

COMP7940 Cloud Computing

2023/24 S2 Lab 4 - Consuming external service (Redis/ChatGPT)

Role	Name	Email
Qichen Wang	Instructor	qcwang@hkbu.edu.hk
Yu Xu	Teaching Assistant	csyuxu@comp.hkbu.edu.hk
Hongduan Tian	Teaching Assistant	cshdtian@comp.hkbu.edu.hk

Intended Learning Outcomes

Throughout this lab you will be able to:

1. Experience in consuming an external database service;
2. Improve your chatbot to respond based on a database;
3. Equip your chatbot with ChatGPT 🤖 ;

1. Redis (Remote Dictionary Server) database

Redis is an open-source in-memory storage, used as a distributed, in-memory **key-value** database, cache and message broker, with optional durability ([Ref: wikipedia](#))

1.1 Register a Redis account

[Redis website](#)

1.2 Add a new cloud database (free subscription)

- After you add a new subscription, you could see the webpage like the follows:

The screenshot shows the Redis Cloud dashboard. On the left, there's a sidebar with options like Subscriptions, Databases, Data Access Control, Access Management, Logs, Account Settings, Usage Report, Billing & Payments, Download Center, Support, What's New?, and Documentation. The main area displays a 'Subscriptions' card for a 'Fixed' subscription named 'xu free subscription' (ID: #2248445) in 'asia-northeast1'. It shows 1 database, 2.3MB / 30MB (8%). To the right, there's a 'Let's get started' section with links for creating a subscription, database, and connecting. A 'Databases' table lists the single database with details: Status (green), Name (xu-free-db), Endpoint (Connect button), Memory (2.3MB), Plan Usage (7.8%), Capabilities (Redis Stack), and Options (three dots). Buttons for 'Upgrade plan' and 'Page 1 of 1' are also present.

1.3 Access your cloud database using Python

- Click the button of `Connect` and select the client type as `Python`; Then you could see the webpage like the follows:

The screenshot shows the 'Connect to xu-free-db' page. At the top, it says 'Let's get started' and 'Create a subscription'. Below that, it displays a 'xu free subscription' (Creation date: 05-Feb-2024 | ID: #2248445). The 'Databases' table is identical to the one in the previous screenshot. To the right, there's a 'Connect to xu-free-db' section. It shows 'RedisInsight' (Recommended) and 'Redis CLI' as options. Under 'Redis Client', it says 'Select your client' and has a dropdown set to 'Python'. Below that, it says 'Copy the following code snippet to your application' and provides a code block for Python:

```
import redis
r = redis.Redis(
    host='redis-19878.c302.asia-northeast1-1.gce.cloud.redislabs.com',
    port=19878,
    password='*****')
```

A 'Copy' button is available next to the code block, and a link to 'Install Redis Client' is at the bottom.

- Click the button of `Copy` and paste in your python environment using terminals to connect your Redis cloud database:

```
(lab3) xuyu@xuyudeMacBook-Air ~ % python
Python 3.12.1 | packaged by Anaconda, Inc. | (main, Jan 19 2024, 09:45:58) [Clang 14.0
Type "help", "copyright", "credits" or "license" for more information.
>>> import redis
>>>
>>> r = redis.Redis(
...     host='redis-19878.c302.asia-northeast1-1.gce.cloud.redislabs.com',
...     port=19878,
...     password='KLB4jPzIEB5kmXwKLwwUTAyeSX5qKVoi')
>>>
```

- Check whether you successfully connected the Redis cloud database:

```
>>> r.ping()
True
```

- Add new **key-value** pairs into your cloud database using the method of `set(key, value)` :

```
>>> r.set("my_name", "xuyu")
True
```

- Get the value of the given key from your cloud database using the method of `get(key)` :

```
>>> r.get("my_name")
b'xuyu'
```

- Check whether a particular key existing in the database or not using the method of `exists(key)` :

```
>>> r.exists("my_name")
1
>>> r.exists("random123")
0
```

- Delete a key in the database using the method of `delete(key)`

```
>>> r.delete("my_name")
1
>>> r.exists("my_name")
0
```

