Cloud Computing Mini Project Report

Slackbot for Heroku

Chirag Dixit (D17B/20), Prathamesh Dongre(D17B/22), Sagar Dung(D17B/24)

Abstract

Chatbots are a popular way to allow real people to perform complex tasks in an easily understandable manner. The aim of this project is to build a chatbot for Slack (a messaging platform) that lets a team control their Heroku (a cloud platform service) account by modifying their deployed instances. This is useful for teams that already have a Slack group and want to provide internal cloud access in a clear and transparent manner

Introduction

Slack is a cloud-based team collaboration tool founded by Stewart Butterfield. Slack began as an internal tool used by their company, Tiny Speck, in the development of Glitch, a now defunct online game. The name is an acronym for "Searchable Log of All Conversation and Knowledge".Slack teams allow communities, groups, or teams to join through a specific URL or invitation sent by a team admin or owner. Although Slack was meant for organizational communication, it has been slowly turning into a community platform, a function for which users had previously used message boards or social media such as Facebook or LinkedIn groups. Many of these communities are categorized by topics which a group of people may be interested in discussing.

Heroku is a cloud Platform-as-a-Service (PaaS) supporting several programming languages that is used as a web application deployment model. Heroku, one of the first cloud platforms, has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it lets the developer build, run and scale applications in a similar manner across all the languages. Heroku was acquired by Salesforce.com in 2010. Applications that are run from the Heroku server use the Heroku DNS Server to direct to the application

domain (typically "applicationname.herokuapp.com"). Each of the application containers, or dynos, are spread across a "dyno grid" which consists of several servers. Heroku's Git server handles application repository pushes from permitted users.

A **chatbot** is a computer program which conducts a conversation via auditory or textual methods. Such programs are often designed to convincingly simulate how a human would behave as a conversational partner. Chatbots are typically used in dialog systems for various practical purposes including customer service or information acquisition. Some chatterbots use sophisticated natural language processing systems, but many simpler systems scan for keywords within the input, then pull a reply with the most matching keywords, or the most similar wording pattern, from a database.

Implementation Details

A bot is a simple server-side program that receives a user's message (from the messaging platform), parses it to determine the user's intent, performs the task and sends a reply back the chat application's server.

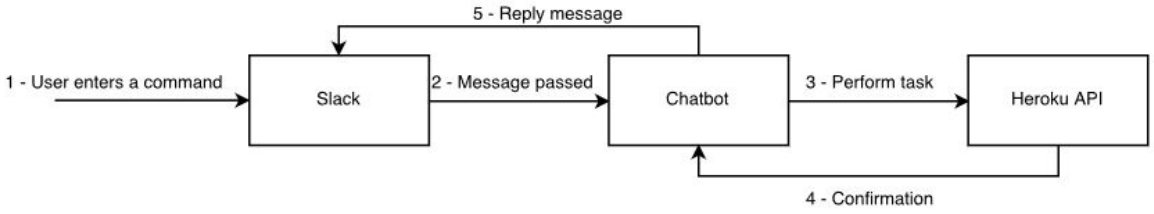
Slack allows developers to deploy bots on channels that they have access to. This involves registering the bot's URL with Slack, which will send an HTTP request whenever an event takes place on the channel. Heroku allows developers to manage their cloud instances through an API. Both of these abilities have been leveraged to make the user experience as simple as possible.

Through the Slack settings for an organization, a bot can be added to a particular channel and given granular access to events that occur in the channel such as messages posted, files uploaded etc.

Heroku applications are bundled into packages called slugs, which contain all the dependencies needed to run the application. This slug is given a unique id, and can be modified, copied and

deleted as required. For example, an application can be forked by simply copying it's slug and redeploying it in another instance.

The bot has been implemented using Python, and deployed on a Heroku instance.



