Tried on Ubuntu 16.04

STEP 1 : Installation

- Open up terminal
- `mkdir tools && cd tools`
- `wget http://apache.claz.org/hbase/stable/hbase-1.2.6-bin.tar.gz`
- `tar -xvzf hbase-1.2.6-bin.tar.gz`
- `cd hbase-1.2.6/`

For HBase 0.98.5 and later, we are required to set the JAVA_HOME environment variable before starting Hbase using conf/hbase-env.sh.

The JAVA_HOME variable should be set to a directory which contains the executable file bin/java.

- `sudo gedit ~/.profile`
- Add these two lines at the end
  export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64
  export PATH=$PATH:$JAVA_HOME/bin
- `source ~/.profile`
- `echo $JAVA_HOME` to check if the path is set correctly

Now starting the shell
The ./bin/start-hbase.sh script is provided as a convenient way to start HBase.

- `./bin/start-hbase.sh`
  starting master, logging to
  /home/rshah9/tools/hbase-1.2.6/bin/../logs/hbase-rshah9-master-bn16-246.dcs.mcnc.org.out
  OpenJDK 64-Bit Server VM warning: ignoring option PermSize=128m; support was removed in 8.0
  OpenJDK 64-Bit Server VM warning: ignoring option MaxPermSize=128m; support was removed in 8.0

- `./bin/hbase shell`
  thbase(main):001:0>

Now the shell has started.

- Inside the shell type `version` to check version or type `help`. You can exit using `exit` command.

STEP 2: Basic Queries

- Connect to your running instance of HBase using the hbase shell command.
- Use the create command to create a new table. You must specify the table name and the Column-Family name.

  hbase> create 'test', 'cf'
  0 row(s) in 1.2200 seconds

- Use the list command to see the List Information About your Table.
  hbase> list 'test'
  TABLE
  test
  1 row(s) in 0.0350 seconds
  => ["test"]
- To put data into your table, use the put command.
  hbase> put 'test', 'row1', 'cf:a', 'value1'
  0 row(s) in 0.1770 seconds

- Use the scan command to scan the table for data.
  hbase> scan 'test'
  ROW COLUMN+CELL
  row1 column=cf:a, timestamp=1403759475114, value=value1
  1 row(s) in 0.0440 seconds

- To get a single row of data at a time, use the get command.
  hbase> get 'test','row1'
  COLUMN CELL
  cf:a timestamp=1403759475114, value=value1
  1 row(s) in 0.0230 seconds

- If you want to delete a table or change its settings, you need to disable the table first, using the disable command. You can re-enable it using the enable command.
  hbase> disable 'test‘
  0 row(s) in 1.6270 seconds

- hbase> enable 'test'
  0 row(s) in 0.4500 seconds

- To drop (delete) a table, use the drop command.
  hbase> drop 'test'
  0 row(s) in 0.2900 seconds

- To exit the HBase Shell type `exit`

- use `./bin/stop-hbase.sh` script to stop hbase service
  $ ./bin/stop-hbase.sh
  stopping hbase..... $

STEP 3: HBase Thrift

- `cd ..` (Now you are in tools directory)
- `sudo apt install python-dev`
- `sudo apt update && sudo apt upgrade && sudo apt autoremove` (to update all packages, will take some time)
- `sudo apt-get install libboost-dev libboost-test-dev libboost-program-options-dev libboost-filesystem-dev libboost-thread-dev libevent-dev automake libtool flex bison pkg-config g++ libssl-dev`
- `wget https://archive.apache.org/dist/thrift/0.6.0/thrift-0.6.0.tar.gz`
- `tar xfz thrift-0.6.0.tar.gz`
- `cd thrift-0.6.0/`
- `./configure`
- `sudo make` (Will take some time)
- `sudo make install`
- `thrift` to see if it is working

