24450804 Li Ching Lam

COMP 7940 Cloud Computing

Lab 4

Questions:

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).

Answer:

1. The chatbot is running on a local machine. It consists of a database, which is running on Redis, which provides storage on cloud. It also contains external resources which is through calling API to receive response from ChatGPT. ChatGPT is running on cloud partform.
2. The chat could handle special commands like /help, /add and /hello. It can also respond to prompt answered by ChatGPT.

```
dispatcher.add_handler(chatgpt_handler)
dispatcher.add_handler(CommandHandler("add", add))
dispatcher.add_handler(CommandHandler("help", help_command))
dispatcher.add_handler(CommandHandler("hello", hello_command))
```

For /help, when user type /help, it responds 'Helping you helping you.'

```
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.')
```

For /add, when user type /add <keyword>, it records the keyword and increase the count by 1. Then response 'You have said' + <keyword> + 'for' + <count>+ ' times.'

```
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 k
        redis1.incr(msg)
        update.message.reply_text('You have said ' + msg + ' for ' +
            redis1.get(msg) + ' times.')
    except (IndexError, ValueError):
        update.message.reply_text('Usage: /add <keyword>')
```

```
"""Send a message when the command /add is issued."""
try:
    global
    logging
    msg = c                                    keyword <-- this should store the keyword
    redis1.incr(msg)
```

```
(method) def incr(
    name: _Key,
    amount: int = 1
) -> int
```

For /hello, when user type /hello <keyword>, it responds 'Good day, ' + <keyword> + '!'

```
#add /hello command
def hello_command(update: Update, context: CallbackContext) -> None:
    """Send a message when the command /hello is issued."""
    try:
        logging.info(context.args[0])
        msg = context.args[0] # /hello keyword <-- this should store the k
        update.message.reply_text('Good day, ' + msg +'!')
    except (IndexError, ValueError):
        update.message.reply_text('Usage: /add <keyword>')
```

For general query, it is handled by ChatGPT through API, prompt is posted to ChatGPT and receive response and output to user.

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

```python
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)
```

3. In main(), after help handler add the following

   dispatcher.add_handler(CommandHandler("hello", hello_command))

   And after add function, add the following function

   #add /hello command

   def hello_command(update: Update, context: CallbackContext) -> None:

   """Send a message when the command /hello is issued."""

   try:

   logging.info(context.args[0])

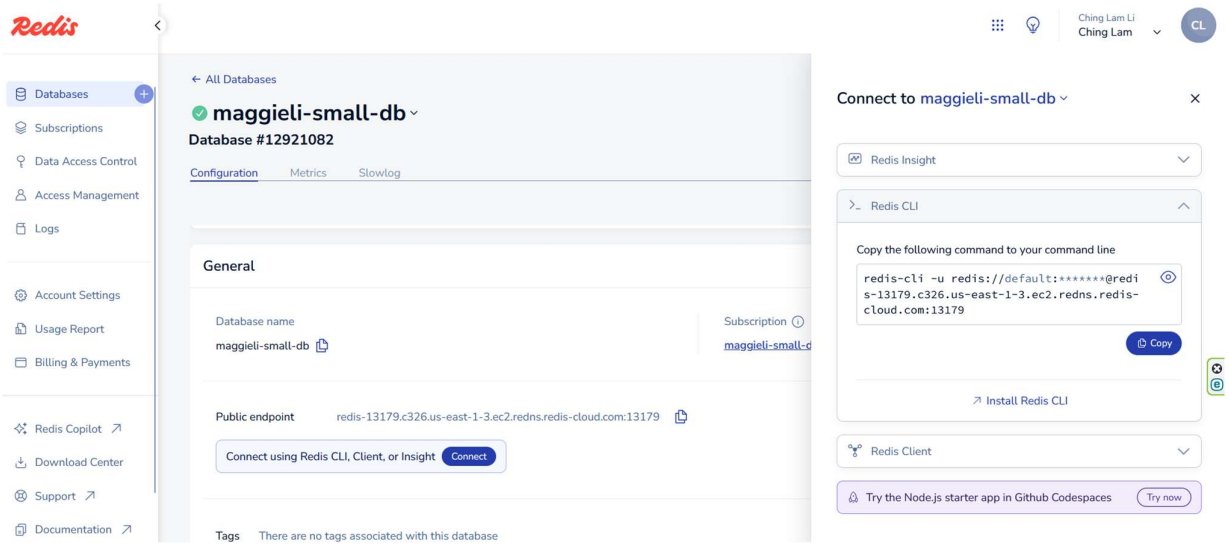
   msg = context.args[0] # /hello keyword <-- this should store the keyword

   update.message.reply_text('Good day, ' + msg +'!')

   except (IndexError, ValueError):

   update.message.reply_text('Usage: /add <keyword>')

4. Created account in Redis



Store credentials in config.ini

```
[REDIS]
HOST = redis-13179.c326.us-east-1-3.ec2.redns.redis-cloud.com
PASSWORD =
REDISPORT = 13179
DECODE_RESPONSE = true
USER_NAME = default
```
in

Connect to redis when starting the chatbot

```
global redis1
redis1 = redis.Redis(host=(config['REDIS']['HOST']),
    password=(config['REDIS']['PASSWORD']),
    port=(config['REDIS']['REDISPORT']),
    decode_responses=(config['REDIS']['DECODE_RESPONSE']),
    username=(config['REDIS']['USER_NAME']))
```

Committed to Github