COMP7940 Cloud Computing

2022/23 S2 Lab 4 Consuming external service

Instructor	Dr. Qichen Wang	qcwang@hkbu.edu.hk
Teaching Assistant	Mr. Zhengheng Tang	zhtang@comp.hkbu.edu.hk
Teaching Assistant	Mr. Zhen Ye	cszhenye@comp.hkbu.edu.hk

Objective:

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;

Continue with your chatbot in lab3. Update the code with the following.

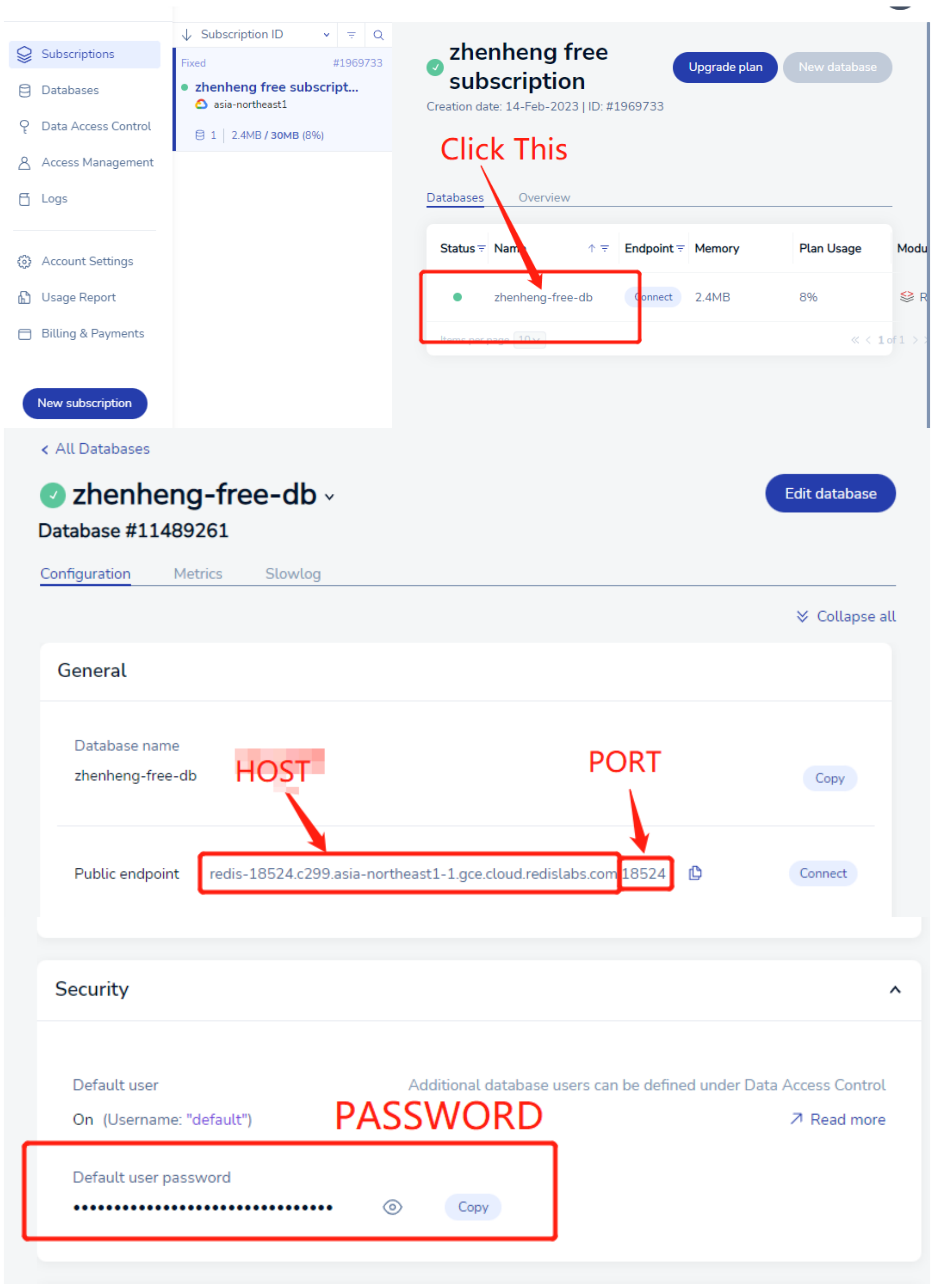
```
from telegram import Update
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters,
CallbackContext

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
```

```
    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()
```

Updating the Redis Server

The redis account currently is owned by Kevin. You should get your own redis account from RedisLab. Then replace the server setting with your redis service. After this step, the counter of all keywords should be reset.