- `cd ../`
- Download Hbase.thrift file from here https://drive.google.com/file/d/0B5dejdhAYHztOGt6OGs5ZTZ3WEk/view?usp=sharing and put it in current folder (`pwd` for current directory)
- `thrift -gen py Hbase.thrift` (This will create a folder named gen-py)
- Start the hbase service if you stopped it in step 1 (inside tools/hbase-1.2.6/ do `./bin/start-hbase.sh`)
- `./hbase-1.2.6/bin/hbase thrift start`
- Open new terminal
- `cd tools/gen-py`
- `gedit table.py` and paste below code (excluding the first line and quotes at end)
  table.py (See explanation below)
  ```python
  from thrift.transport.TSocket import TSocket
  from thrift.transport.TTransport import TBufferedTransport
  from thrift.protocol import TBinaryProtocol
  from hbase import Hbase

  transport = TBufferedTransport(TSocket('127.0.0.1', 9090))
  transport.open()
  protocol = TBinaryProtocol.TBinaryProtocol(transport)
  client = Hbase.Client(protocol)
  print(client.getTableNames())
  ```

```

- `sudo apt install python-pip`
- `sudo pip install thrift`
- `python table.py` should give output of the tables that you have created in the hbase database. ['test']

Explanation of the code that we just wrote :

The below commands will import all the required HBase Thrift modules.

```
from thrift.transport.TSocket import TSocket
from thrift.transport.TTransport import TBufferedTransport
from thrift.protocol import TBinaryProtocol
from hbase import Hbase
```

The below command will create the socket transport and line protocol and allows the Thrift client to connect and talk to the Thrift server.

```
transport = TBufferedTransport(TSocket('localhost', 9090))
```

Next we need to open the socket to the Thrift server.

```
transport.open()
```

Tbinary is binary implementation of thrift (converting transport to binary implementation)

```
protocol = TBinaryProtocol.TBinaryProtocol(transport)
```

The below lines create the Client object which will be used to interact with HBase. From this client object, you will issue all your Gets and Puts.

```
client = Hbase.Client(protocol)
print(client.getTableNames())
```

STEP 4: HBase Complex Application with APIs

We will be creating a todo application using Hbase and Python.

- `cd ~/tools`
- `git clone https://github.com/rikenshah/flask-hbase-todos.git`
- `cd flask-hbase-todos`
- `sudo pip install happybase flask`
```

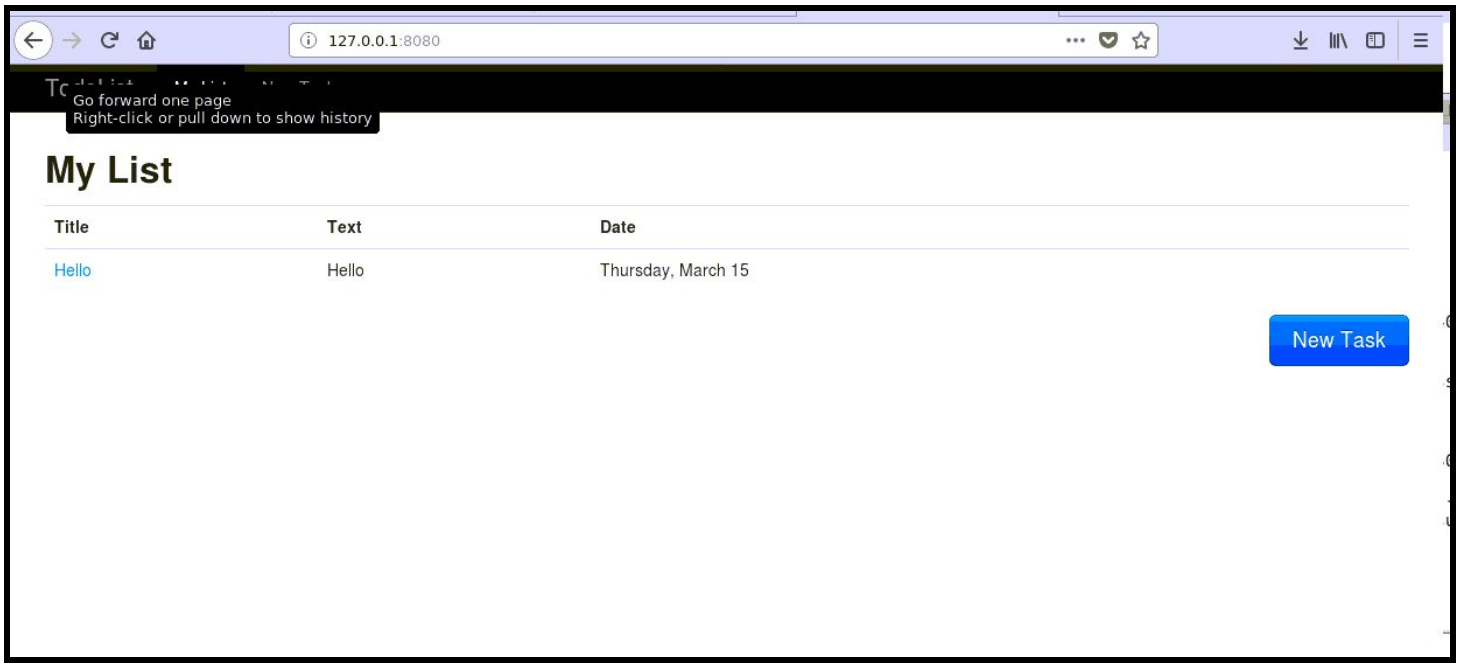
- In new terminal open hbase shell as described in STEP 1
- In the shell do `create 'todos', 'todos'`
- In another shell start thrift server like in Step 3. (`./tools/hbase-1.2.6/bin/hbase thrift start`)
- In current terminal do `python app.py`
- In browser visit `http://127.0.0.1:8080/`.
- Explore the code and play around
- Inside of third terminal, do check how entries are made from shell (Use `scan` command as described in Step 2)