2. Employing your chatbot integrating Redis cloud services

2.1 Update your configuration file created in Lab3

```
[TELEGRAM]  
ACCESS_TOKEN = 6735106346:AAFHGTTtRYDzw7YMxq7dJ0wwS-i8Ids4SKGM  
[REDIS]  
HOST = edis-19878.c302.asia-northeast1-1.gce.cloud.redislabs.com  
PASSWORD = KLB4jPzIEB5kmXwKLwwUTAyeSX5qKvoi  
REDISPORT = 19878
```

2.2 Update `chatbot.py` of the echo chatbot to integrate Redis

```
from telegram import Update
from telegram.ext import (Updater, CommandHandler, MessageHandler, Filters,
                          CallbackContext)
import configparser
import logging
import redis

global redis1
def main():
    # Load your token and create an Updater for your Bot
    config = configparser.ConfigParser()
    config.read('config.ini')
    updater = Updater(token=(config['TELEGRAM']['ACCESS_TOKEN']), use_context=True)
    dispatcher = updater.dispatcher
    global redis1
    redis1 = redis.Redis(host=(config['REDIS']['HOST']),
                         password=(config['REDIS']['PASSWORD']),
                         port=(config['REDIS']['REDISPORT']))

    # You can set this logging module, so you will know when
    # and why things do not work as expected Meanwhile, update your config.ini as:
    logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                        level=logging.INFO)

    # register a dispatcher to handle message: here we register an echo dispatcher
    echo_handler = MessageHandler(Filters.text & (~Filters.command), echo)
    dispatcher.add_handler(echo_handler)

    # on different commands - answer in Telegram
    dispatcher.add_handler(CommandHandler("add", add))
    dispatcher.add_handler(CommandHandler("help", help_command))

    # To start the bot:
    updater.start_polling()
    updater.idle()

def echo(update, context):
    reply_message = update.message.text.upper()
    logging.info("Update: " + str(update))
    logging.info("context: " + str(context))
    context.bot.send_message(chat_id=update.effective_chat.id, text= reply_message)
```

```

# Define a few command handlers. These usually take the two arguments update and
# context. Error handlers also receive the raised TelegramError object in error.

def help_command(update: Update, context: CallbackContext) -> None:
    """Send a message when the command /help is issued."""
    update.message.reply_text('Helping you helping you.')

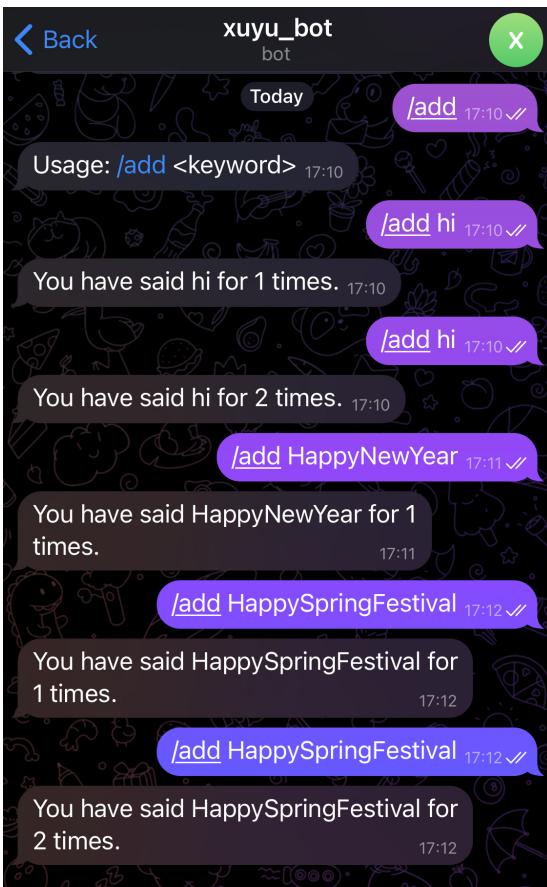

def add(update: Update, context: CallbackContext) -> None:
    """Send a message when the command /add is issued."""
    try:
        global redis1
        logging.info(context.args[0])
        msg = context.args[0] # /add keyword <-- this should store the keyword
        redis1.incr(msg)

        update.message.reply_text('You have said ' + msg + ' for ' +
                                 redis1.get(msg).decode('UTF-8') + ' times.')
    except (IndexError, ValueError):
        update.message.reply_text('Usage: /add <keyword>')

if __name__ == '__main__':
    main()

```

- We add two commands to the echo chatbot (created in the Lab3): `/help` and `/add`. The command of `/help` could be designed to provide information for users to use your chatbot. We implemented the command of `/add` as a simple example which counts the frequency of specific input words.