1. Apply A RedisLab FREE account: https://app.redislabs.com/#/, and create a cloud database for yourself.
2. Following this page https://docs.redis.com/latest/rc/security/database-security/passwords-users-roles/ to find the account and password information of your database.



3. Meanwhile, update your `config.ini` as: (The HOST, PASSWORD, REDISPORT should be changed as yours)

```
[TELEGRAM]
ACCESS_TOKEN = YOUR_TOKEN_HERE

[REDIS]
HOST = redis-18524.c299.asia-northeast1-1.gce.cloud.redislabs.com
PASSWORD = ***********************************
REDISPORT = 18524
```

You can try running the program again. This program has added two new command `/help` and `/add`.

Type `/help` in telegram. You should see a different reply.

Also, try to type `/add abc` for a few times. And try to type `/add hello` or other keywords and observe what is happening.

Redis

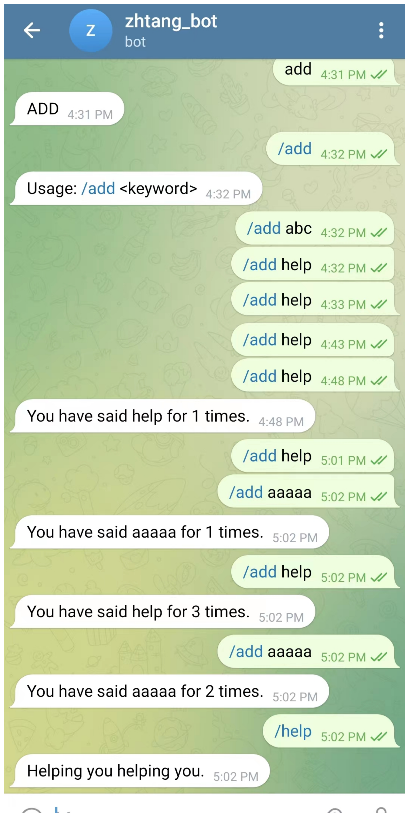
We use an external database service redis here. Redis is a NoSQL database. The database currently containing many keys-value pairs, like:

Keys	Value
haha	1
hi	5
abc	5
def	3
...	

The following command establish a connection to the redis server:

```
redis1 = redis.Redis(host=(config['REDIS']['HOST']), password=(config['REDIS']
['PWD']), port=(config['REDIS']['PORT']))
```

The command `redis1.incr(msg)` increases the count of the key and `redis1.get(msg).decode('UTF-8')` returns the string value of the count. By using the redis server, we can save and load data very easily.



add 4:31 PM

ADD 4:31 PM

/add 4:32 PM

Usage: /add <keyword> 4:32 PM

/add abc 4:32 PM

/add help 4:32 PM

/add help 4:33 PM

/add help 4:43 PM

/add help 4:48 PM

You have said help for 1 times. 4:48 PM

/add help 5:01 PM

/add aaaaa 5:02 PM

You have said aaaaa for 1 times. 5:02 PM

/add help 5:02 PM

You have said help for 3 times. 5:02 PM

/add aaaaa 5:02 PM

You have said aaaaa for 2 times. 5:02 PM

/help 5:02 PM

Helping you helping you. 5:02 PM

Push your code to GitHub

This is the end of Lab 4, please push your code to your 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 are they running now.
2. Explain how do your chatbot handle the special command. You need to trace the code and explain that.
3. Update your code so that when user type `/hello Kevin`, it will reply `Good day, Kevin!`. Write down the change 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).