Alternative to relational database system - NoSQL

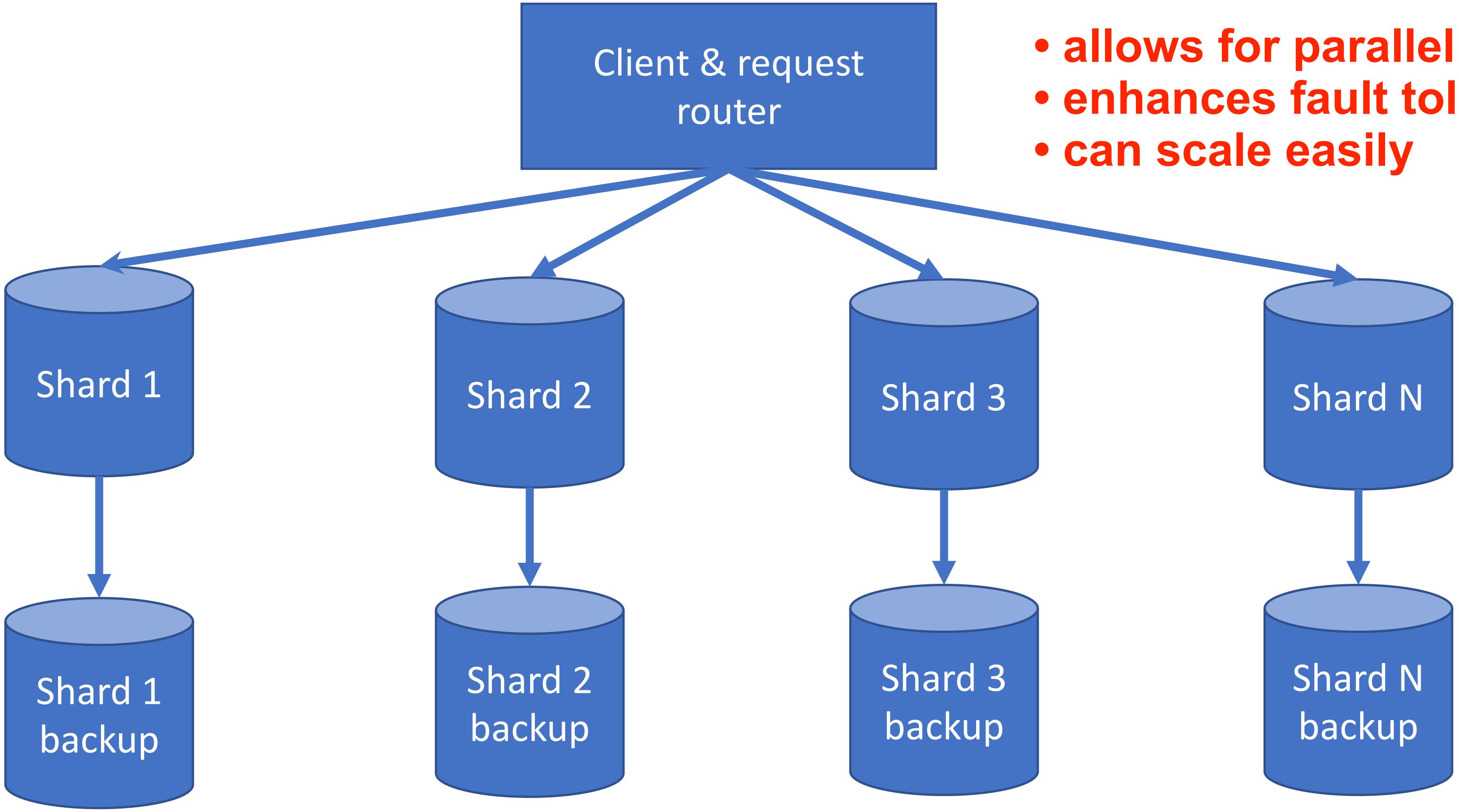
Not Only SQL

Bernard Lee Kok Bang

Do we really need SQL?

- High-transaction queries are probably pretty simple once de-normalized [such as using JSON file format]
- A simple get/put API may meet our needs [*“given this customer ID, give me back this list of information”*]
- Looking up values for a given key is simple, fast, and scalable [at high level]
- Questions:
    - *What’s the item information associated with this item?*
    - *What pages have this customer looked at?*
- Don’t need a rich query language for answering the above question
- Don’t need big fancy relational database; instead, we need more scalable system that can easily horizontally partitioned for given key ranges