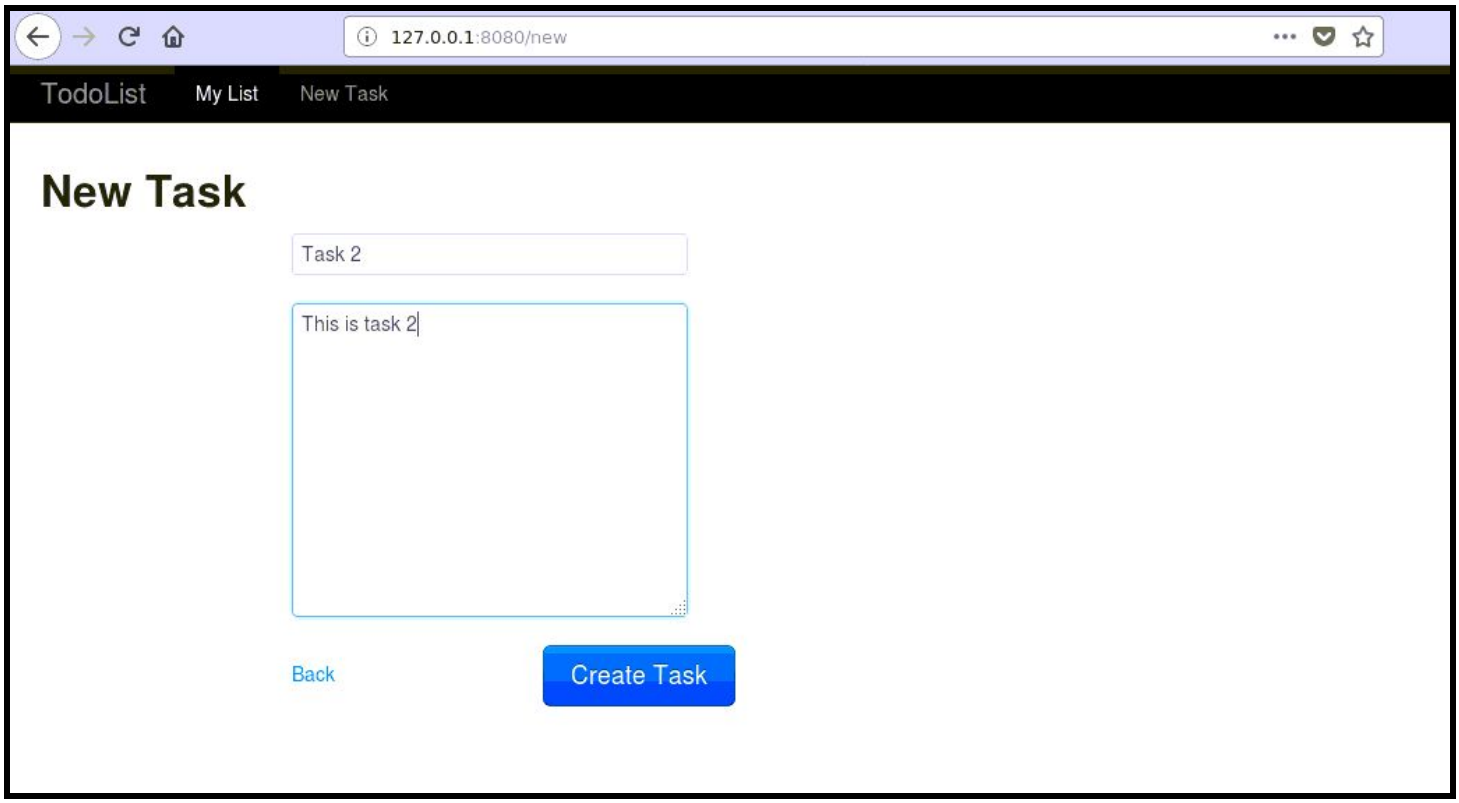
The main logic resides in `todoapp.py` file. It has following code.