(Optional) Try to use `set()`, `get()`, or `delete()` into the `chatbot.py` to enable your bot replying certain answers given particular messages.

3. ChatGPT: integrating large language models (LLM) into your chatbot

Breakthrough: The top-tier scientific journal *Nature* featured ChatGPT as one of "Ten people (and one non-human) who helped shape science in 2023".

3.1 Get the access token of HKBU's ChatGPT

- HKBU's ChatGPT webpage: <https://chatgpt.hkbu.edu.hk/>
- Select `API` in the upper-right corner of the webpage

Signed in as
[redacted]@life.hkbu.edu.hk

FAQ

API

Terms & Conditions

Privacy Policy

Feedback

Contact Us

Sign out

Copyright © 2023, Hong Kong Baptist University. All Rights Reserved.

- Generate a access token

HKBU ChatGPT API Service

This page contains information about the REST API endpoint for the HKBU ChatGPT API Service. To make use of the service, you will require your key and endpoint. **The key is a unique identifier and provides access to your account, please DO NOT share it with others.** The API usage will be monitored, and the API will share the same monthly quota as the HKBU ChatGPT Service.

You can consult the following supported versions of swagger specification for more guidance.

[2023-08-01-preview](#)

[2023-12-01-preview](#)

To obtain your own key, click on the **Generate** button below. Keep in mind that the new key will only be displayed once, and the previous key will become invalid if it exists.

Key:



Generate

Close



Type your query



3.2 Update your configuration file

[TELEGRAM]

ACCESS_TOKEN = [6735106346:AAFHGTTtRYDzw7YMxq7dJ0wwS-i8Ids4SKGM](#)

[REDIS]

HOST = edis-19878.c302.asia-northeast1-1.gce.cloud.redislabs.com

PASSWORD = KLB4jPzIEB5kmXwKLwwUTAyeSX5qKVoi

REDISPORT = 19878

[CHATGPT]

BASICURL = <https://chatgpt.hkbu.edu.hk/general/rest>

MODELNAME = gpt-35-turbo-16k

APIVERSION = [2023-12-01-preview](#)

ACCESS_TOKEN = YOUR_ACCESS_TOKEN

3.3 Access ChatGPT using Python

- You may need to install requests first if you have not installed before.

```
pip install requests
```

- Now we create another Python file `ChatGPT_HKBU.py` for accessing ChatGPT

```

import configparser
import requests

class HKBU_ChatGPT():
    def __init__(self, config_= './config.ini'):
        if type(config_) == str:
            self.config = configparser.ConfigParser()
            self.config.read(config_)
        elif type(config_) == configparser.ConfigParser:
            self.config = config_

    def submit(self, message):
        conversation = [{"role": "user", "content": message}]

        url = (self.config['CHATGPT']['BASICURL']) +
        "/deployments/" + (self.config['CHATGPT']['MODELNAME']) +
        "/chat/completions/?api-version=" +
        (self.config['CHATGPT']['APIVERSION'])

        headers = { 'Content-Type': 'application/json',
        'api-key': (self.config['CHATGPT']['ACCESS_TOKEN']) }
        payload = { 'messages': conversation }
        response = requests.post(url, json=payload, headers=headers)
        if response.status_code == 200:
            data = response.json()
            return data['choices'][0]['message']['content']
        else:
            return 'Error:', response

if __name__ == '__main__':
    ChatGPT_test = HKBU_ChatGPT()

    while True:
        user_input = input("Typing anything to ChatGPT:\t")
        response = ChatGPT_test.submit(user_input)
        print(response)