NoSQL database sample architecture

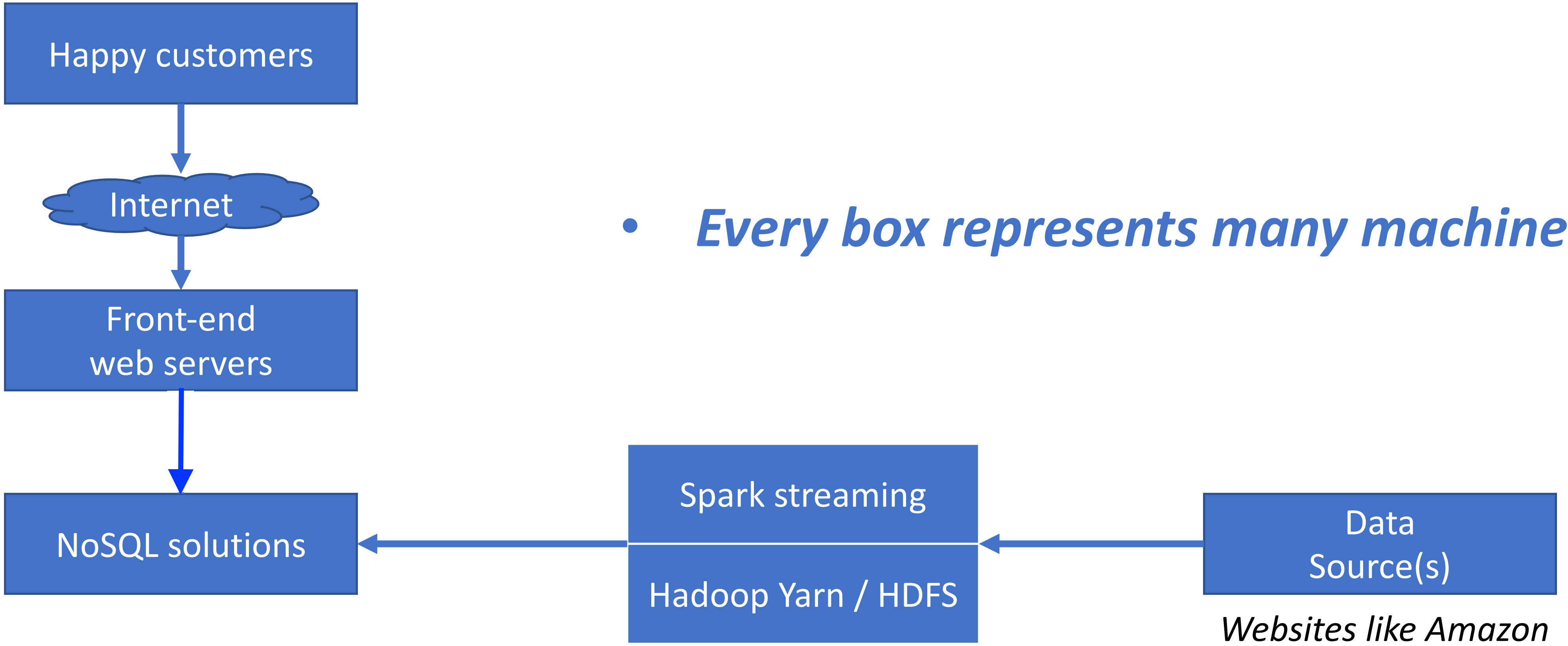


- **allows for parallel processing**
- **enhances fault tolerance**
- **can scale easily**

Use the right tool for the job

- For *analytic queries*, Hive, Pig, Spark, etc. work great [business report]
- Exporting data to MySQL is plenty fast for most applications too
- But if we work at ***giant scale*** [Amazon, Google…] - export our data to a non-relational database for fast and scalable serving of that data to web applications, etc.

Sample application architecture (simplified)



Websites like Amazon

- ***Every box represents many machines***

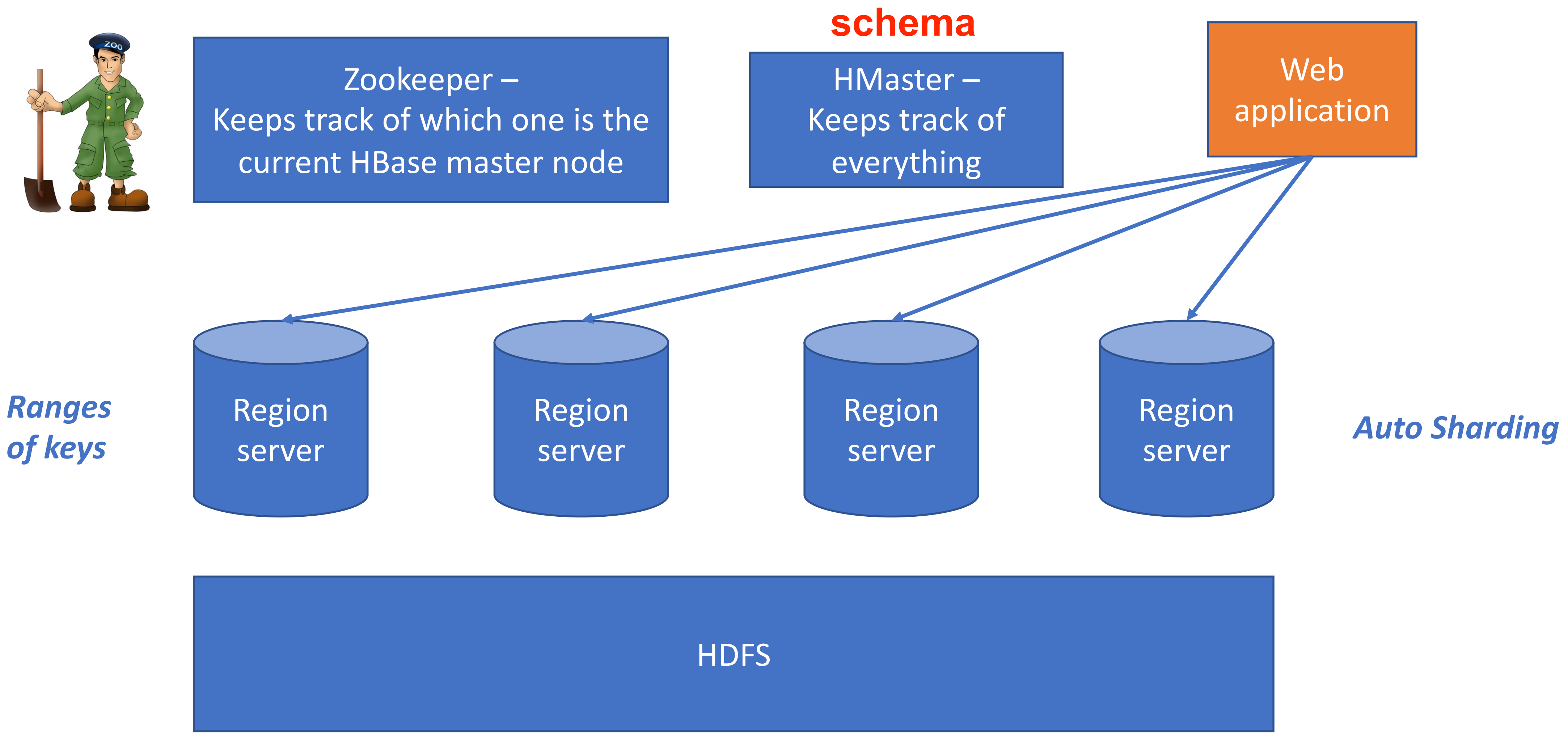
HBase

- Non-relational, scalable database built on HDFS
- Expose massive data sitting on HDFS to web service or web application [that can operate very quickly and high scale]
- Don’t have a query language
- Have an API that can quickly answer the question
  - “what are the values for this key”
  - “store this value for this key”

CRUD operations

- Create
- Read
- Update
- Delete
- No query language, only CRUD API
- Offers transactional access -> ***FAST & LARGE SCALE***!!!