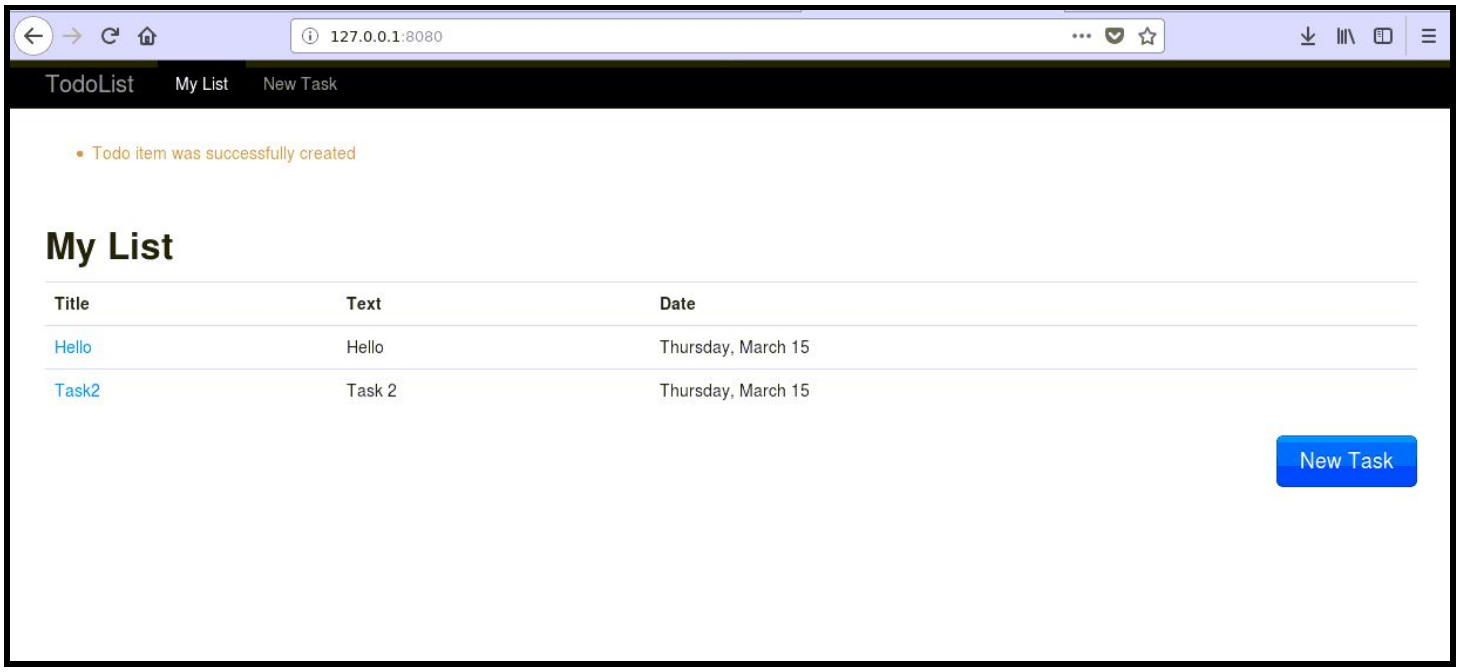
```python
from datetime import datetime
import happybase
import json
import time
import calendar
from flask import Flask, request, flash, url_for, redirect, \
     render_template, abort

app = Flask(__name__)
app.config.from_pyfile('todoapp.cfg')
time_format = "%Y-%m-%d %H:%M:%S"
time_format_w_ms = "%Y-%m-%d %H:%M:%S.%f"

connection = happybase.Connection(app.config['HBASE_HOST'], app.config['HBASE_PORT'])
column_and_key = app.config['HBASE_TABLE']+":"+app.config['HBASE_TABLE']

@app.route('/')
def index():
   return render_template('index.html', todos=getTasks())

@app.route('/new', methods=['GET', 'POST'])
def new():
   if request.method == 'POST':
      if not request.form['title']:
         flash('you must provide a basic description', 'error')
      elif not request.form['text']:
         flash('additional notes are required', 'error')
      else:
         saveTask({"title": request.form['title'], "text":request.form['text'], "pub_date": str(datetime.now()),
"done": "False" })
         flash(u'Todo item was successfully created')
         return redirect(url_for('index'))
```

```
    return render_template('new.html')

@app.route('/todos/<int:id>', methods = ['GET' , 'POST'])
def show_or_update(id):
    item = getTask(id)
    if request.method == 'GET':
        return render_template('view.html',todo=item)
    item['title'] = request.form['title']
    item['pub_date'] = item['pub_date'].strftime(time_format_w_ms)
    item['text']  = request.form['text']
    item['done']  = str(('done.%d' % id) in request.form)
    if request.method == 'POST':
        saveTask(item)
    return redirect(url_for('index'))

def getTasks():
    hbase = connection.table(app.config['HBASE_TABLE'])
    print "get all"
    results = []
    for key, data in hbase.scan():
        print key, data
        results.append(_decode(data))
    return results

def getTask(id):
    print "get "+str(id)
    #timestmp = _unix_to_datetime(id)
    hbase = connection.table(app.config['HBASE_TABLE'])
    #return _decode(hbase.row(timestmp))
    return _decode(hbase.row(str(id)))

