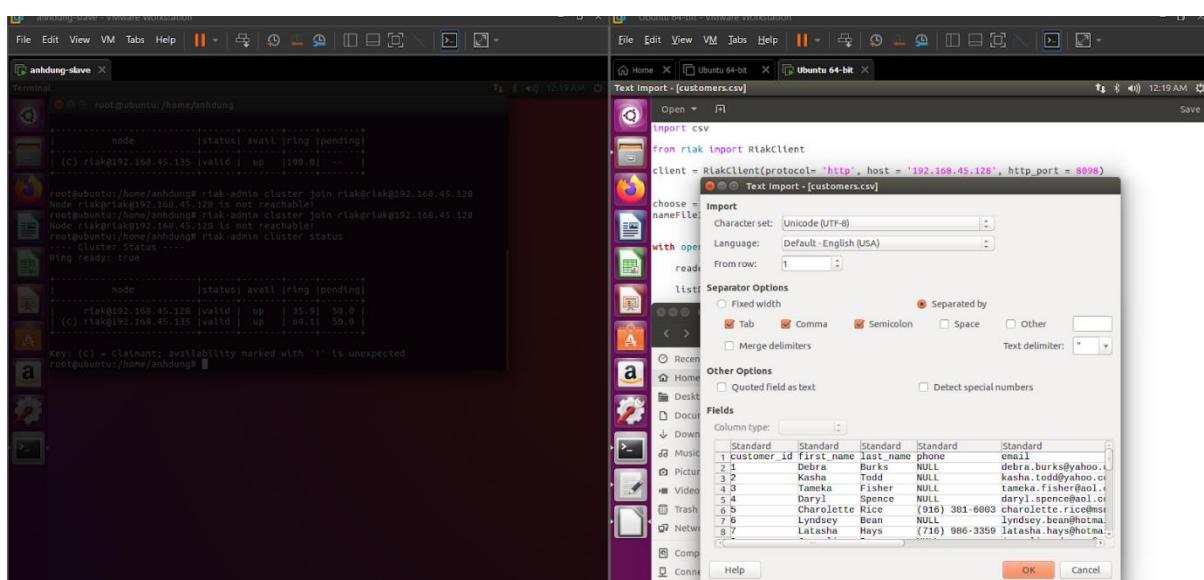
city: thành phố

state: Quận

zip_code: zip code của khách hàng

}

Link: [E-Commerce Sales Dataset \(kaggle.com\)](https://www.kaggle.com/mwaskom/wine-reviews)



file: orders.csv

orders.csv: nói về việc đặt hàng của customers tại store đó bao gồm các thông tin