HBase architecture



Zookeeper –
Keeps track of which one is the current HBase master node

schema

HMaster –
Keeps track of everything

Web application

Ranges of keys

Region server

Region server

Region server

Region server

Auto Sharding

HDFS

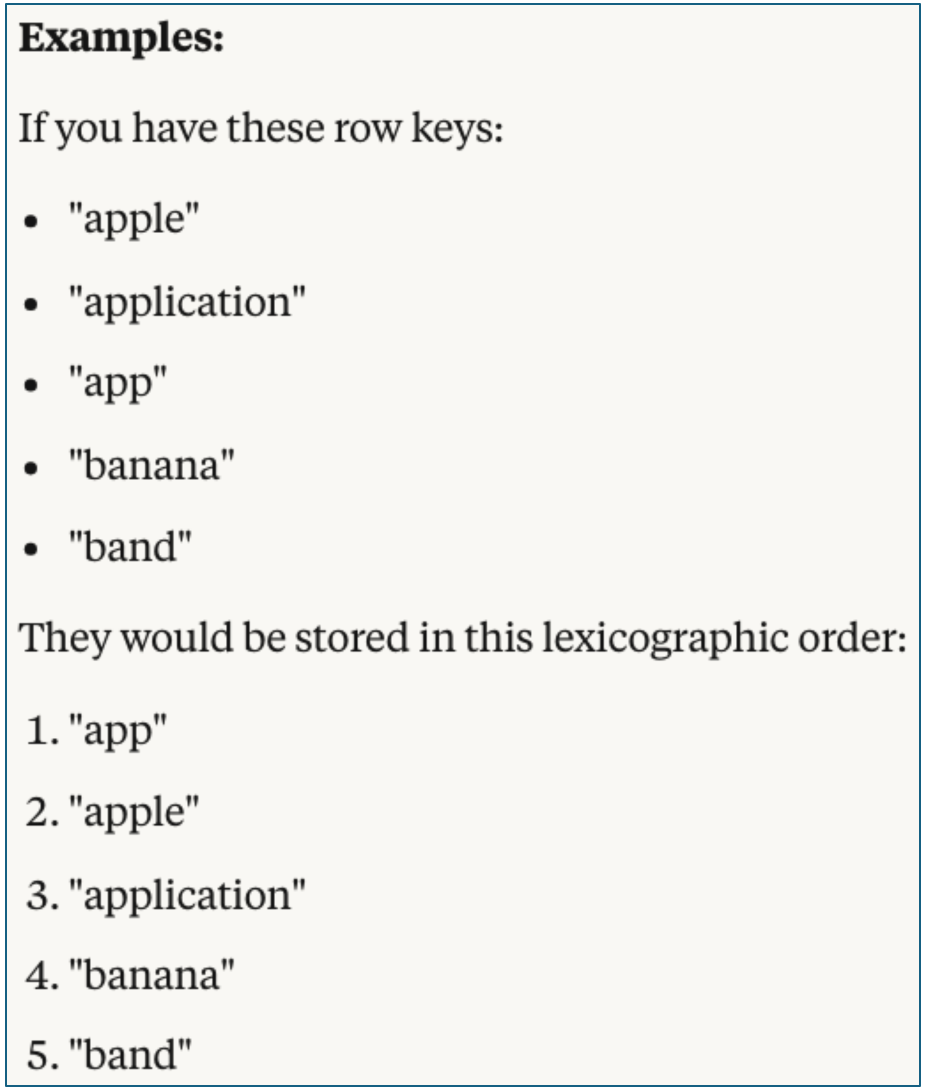
HBase Data Model

- HBase organizes data as a large, sparse, distributed table.
- Think of it like a massive spreadsheet where:
    - Rows can have millions of columns
    - Most cells are empty (sparse)
    - Columns don't need to be predefined
    - Data is stored across multiple machines

Key Components of HBase Data Model

- ***Row Key***:
  - like a primary key in relational databases, but it's the only way to uniquely identify a row.
  - Rows are automatically sorted by row key ***lexicographically***.
- ***Column Families***:
  - groups of related columns that are stored together physically.
  - must define column families when creating a table, but individual columns within families can be added dynamically.
- ***Columns (Qualifiers)***:
  - Within each column family, we can have any number of columns.
  - The full column name is "family:qualifier" (e.g., "***personal:name***").