def saveTask(task):
    item = _encode(task)
    pub_date = datetime.strptime(task['pub_date'], time_format)
    id = str(_datetime_to_unix(pub_date))
    hbase = connection.table(app.config['HBASE_TABLE'])
    hbase.put(id, {column_and_key: str(item)})

def delTask(id):
    hbase = connection.table(app.config['HBASE_TABLE'])
    table.delete(str(id))
    print "deleting: "+str(id)
```

```
def resetTable():
    dropTable()
    createTable()

def dropTable():
    print "dropping our table..."
    if( app.config['HBASE_TABLE'] in connection.tables()):
        connection.delete_table( app.config['HBASE_TABLE'], True)

def createTable():
    a = {app.config['HBASE_TABLE']: dict()}
    print a
    if( app.config['HBASE_TABLE'] not in connection.tables()):
        print "creating a table schema..."
        connection.create_table( app.config['HBASE_TABLE'], a)

def _encode(item):
    pub_date = datetime.strptime(item['pub_date'], time_format_w_ms)
    item['pub_date'] = pub_date.strftime(time_format)
    item['done'] = str(item['done'])
    return json.dumps(item)

def _decode(item):
    task = json.loads(item[column_and_key])
    pub_date = datetime.strptime(task['pub_date'], time_format)
    done = True if task['done'] == 'True' else False
    id = _datetime_to_unix(pub_date)
    return {"title": str(task['title']), "text": str(task['text']), "pub_date": pub_date, "done": done, "id": id }

def _unix_to_datetime(id):
    return datetime.fromtimestamp(int(id)+25200).strftime(time_format)

def _datetime_to_unix(date):
    return calendar.timegm(date.timetuple())


if __name__ == '__main__':
    print connection.tables()
    #createTable()
    resetTable()
    app.run()

```
```
```


127.0.0.1:8080
Go forward one page
Right-click or pull down to show history
My List
Title
Text
Date
Hello
Hello
Thursday, March 15
New Task


127.0.0.1:8080/new
TodoList
My List
New Task
New Task
Task 2
This is task 2
Back
Create Task


127.0.0.1:8080
TodoList
My List
New Task
Todo item was successfully created
My List
Title
Text
Date
Hello
Hello
Thursday, March 15
Task2
Task 2
Thursday, March 15
New Task

THANK YOU...!