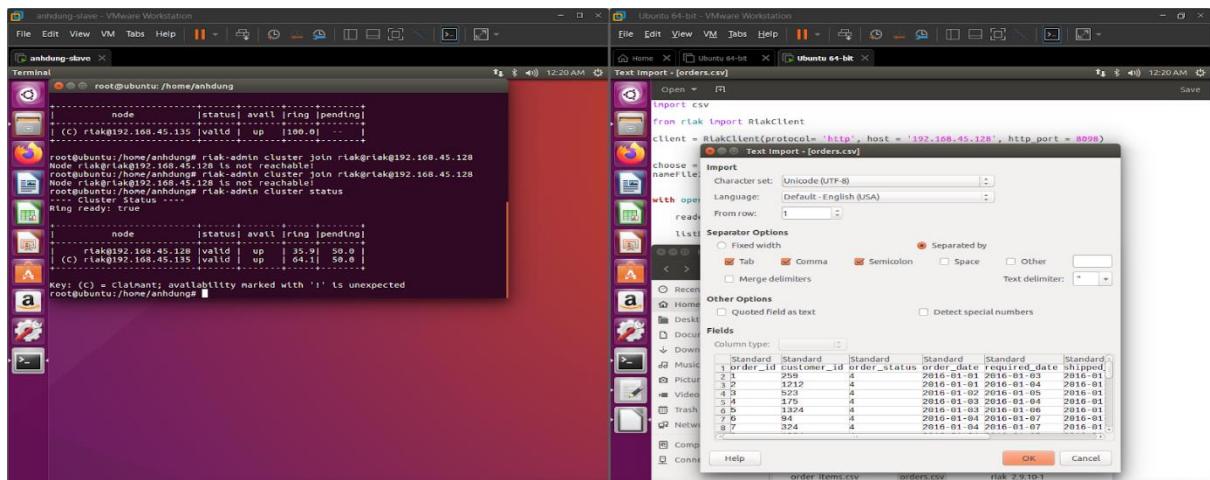
key = order_id

value = {

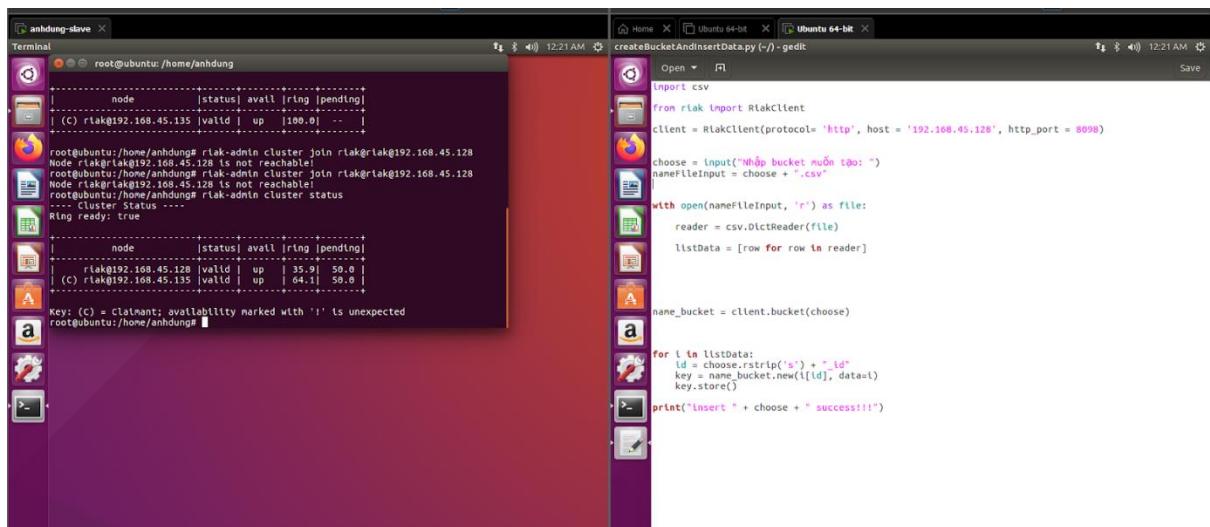
customer_id: id của khách hàng
order_status tình trạng,
order_date: ngày đặt hàng
required_date: ngày yêu cầu
shipped_date: ngày giao hàng
store_id: id của store
staff_id: id của nhân viên lên order đó

}

Link: [E-Commerce Sales Dataset \(kaggle.com\)](https://www.kaggle.com/mwaskom/wine-reviews)



createBucketAndInsertData.py: code này thực hiện việc tạo bucket với đưa dữ liệu vào bucket



Bắt đầu tạo bucket(customers và orders) và đọc file và đưa dữ liệu từ 2 file customers.csv và orders.csv vào 2 bucket đó từ máy 2

```

root@ubuntu:/home/anhndung
+-----+-----+-----+-----+
| node | status | avail | ring | pending |
+-----+-----+-----+-----+
| (C) riak@192.168.45.135 | valid | up | [100.0] | ... |
+-----+-----+-----+-----+
root@ubuntu:/home/anhndung# riak-admin cluster join riak@riak@192.168.45.128
Node riak@riak@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster join riak@riak@192.168.45.128
Node riak@riak@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster status
-- Cluster Status --
Ring ready: true
+-----+-----+-----+-----+
| node | status | avail | ring | pending |
+-----+-----+-----+-----+
| riak@192.168.45.128 | valid | up | [35.9] | 50.0 |
| riak@192.168.45.135 | valid | up | [64.1] | 50.0 |
+-----+-----+-----+-----+
Key: (C) = Claimant; availability marked with '!' is unexpected
root@ubuntu:/home/anhndung#

```

File Edit View Search Terminal Help

```

+-----+-----+-----+-----+
| node | status | avail | ring | pending |
+-----+-----+-----+-----+
| riak@192.168.45.128 | valid | up | [32.8] | 50.0 |
| riak@192.168.45.135 | valid | up | [67.2] | 50.0 |
+-----+-----+-----+-----+
Nhập bucket muốn tạo: customers
Insert customers successfully!
Nhập bucket muốn tạo: orders
Nhập bucket muốn tạo: orders
Insert orders successfully!
root@ubuntu:/home/anhndung#

```

```

name_bucket = client.bucket(choose)

for i in listData:
    id = choose.rstrip('s') + " " + id
    key = name_bucket.new([id], data=i)
    key.store()

print("insert " + choose + " success!!!")