"Lexicographical" in HBase



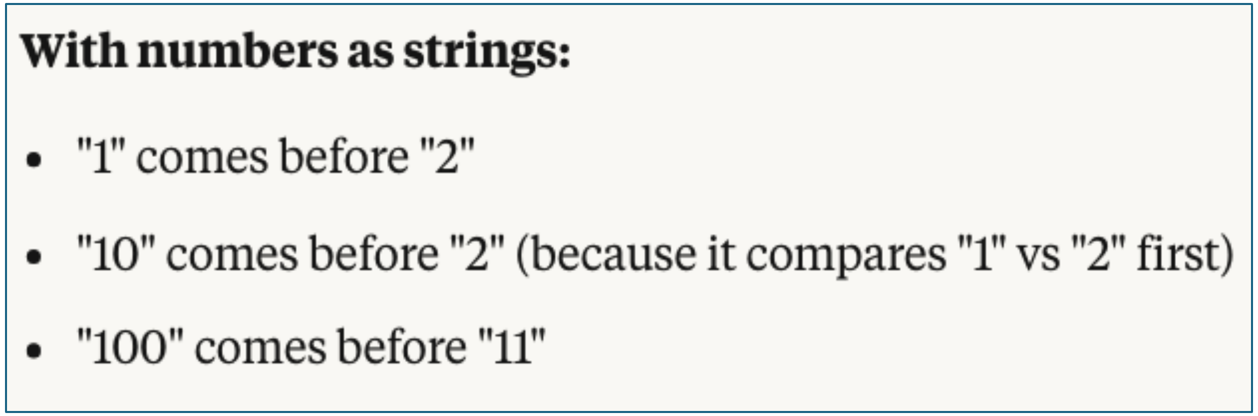
Examples:

If you have these row keys:

- "apple"
- "application"
- "app"
- "banana"
- "band"

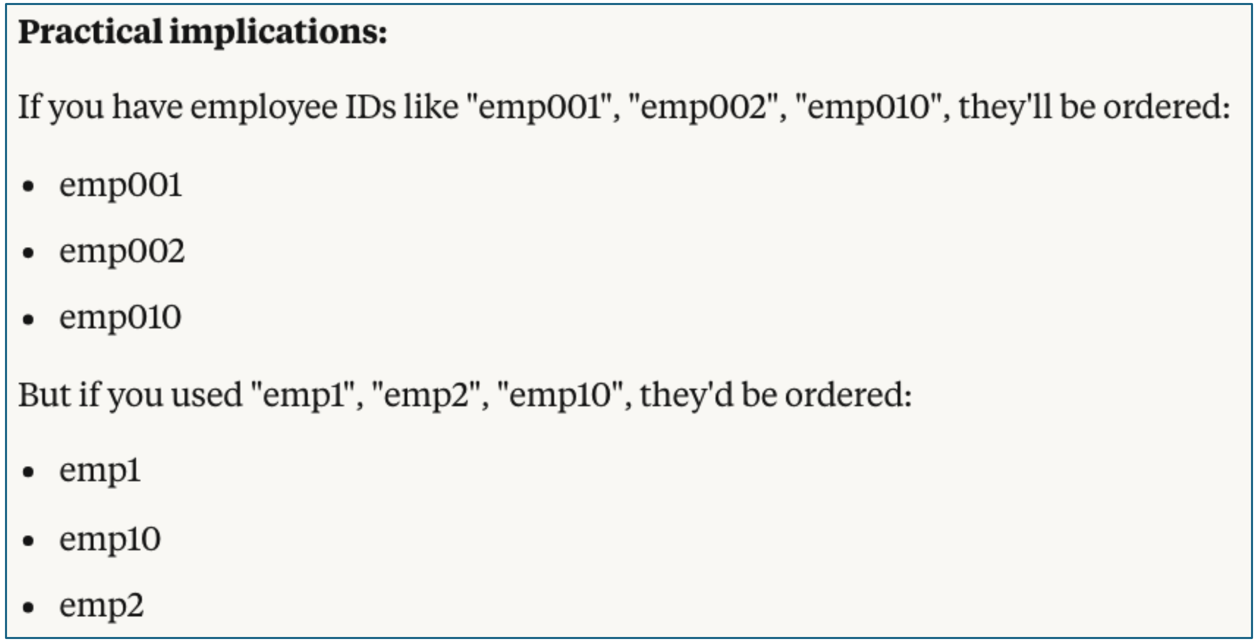
They would be stored in this lexicographic order:

1. "app"
2. "apple"
3. "application"
4. "banana"
5. "band"



With numbers as strings:

- "1" comes before "2"
- "10" comes before "2" (because it compares "1" vs "2" first)
- "100" comes before "11"



Practical implications:

If you have employee IDs like "emp001", "emp002", "emp010", they'll be ordered:

- emp001
- emp002
- emp010

But if you used "emp1", "emp2", "emp10", they'd be ordered:

- emp1
- emp10
- emp2

Using HBase Shell - PuTTY

```
#Opens the HBase shell
hbase shell

#Lists down all the tables present in HBase
list

#Creates a new table
create ‘newtbl’, ‘knowledge’

#Checks if the table was created
describe ‘newtbl’

#Checks the status of HBase
status ‘summary’

#Put some data into the 'newtbl'
put ‘newtbl’, ‘r1’, ‘knowledge:sports’, ‘cricket’
put ‘newtbl’, ‘r1’, ‘knowledge:science’, ‘chemistry’
put ‘newtbl’, ‘r1’, ‘knowledge:science’, ‘physics’
put ‘newtbl’, ‘r2’, ‘knowledge:economics’, ‘macro economics’
put ‘newtbl’, ‘r2’, ‘knowledge:music’, ‘pop music’
```