```

- Run the script of `ChatGPT_HKBU.py` and ask some things in the terminal:

```
% python ChatGPT_HKBU.py
Typing anything to ChatGPT:      hello
Hello! How can I assist you today?
Typing anything to ChatGPT:
```

3.4 Update chatbot.py

Add the below code:

```
from ChatGPT_HKBU import HKBU_ChatGPT

def equiped_chatgpt(update, context):
    global chatgpt
    reply_message = chatgpt.submit(update.message.text)
    logging.info("Update: " + str(update))
    logging.info("context: " + str(context))
    context.bot.send_message(chat_id=update.effective_chat.id, text=reply_message)
```

Add `dispatcher` for chatgpt and disable the part of echo function in `main()`:

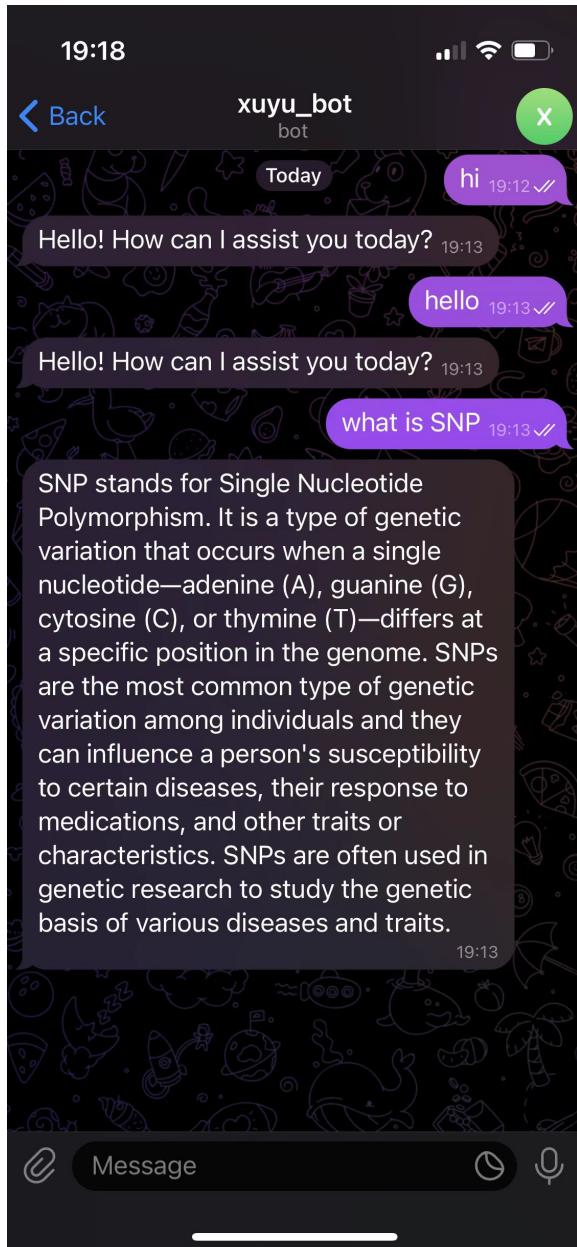
```
# register a dispatcher to handle message: here we register an echo dispatcher
# echo_handler = MessageHandler(Filters.text & (~Filters.command), echo)
# dispatcher.add_handler(echo_handler)

# dispatcher for chatgpt
global chatgpt
chatgpt = HKBU_ChatGPT(config)
chatgpt_handler = MessageHandler(Filters.text & (~Filters.command),
                                 equiped_chatgpt)
dispatcher.add_handler(chatgpt_handler)
```

3.5 Telegram chatbot equipped with ChatGPT

```
python chatbot.py
```

An example screenshot:



Push your code to Github

This is the end of Lab4. Please push your code to Github.

Writeup

Answer the following questions and submit them to moodle:

1. Please describe the architecture of the current chatbot system. Identify the components and check where they are running now.
2. Explain how your chatbot handles the special command. You need to trace the code and explain that.
3. Update your code so that when user types /hello Kevin, it will reply Good day, Kevin!. Write down the changes you have made.

4. Make a few screen caps to prove that you have applied your own Redis account, used it in your chatbot, and push the code on GitHub (at least 2 commits - lab3/lab4).