```

5.2.2. ShowData trong 1 bucket

showData.py: trong máy 1 sẽ thực hiện kiểm tra dữ liệu của các bucket vừa được tạo và đưa dữ liệu vào từng các bucket tương ứng trong máy 1

```

showData.py (~) - gedit
Save
from riak import RiakClient
import json
client = RiakClient(protocol='http', host='192.168.45.135', http_port=8090)
nameBucket = input("Nhập nameBucket: ")
bucket = client.bucket(nameBucket)

for key in bucket.get_keys():
    obj = bucket.get(key)
    data = obj.data
    formatted_data = json.dumps(obj.data, indent=4, ensure_ascii=False)
    print(formatted_data)

```

File Edit View Search Terminal Help

```

+-----+-----+-----+-----+
| node | status | avail | ring | pending |
+-----+-----+-----+-----+
| riak@192.168.45.128 | valid | up | [32.8] | 50.0 |
| (C) riak@192.168.45.135 | valid | up | [67.2] | 50.0 |
+-----+-----+-----+-----+
Key: (C) = Claimant; availability marked with '!' is unexpected
root@ubuntu:/home/anhndung# python3 createBucketAndInsertData.py
Nhập bucket muốn kiểm tra: customers
Insert customers successfully!
root@ubuntu:/home/anhndung# root@ubuntu:/home/anhndung# python3 showData.py
Nhập nameBucket: customers
nhập nameBucket: customers
Insert orders successfully!
root@ubuntu:/home/anhndung#

```

```

name_bucket = client.bucket(choose)

for i in listData:
    System Settings.rstrip('s') + " " + id
    key = name_bucket.new([id], data=i)
    key.store()

print("insert " + choose + " success!!!")

```

Bắt đầu kiểm tra trong máy 1

```

root@ubuntu:/home/anhndung# riak-admin cluster join riakr@192.168.45.128
Node riakr@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster join rtaik@192.168.45.128
Node rtaik@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster status
-- Cluster Status --
Ring ready: true
+-----+-----+-----+-----+
| node | status| avail |ring |pending|
+-----+-----+-----+-----+
| riakr@192.168.45.128 | valid | up | 33.9 | 50.0 |
| rtaik@192.168.45.135 | valid | up | 64.1 | 50.0 |
+-----+-----+-----+-----+
Key: (C) = Claimant; availability marked with '!' is unexpected
Nhập nameBucket: customer
root@ubuntu:/home/anhndung# python3 showData.py
Nhập nameBucket: customers

```

The left terminal shows the execution of riak-admin commands to join nodes and check the cluster status. The right terminal shows the creation of a bucket named 'customer' and the execution of a Python script 'showData.py'.

Trong máy 1 sẽ hiện ra kết quả :

```

root@ubuntu:/home/anhndung# riak-admin cluster join riakr@192.168.45.128
Node riakr@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster join rtaik@192.168.45.128
Node rtaik@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster status
-- Cluster Status --
Ring ready: true
+-----+-----+-----+-----+
| node | status| avail |ring |pending|
+-----+-----+-----+-----+
| riakr@192.168.45.128 | valid | up | 32.8 | 50.0 |
| rtaik@192.168.45.135 | valid | up | 67.2 | 50.0 |
+-----+-----+-----+-----+
Key: (C) = Claimant; availability marked with '!' is unexpected
Nhập nameBucket: customers
Insert customers successful!!
root@ubuntu:/home/anhndung# riak-admin cluster join riakr@192.168.45.128
Node riakr@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster join rtaik@192.168.45.128
Node rtaik@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster status
-- Cluster Status --
Ring ready: true
+-----+-----+-----+-----+
| node | status| avail |ring |pending|
+-----+-----+-----+-----+
| riakr@192.168.45.128 | valid | up | 32.8 | 50.0 |
| rtaik@192.168.45.135 | valid | up | 67.2 | 50.0 |
+-----+-----+-----+-----+
Key: (C) = Claimant; availability marked with '!' is unexpected
Nhập nameBucket: orders
Insert orders successful!!
root@ubuntu:/home/anhndung#

```

The left terminal shows the execution of riak-admin commands to join nodes and check the cluster status. The right terminal shows the creation of a bucket named 'orders' and the execution of a Python script 'createBucketAndInsertData.py'.