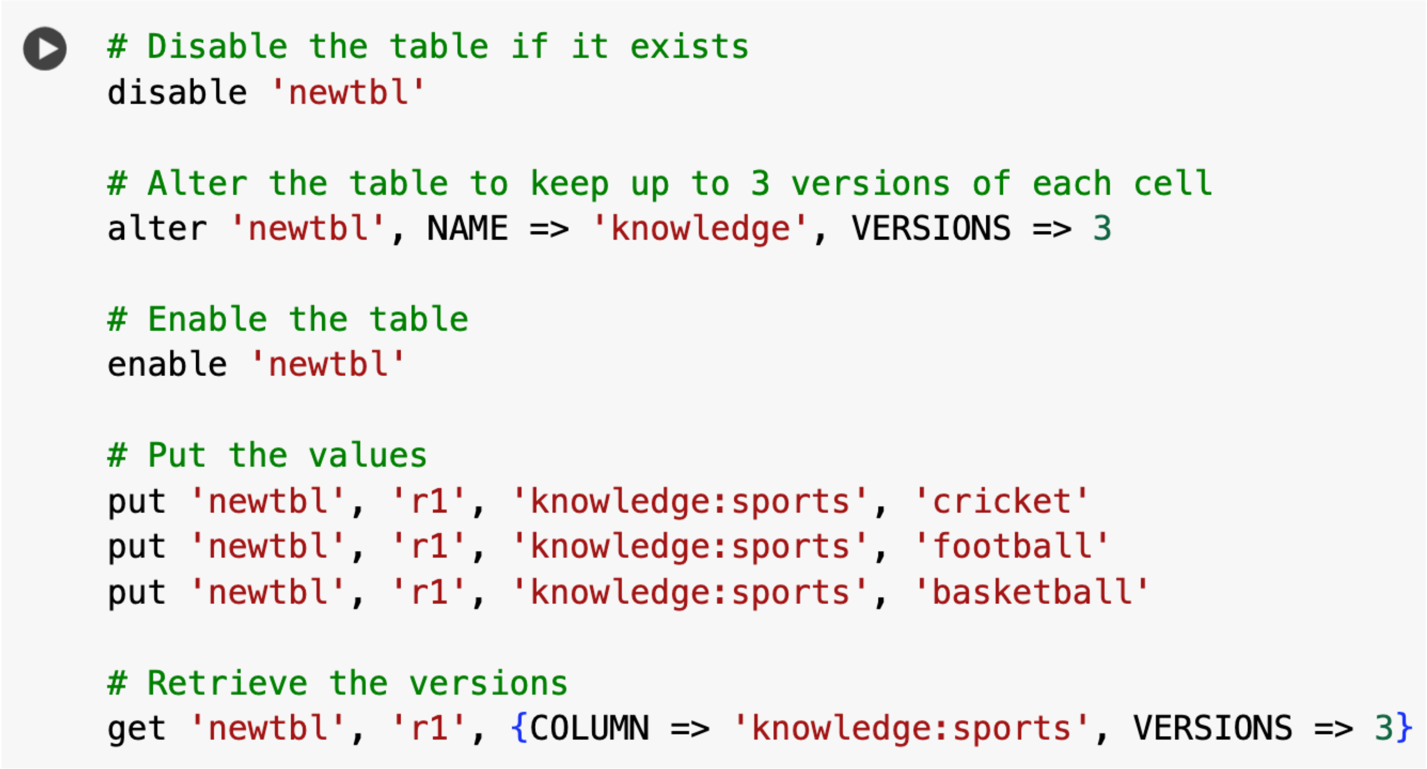
Using HBase Shell – PuTTY (cont…)

```
#List the contents of the 'newtbl'
scan ‘newtbl’

#Checks if the table is enabled
is_enabled ‘newtbl’

#Disables the table
disable ‘newtbl’

#Lists the contents of the 'newtbl'. Note that this will throw an error as the table is disabled.
scan ‘newtbl’

#Updates column family in the table
alter ‘newtbl’, ‘test_info’

#Enables the table
enable ‘newtbl’

#Checks the column families after updating
describe ‘newtbl’

#Extracts the values for r1 in the 'newtbl'
get ‘newtbl’, ‘r1’
```

Using HBase Shell – PuTTY (cont…)

```
#Adds new information to r1 for economics. Note that this will update the table but will
not override the information
put ‘newtbl’, ‘r1’, ‘knowledge:economics’, ‘market economics’

#Displays the results for r1
get ‘newtbl’, ‘r1’

#Count number of rows in 'newtbl'
count 'newtbl'

# Delete the 'newtbl'. Note: Error because the 'newtbl' is still enabled
drop 'newtbl'

#Disable first, then only delete the the 'newtbl'
disable 'newtbl'

# Check if the 'newtbl' still available
list
```

What if I wanted to add in more values in same row?



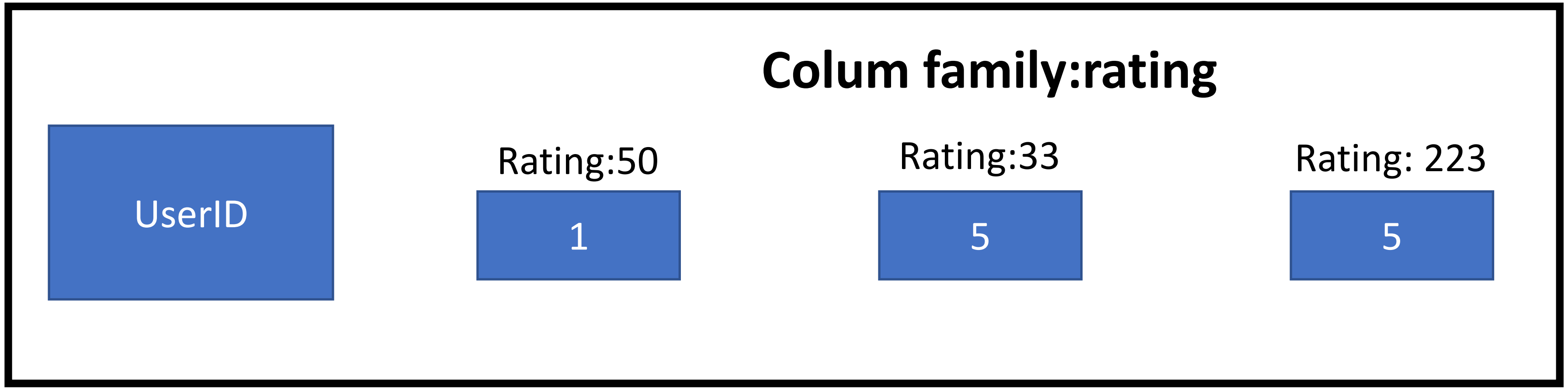
```
# Disable the table if it exists
disable 'newtbl'

# Alter the table to keep up to 3 versions of each cell
alter 'newtbl', NAME => 'knowledge', VERSIONS => 3

# Enable the table
enable 'newtbl'

# Put the values
put 'newtbl', 'r1', 'knowledge:sports', 'cricket'
put 'newtbl', 'r1', 'knowledge:sports', 'football'
put 'newtbl', 'r1', 'knowledge:sports', 'basketball'

# Retrieve the versions
get 'newtbl', 'r1', {COLUMN => 'knowledge:sports', VERSIONS => 3}
```