=> bucket customers đã được thêm dữ liệu từ máy 1 thành công

Tương tự như buckets orders

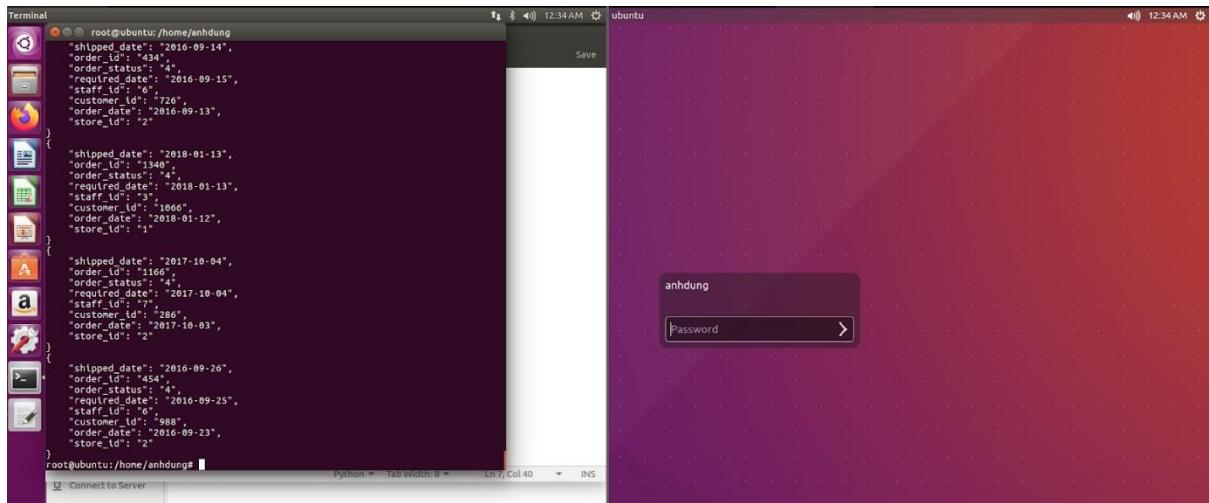
```

root@ubuntu:/home/anhndung# riak-admin cluster join riakr@192.168.45.128
Node riakr@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster join rtaik@192.168.45.128
Node rtaik@192.168.45.128 is not reachable!
root@ubuntu:/home/anhndung# riak-admin cluster status
-- Cluster Status --
Ring ready: true
+-----+-----+-----+-----+
| node | status| avail |ring |pending|
+-----+-----+-----+-----+
| riakr@192.168.45.128 | valid | up | 35.7 | 50.0 |
| rtaik@192.168.45.135 | valid | up | 64.3 | 50.0 |
+-----+-----+-----+-----+
Key: (C) = Claimant; availability marked with '!' is unexpected
Nhập nameBucket: orders

```

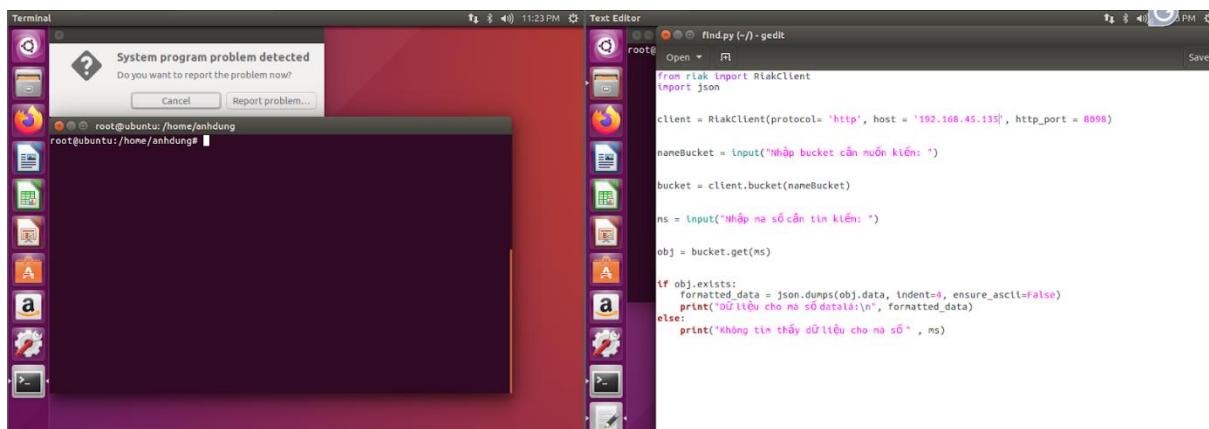
The left terminal shows the execution of riak-admin commands to join nodes and check the cluster status. The right terminal shows the creation of a bucket named 'orders' and the execution of a Python script 'createBucketAndInsertData.py'.

Kết quả trong máy 1 sẽ hiện ra các dữ liệu trong buckets orders

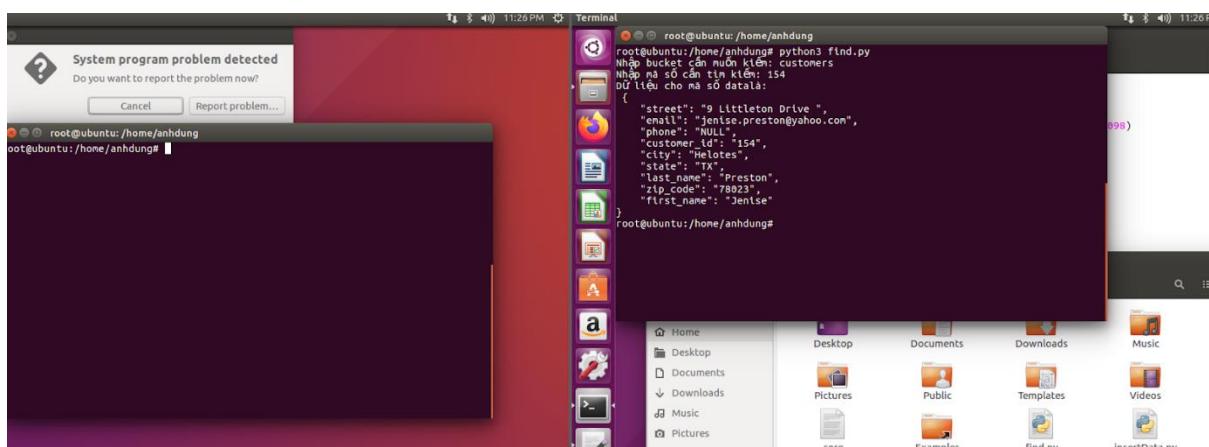


5.2.3. Tìm kiếm 1 value tương ứng với 1 key trong 1 bucket

find.py



ví dụ ta kiểm 1 value theo key = 154 trong bucket customers



key = 154

```

=> value = {

    "street": "9 Littleton Drive ",

    "email": "jenise.preston@yahoo.com",

    "phone": "NULL",

    "customer_id": "154",

    "city": "Helotes",

    "state": "TX",

    "last_name": "Preston",

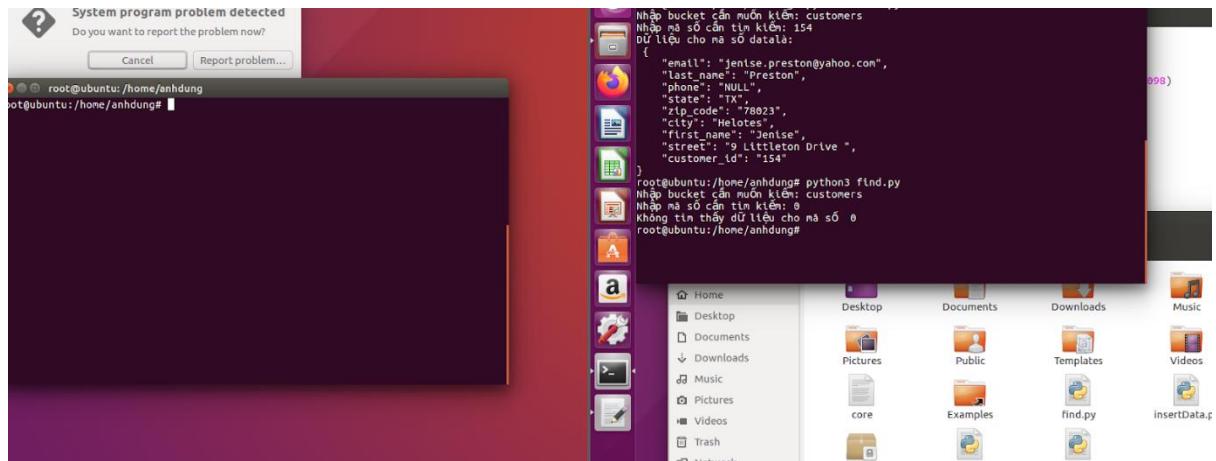
    "zip_code": "78023",

    "first_name": "Jenise"
}