Using HBase Shell – PuTTY (cont…)

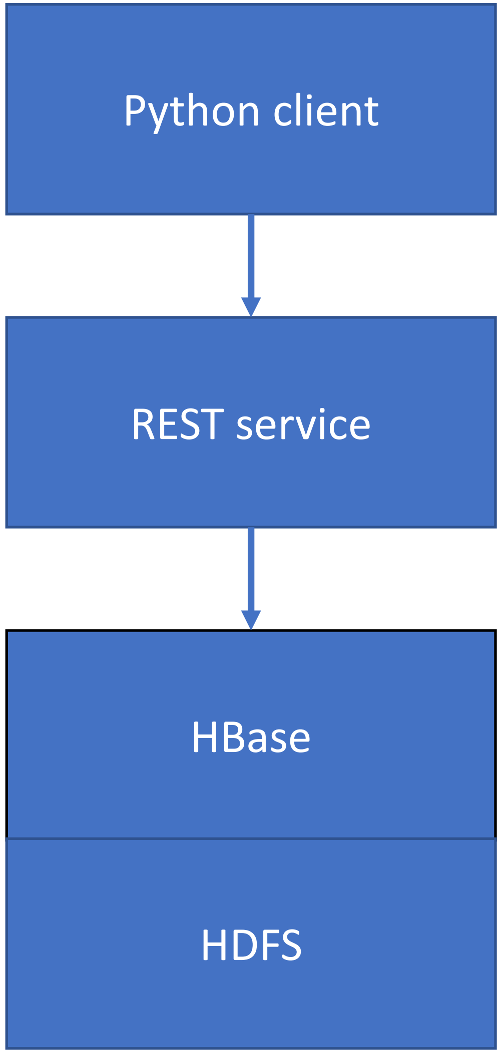
```
#Check current HBase user
whoami

#Different HBase table command usages and its syntaxes
table_help

#HBase status
status

#HBase version
version
```

Let’s play with HBase

- Create a HBase table for *movie ratings grouped by user*
- Show how we can quickly query for individual users
- Good example of sparse data



How are we doing it?



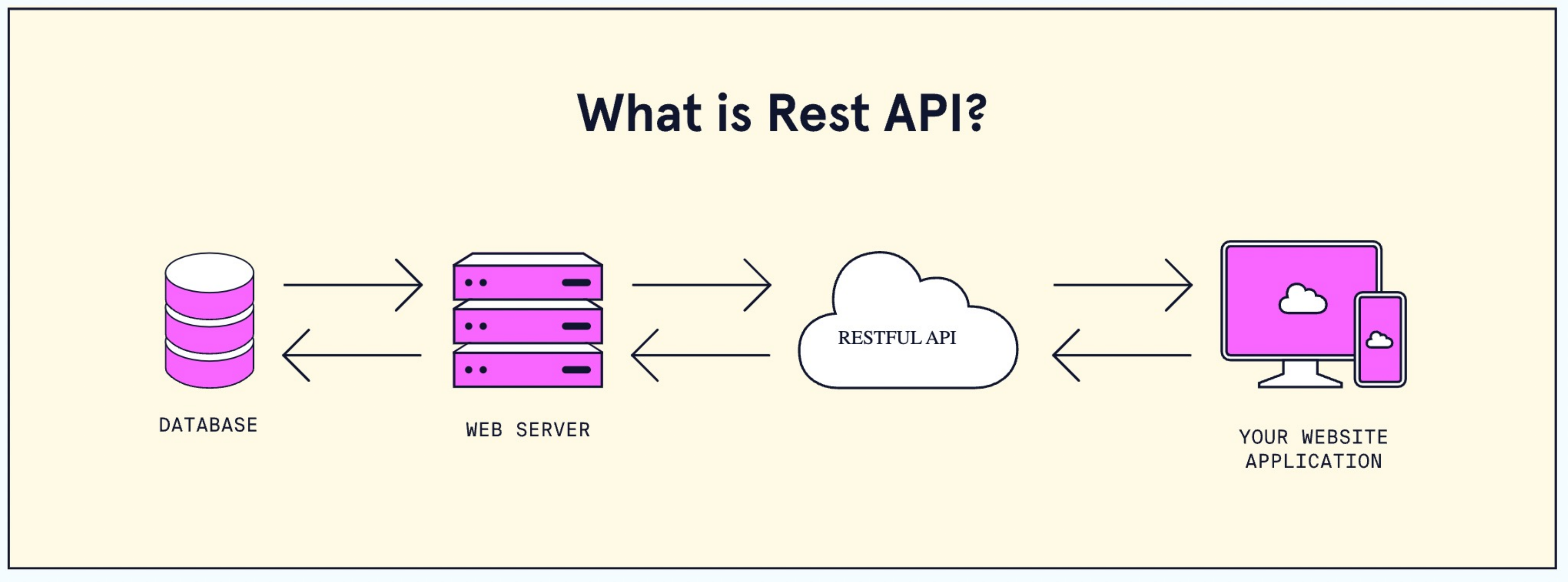
Python client

REST service

HBase

HDFS

REST API



- providing standards between computer systems on the web, making it easier for systems to communicate with each other

https://bit.ly/3whbwMF

Change some settings: Port 8000 & Ambari

- Open a port so that Python script can talk to REST service

1. Right click Horthonworks Docker Sandbox on VirtualBox
2. Choose Settings, Network, Advanced, Port Forwarding
3. Click the ADD button and add in the following details

   →Name: ***HBase REST***; Protocol: ***TCP***; Host IP: ***127.0.0.1***; Host Port:***8000***; Guest Port: ***8000***
4. If Port 8000 has been occupied, then just change the Name to ***HBase REST***
5. Log in to Ambari using ***admin*** username
6. Choose ***HBase*** [left panel], and ***start*** the HBase under ***Service Action***

Change some settings: PuTTY

Starts REST server sitting on top of HBase and HDFS

1. Login as maria_dev

```
su root
Password:
/usr/hdp/current/hbase-master/bin/hbase-daemon.sh start rest -p 8000 --infoport 8001
```

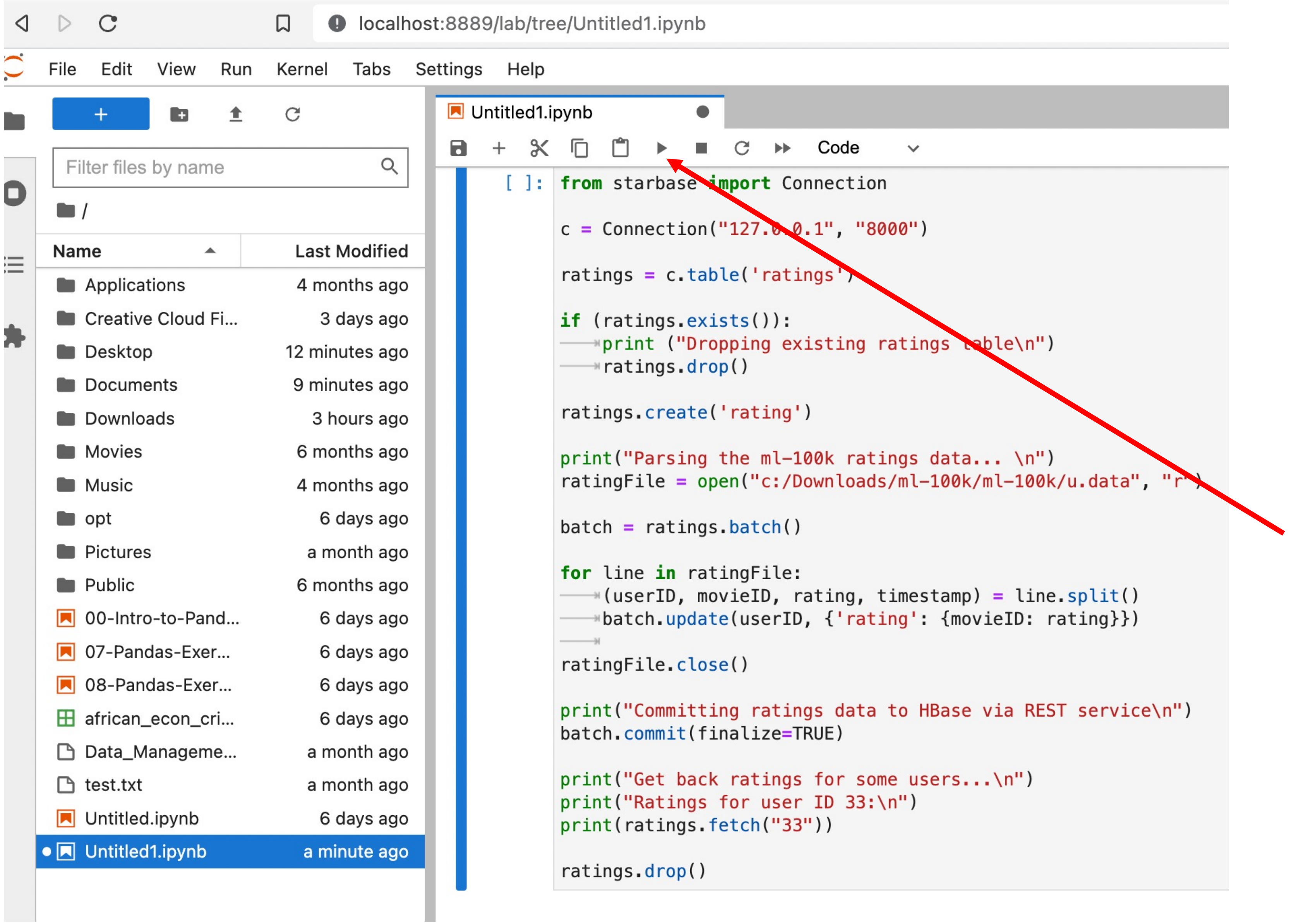
Start coding

```
pip install starbase

from starbase import Connection

c = Connection("127.0.0.1", "8000")

ratings = c.table('ratings')

if (ratings.exists()):
        print ("Dropping existing ratings table\n")
        ratings.drop()

ratings.create('rating')

print("Parsing the ml-100k ratings data... \n")
ratingFile = open(r"c:/Downloads/ml-100k/ml-100k/u.data", "r")

batch = ratings.batch()
```

Change directory accordingly

```
for line in ratingFile:
        (userID, movieID, rating, timestamp) = line.split()
        batch.update(userID, {'rating': {movieID: rating}})

ratingFile.close()

print("Committing ratings data to HBase via REST service\n")
batch.commit(finalize=~~TRUE~~)
                      True

print("Get back ratings for some users...\n")
print("Ratings for user ID 33:\n")
print(ratings.fetch("33"))

ratings.drop()
```