```

Ta thử với 1 key-value không tồn tại trong bucket customers

key = 0



=> kết quả: Không tìm thấy dữ liệu cho mã số 0

5.2.4. Thêm key value vào 1 bucket

insertData.py: thực hiện việc insert 1 key value trong máy 1

```

Text Editor          Terminal
-----|-----|-----|-----|-----|-----|-----|-----|
showData.py          root@ubuntu:/home/anhdung
from riak import RiakClient
client = RiakClient(protocol='http', host = '192.168.45.135', http_port = 8098)
nameBucket = input("Nhập tên bucket muốn sử dụng: ")
bucket = client.bucket(nameBucket)

ns = input("Nhập ns số: ")

def dataOrder():
    customer_id = input("customer_id: ")
    bucketCustomer.get(customer_id).exists:
        order_status = input("order_status: ")
        order_date = input("order_date: ")
        required_date = input("required_date: ")
        shipped_date = input("shipped_date: ")
        new_data = {
            'customer_id': customer_id,
            'order_status': order_status,
            'order_date': order_date,
            'required_date': required_date,
            'shipped_date': shipped_date
        }
        return new_data
    print("Không tồn tại khách hàng đó mời bạn nhập lại!!!\n")
    return None

insertData.py          root@ubuntu:/home/anhdung
Save
-----|-----|-----|-----|-----|-----|-----|-----|
node | status | avail | ring | pending |
-----|-----|-----|-----|-----|
riak@192.168.45.128 | valid | up | 32.8 | 50.0 |
| (C) riak@192.168.45.135 | valid | up | 67.2 | 50.0 |
-----|-----|-----|-----|-----|
Key: (C) = claimant; availability marked with '!' is unexpected
Nhập bucket muốn tạo: customers
Insert customers success!!!
root@ubuntu:/home/anhdung#
root@ubuntu:/home/anhdung# python3 showData.py
Nhập tên bucket muốn sử dụng: orders
Insert orders success!!!
root@ubuntu:/home/anhdung#
-----|-----|-----|-----|-----|-----|-----|-----|
name_bucket = client.bucket(choose)
Amazon
for i in listData:
    id = choose.rstrip('s') + "_id"
    key = name_bucket.new(i[id], data=i)
    key.store()
print("insert " + choose + " success!!!")
-----|-----|-----|-----|-----|-----|-----|-----|

```

Thực hiện insert:

Key = kh001

value = {

first_name: anhdung

last_name: gnuyen

phone: 0878888123

email: dungnguyen@gmail.com

city: HCM

state: Quan 7}

```

showData.py          root@ubuntu:/home/anhdung
from riak import RiakClient
client = RiakClient(protocol='http', host = '192.168.45.135', http_port = 8098)
nameBucket = input("Nhập tên bucket muốn sử dụng: ")
bucket = client.bucket(nameBucket)
root@ubuntu:/home/anhdung# python3 insertData.py
Nhập ns số : kh001
Nhập tên: anhdung
last_name: gnuyen
phone: 0878888123
email: dungnguyen@gmail.com
city: HCM
city: Quan 7
Đã lưu dữ liệu vào Riak.
root@ubuntu:/home/anhdung#
-----|-----|-----|-----|-----|-----|-----|-----|
insertData.py          root@ubuntu:/home/anhdung
Save
-----|-----|-----|-----|-----|-----|-----|-----|
node | status | avail | ring | pending |
-----|-----|-----|-----|-----|
riak@192.168.45.128 | valid | up | 32.8 | 50.0 |
| (C) riak@192.168.45.135 | valid | up | 67.2 | 50.0 |
-----|-----|-----|-----|-----|
Key: (C) = Claimant; availability marked with '!' is unexpected
Nhập bucket muốn tạo: customers
Insert customers success!!!
root@ubuntu:/home/anhdung#
root@ubuntu:/home/anhdung# python3 showData.py
Nhập tên bucket muốn sử dụng: orders
Insert orders success!!!
root@ubuntu:/home/anhdung#
-----|-----|-----|-----|-----|-----|-----|-----|
name_bucket = client.bucket(choose)
Amazon
for i in listData:
    id = choose.rstrip('s') + "_id"
    key = name_bucket.new(i[id], data=i)
    key.store()
print("insert " + choose + " success!!!")
-----|-----|-----|-----|-----|-----|-----|-----|