Using JupyterLab

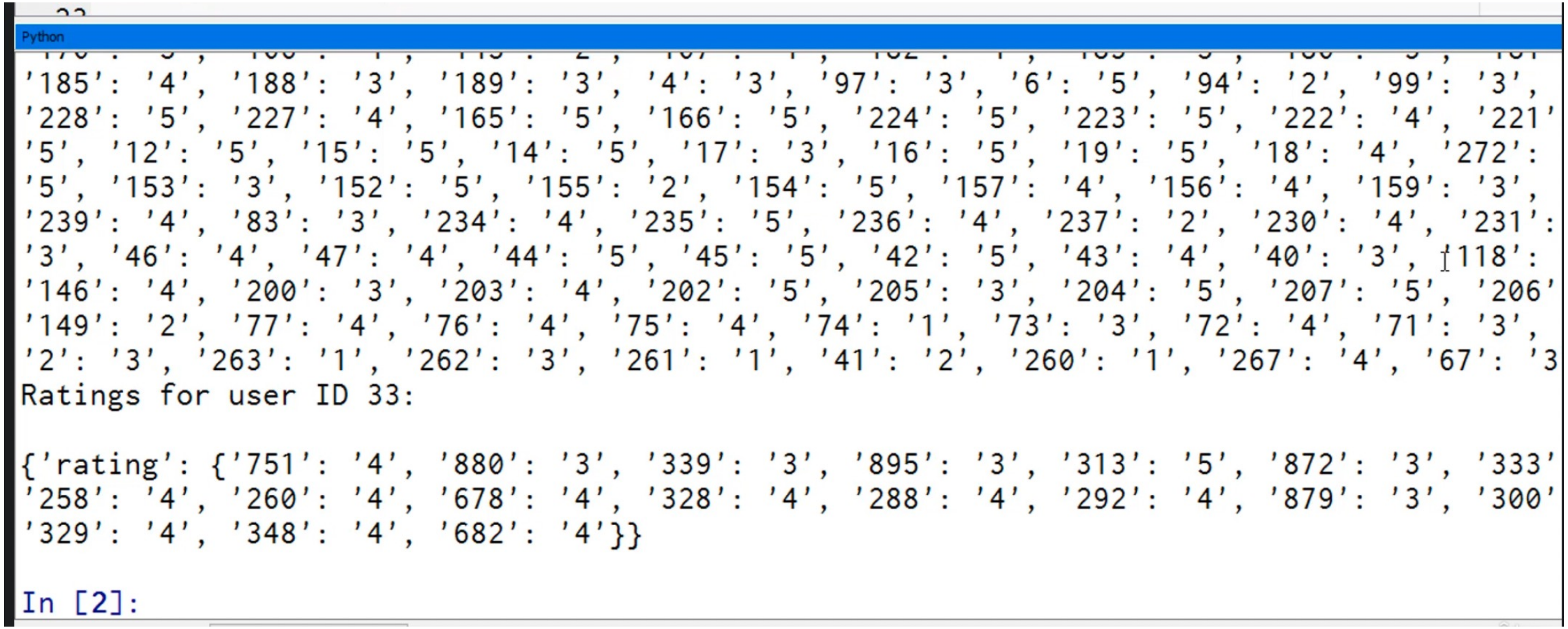


Name	Last Modified
Applications	4 months ago
Creative Cloud Fi...	3 days ago
Desktop	12 minutes ago
Documents	9 minutes ago
Downloads	3 hours ago
Movies	6 months ago
Music	4 months ago
opt	6 days ago
Pictures	a month ago
Public	6 months ago
00-Intro-to-Pand...	6 days ago
07-Pandas-Exer...	6 days ago
08-Pandas-Exer...	6 days ago
african_econ_cri...	6 days ago
Data_Manageme...	a month ago
test.txt	a month ago
Untitled.ipynb	6 days ago
Untitled1.ipynb	a minute ago

```
[ ]: from starbase import Connection

     c = Connection("127.0.0.1", "8000")

     ratings = c.table('ratings')

     if (ratings.exists()):
         print ("Dropping existing ratings table\n")
         ratings.drop()

     ratings.create('rating')

     print("Parsing the ml-100k ratings data... \n")
     ratingFile = open("c:/Downloads/ml-100k/ml-100k/u.data", "r")

     batch = ratings.batch()

     for line in ratingFile:
         (userID, movieID, rating, timestamp) = line.split()
         batch.update(userID, {'rating': {movieID: rating}})

     ratingFile.close()

     print("Committing ratings data to HBase via REST service\n")
     batch.commit(finalize=TRUE)

     print("Get back ratings for some users...\n")
     print("Ratings for user ID 33:\n")
     print(ratings.fetch("33"))

     ratings.drop()
```

Click the icon to start running the python script

Output



```
'185': '4', '188': '3', '189': '3', '4': '3', '97': '3', '6': '5', '94': '2', '99': '3',
'228': '5', '227': '4', '165': '5', '166': '5', '224': '5', '223': '5', '222': '4', '221'
'5', '12': '5', '15': '5', '14': '5', '17': '3', '16': '5', '19': '5', '18': '4', '272':
'5', '153': '3', '152': '5', '155': '2', '154': '5', '157': '4', '156': '4', '159': '3',
'239': '4', '83': '3', '234': '4', '235': '5', '236': '4', '237': '2', '230': '4', '231':
'3', '46': '4', '47': '4', '44': '5', '45': '5', '42': '5', '43': '4', '40': '3', '118':
'146': '4', '200': '3', '203': '4', '202': '5', '205': '3', '204': '5', '207': '5', '206'
'149': '2', '77': '4', '76': '4', '75': '4', '74': '1', '73': '3', '72': '4', '71': '3',
'2': '3', '263': '1', '262': '3', '261': '1', '41': '2', '260': '1', '267': '4', '67': '3
Ratings for user ID 33:

{'rating': {'751': '4', '880': '3', '339': '3', '895': '3', '313': '5', '872': '3', '333'
'258': '4', '260': '4', '678': '4', '328': '4', '288': '4', '292': '4', '879': '3', '300'
'329': '4', '348': '4', '682': '4'}}

In [2]:
```

Ratings for user ID 33

To stop REST interface:
/usr/hdp/current/hbase-master/bin/hbase-daemon.sh stop rest