```

Kiểm tra việc insert với key = kh001 trong máy 2

find.py: là việc tìm kiếm 1 key value trong 1 bucket được nhập từ bàn phím

```

Terminal 1: showData.py
from riaak import RiaakClient

client = RiaakClient(protocol='http', host = '192.168.45.135', http_port = 8098)
nameBucket = input("Nhập tên bucket muốn sử dụng: ")
bucket = client.bucket(nameBucket)
root@ubuntu:/home/anhndung
Nhập tên bucket muốn sử dụng: customers
Nhập mã số : kh001
first_name: anhndung
last_name: gnuyen
phone: 0978888123
email: dungnguyen@gmail.com
city: HCM
state: Quan 7
Dữ liệu đã được ghi vào Riaak.
root@ubuntu:/home/anhndung# 

Terminal 2: find.py
from riaak import RiaakClient
import json

client = RiaakClient(protocol='http', host = '192.168.45.128', http_port = 8098)

nameBucket = input("Nhập bucket cần muốn kiểm: ")

bucket = client.bucket(nameBucket)

ns = input("Nhập mã số cần tìm kiếm: ")

obj = bucket.get(ns)

if obj.exists:
    formatted_data = json.dumps(obj.data, indent=4, ensure_ascii=False)
    print("Dữ liệu cho mã số data là:\n", formatted_data)
else:
    print("Không tìm thấy dữ liệu cho mã số", ns)

```

Thực hiện việc chạy và tìm kiếm trong bucket customers với key = kh001

```

Terminal 1: showData.py
from riaak import RiaakClient

client = RiaakClient(protocol='http', host = '192.168.45.135', http_port = 8098)
nameBucket = input("Nhập tên bucket muốn sử dụng: ")
bucket = client.bucket(nameBucket)
root@ubuntu:/home/anhndung
Nhập tên bucket muốn sử dụng: customers
Nhập mã số : kh001
first_name: anhndung
last_name: gnuyen
phone: 0978888123
email: dungnguyen@gmail.com
city: HCM
state: Quan 7
Dữ liệu đã được ghi vào Riaak.
root@ubuntu:/home/anhndung# 

Terminal 2: find.py
root@ubuntu:/home/anhndung$ sudo su
[sudo] password for anhndung:
root@ubuntu:/home/anhndung# riaak start
!!!!
!!!! WARNING: ulimit -n 1024; 65536 is the recommended minimum.
root@ubuntu:/home/anhndung# python3 find.py
Nhập bucket cần muốn kiểm: customers
Nhập mã số cần tìm kiếm: kh001
Dữ liệu cho mã số data là:
{
    "state": "Quan 7",
    "email": "dungnguyen@gmail.com",
    "phone": "0978888123",
    "first_name": "anhndung",
    "city": "HCM",
    "last_name": "gnuyen"
}
root@ubuntu:/home/anhndung# 
print("Không tìm thấy dữ liệu cho mã số", ns)

```

Hiện ra đúng với dữ liệu mà ta mới thêm vào được từ máy 1

5.2.5. Sửa value tương ứng với 1 key trong 1 bucket

updateData.py: thực hiện việc update 1 value tương ứng với 1 key và 1 bucket được nhập vào từ bàn phím

The screenshot shows a dual-monitor setup. The left monitor displays a Python code editor with several tabs open, including `showData.py`, `insertData.py`, and `updateData.py`. The code involves interacting with a Riak database using the `riak` library. The right monitor shows a terminal window running as root on an Ubuntu system. The user has run `sudo su` and entered their password. They then ran `riak start`, which outputted a warning about the `ulimit -n` setting. The user then ran `python3 find.py` and provided a bucket name (`customers`). The terminal shows the search results for a specific key (`Quan Nguyen`) and ends with the command `root@ubuntu:/home/anhdung#`. A context menu is visible over the terminal window, with the option "System Settings" highlighted.

Thực hiện chạy code updateData.py với bucket = customers và key =kh001(đây là dữ liệu mới được insert vào từ máy 1)

```
showData.py      insertData.py      updateData.py
from riak import RiakClient
client = RiakClient(protocol='http', host = '192.168.45.135', http_port = 8098)
nameBucket = input("Nhập tên bucket: ")
bucket = client.bucket(nameBucket)

obj = client.get(nameBucket)
klient = obj.new_client()
print("Nhập tên khách hàng cần sửa đổi: customers")
Nhập mã số : kho01
first_name: anhdung
last_name: gnuyen
phone: 0878888123
email: dungnguyen@gmail.com
city: HCM
state: Quan 7
Đã thêm và lưu ghi vào Riak.
root@ubuntu:/home/anhndung# python3 updateData.py
Nhập tên bucket: customers
Nhập mã số cần cập nhật: kho01
Nhập tên khách hàng cần cập nhật: anhdung
{"state": "Quan 7", "phone": "0878888123", "first_name": "anhndung", "email": "dungnguyen@gmail.com", "city": "HCM", "last_name": "gnuyen"}
Nhập tên mới của khách hàng: Vo
Nhập số điện thoại mới của khách hàng: 0123213
Nhập email mới của khách hàng: vo@gmail.com
Nhập thành phố mới của khách hàng: Ha noi
Nhập mã số mới của khách hàng: Cau giay
Thông tin của customer đã được cập nhật thành công.
Root@ubuntu:/home/anhndung#
```

```
root@ubuntu:/home/anhndung# riak start
!!!! WARNING: ulimit -n ts 1024; 65536 ls is the recommended minimum.
root@ubuntu:/home/anhndung# python3 find.py
Nhập bucket cần muôn kiểm: customers
Nhập mã số cần tìm kiếm: kho01
Đã lấy cho mã số data:
[{"id": "kho01", "first_name": "anhndung", "last_name": "gnuyen", "phone": "0878888123", "city": "HCM", "state": "Quan 7", "email": "dungnguyen@gmail.com", "last_update": "2018-05-10T10:45:00Z"}]
root@ubuntu:/home/anhndung#
```

```
print("Không tìm thấy dữ liệu cho mã số", ms)
System Settings
```

Kiểm tra lại dữ liệu trong máy 2 mới update từ máy 1

```
anhdung-slave
```

```
Ubuntu 64-bit
```

```
Ubuntu 64-bit
```

```
Terminal
```

```
root@ubuntu:/home/anhdung
Nhập bucket cần muôn kiểm: customers
Nhập mã số cần tìm kiếm: kho01
Đữ liệu cho mã số data1:
{
    "state": "Quan 7",
    "email": "dungnguyen@gmail.com",
    "phone": "0878886123",
    "first_name": "anhdung",
    "city": "HCM",
    "last_name": "nguyen"
}
root@ubuntu:/home/anhdung# python3 find.py
Nhập bucket cần muôn kiểm: customers
Nhập mã số cần tìm kiếm: kho01
Đữ liệu cho mã số data1:
{
    "last_name": "pham",
    "state": "Cau Giay",
    "first_name": "Tran",
    "email": "vovanhai.com",
    "city": "Hanoi",
    "phone": "0123456789"
}
root@ubuntu:/home/anhdung# print("Không tìm thấy dữ liệu cho mã số", ns)
```

So sánh 2 dữ liệu tìm kiếm lúc nãy và mới tìm kiếm thì cùng 1 key nhưng dữ liệu đã được thay đổi từ máy 1

=> update thành công

Tài Liệu Tham Khảo

- 1 <https://en.wikipedia.org/wiki/Riak>
- 2 <https://www.infoq.com/news/2011/12/andy-gross-riak-database/>
- 3 <https://www.infoq.com/news/2011/12/andy-gross-riak-database/>
- 4 <https://www.naukri.com/code360/library/riak-key-value-database>
- 5 [Backend Configuration \(riak.com\)](#)
- 6 https://www.bing.com/search?pglt=41&q=riak+backend&cvid=d312f1351c384b19a9a1341f0f9bbfeb&gs_lcrp=EgZjaHJvbWUqBggAEEUYOzIGCAAQRRg70gEIMzMzOWowajGoAgCwAgA&FORM=ANNTA1&PC=ASTS