System Architecture

herokubot.py

```
import time
from slackclient import SlackClient
import heroku3
import requests

SLACK_BOT_TOKEN = 'XXXXXXXX'
SLACK_BOT_ID = 'XXXXXX'
SLACK_BOT_TAG = "<@" + SLACK_BOT_ID + ">"
HEROKU_API_KEY = 'XXXX-XXXXX-XXXXX-XXXX-XXXXXX'

slack_client = SlackClient(SLACK_BOT_TOKEN)
heroku_conn = heroku3.from_key(HEROKU_API_KEY)

def create_app(app_name):
  try:
    heroku_conn.create_app(app_name)
    return True
  except TypeError as te:
    return False
```

```
def rename_app(old_app_name, new_app_name):
    try:
        heroku_conn.app(old_app_name).rename(new_app_name)
        return True
    except requests.exceptions.HTTPError as he:
        return False

def delete_app(app_name):
    try:
        heroku_conn.app(app_name).delete()
        return True
    except requests.exceptions.HTTPError as he:
        return False

def app_exists(app_name):
    apps = heroku_conn.apps()
    for app in apps:
        if app.name == app_name:
            return True
    return False

def fork_app(original_app_name, new_app_name):
    response = ''
    if app_exists(original_app_name) == False:
        response = original_app_name + ' does not exist'
        return False, response
    if app_exists(new_app_name) == False:
        response = new_app_name + ' does not exist. Creating...\n'
        create_success = create_app(new_app_name)
        if create_success:
            response += (new_app_name + ' created successfully\n')
        else:
            response += (new_app_name + ' could not be created')
            return False, response
    slug_id = get_latest_slug_id(original_app_name)
    create_new_release(slug_id, new_app_name)
    response += (original_app_name + ' has been forked successfully')
    return True, response
```

```
def get_latest_slug_id(app_name):
  url = 'https://api.heroku.com/apps/' + app_name + '/releases'
  headers = {'Content-Type': 'application/json',
             'Accept': 'application/vnd.heroku+json; version=3',
             'Authorization': 'Bearer ' + HEROKU_API_KEY}
  response = requests.get(url=url, headers=headers)
  return response.json()[-1]['slug']['id']

def create_new_release(slug_id, new_app_name):
  url = 'https://api.heroku.com/apps/' + new_app_name + '/releases'
  headers = {'Content-Type': 'application/json',
             'Accept': 'application/vnd.heroku+json; version=3',
             'Authorization': 'Bearer ' + HEROKU_API_KEY}
  data = '{"slug": "' + slug_id + '"}'
  requests.post(url=url, headers=headers, data=data)

def get_logs(app_name, lines):
  if app_exists(app_name) == False:
    response = app_name + ' does not exist'
    return False, response
  log_url = create_log_url(app_name, lines)
  return True, requests.get(log_url).text

def create_log_url(app_name, lines):
  url = 'https://api.heroku.com/apps/' + app_name + '/log-sessions'
  headers = {'Content-Type': 'application/json',
             'Accept': 'application/vnd.heroku+json; version=3',
             'Authorization': 'Bearer ' + HEROKU_API_KEY}
  data = '{"lines": "' + lines + '"}'
  r = requests.post(url=url, headers=headers, data=data)
  return r.json()["logplex_url"]

def handle_command(command, channel):
  """
    Receives commands directed at the bot and determines if they
    are valid commands. If so, then acts on the commands.
  """

  print('Received request : ' + command)
```

```
response = ''
# parse command arguments
try:

    if command.startswith("create app"):
        app_name = command.split(" ")[2]
        success = create_app(app_name)
        if success:
            response = app_name + ' has been created successfully'
        else:
            response = app_name + ' could not be created as the name has been taken already'

    elif command.startswith("rename app"):
        old_app_name = command.split(" ")[2]
        new_app_name = command.split(" ")[4]
        success = rename_app(old_app_name, new_app_name)
        if success:
            response = old_app_name + ' has been renamed successfully'
        else:
            response = old_app_name + ' could not be renamed as it does not exist, or the name
' + new_app_name + \
                    ' has already been taken'

    elif command.startswith("delete app"):
        app_name = command.split(" ")[2]
        success = delete_app(app_name)
        if success:
            response = app_name + ' has been deleted successfully'
        else:
            response = app_name + ' does not exist'

    elif command.startswith("list apps"):
        apps = heroku_conn.apps()
        response = ''
        for app in apps:
            response = response + "ID : " + app.id + " Name : " + app.name + "\n"

    elif command.startswith("fork app"):
        original_app_name = command.split(" ")[2]
```

```
            new_app_name = command.split(" ")[4]
            success, response = fork_app(original_app_name, new_app_name)

        elif command.startswith("get logs for"):
            app_name = command.split(" ")[3]
            lines = command.split(" ")[4][6:]
            success, response = get_logs(app_name, lines)

        else:
            response = "Invalid command."
            print("An invalid command was given : " + command)

    except Exception as e:
        print("An exception was thrown : " + str(e))
        response = 'An error occurred.'

    slack_client.api_call("chat.postMessage", channel=channel,
                          text=response, as_user=True)


def parse_slack_output(slack_rtm_output):
    """
        The Slack Real Time Messaging API is an events firehose.
        this parsing function returns None unless a message is
        directed at the Bot, based on its ID.
    """
    output_list = slack_rtm_output
    if output_list and len(output_list) > 0:
        for output in output_list:
            if output and output['type'] == 'message' and 'text' in output:
                message_text = output['text']
                if message_text != '' and message_text.startswith(SLACK_BOT_TAG):
                    return message_text.split(SLACK_BOT_TAG)[1].strip().lower(), output['channel']
    return None, None


if __name__ == "__main__":
    READ_WEBSOCKET_DELAY = 1
    if slack_client.rtm_connect():
```

```
    print("HerokuBot has started successfully.")
    while True:
        command, channel = parse_slack_output(slack_client.rtm_read())
        if command and channel:
            handle_command(command, channel)
        time.sleep(READ_WEBSOCKET_DELAY)
else:
    print("Connection failed. Invalid Slack token or bot ID ?")
```

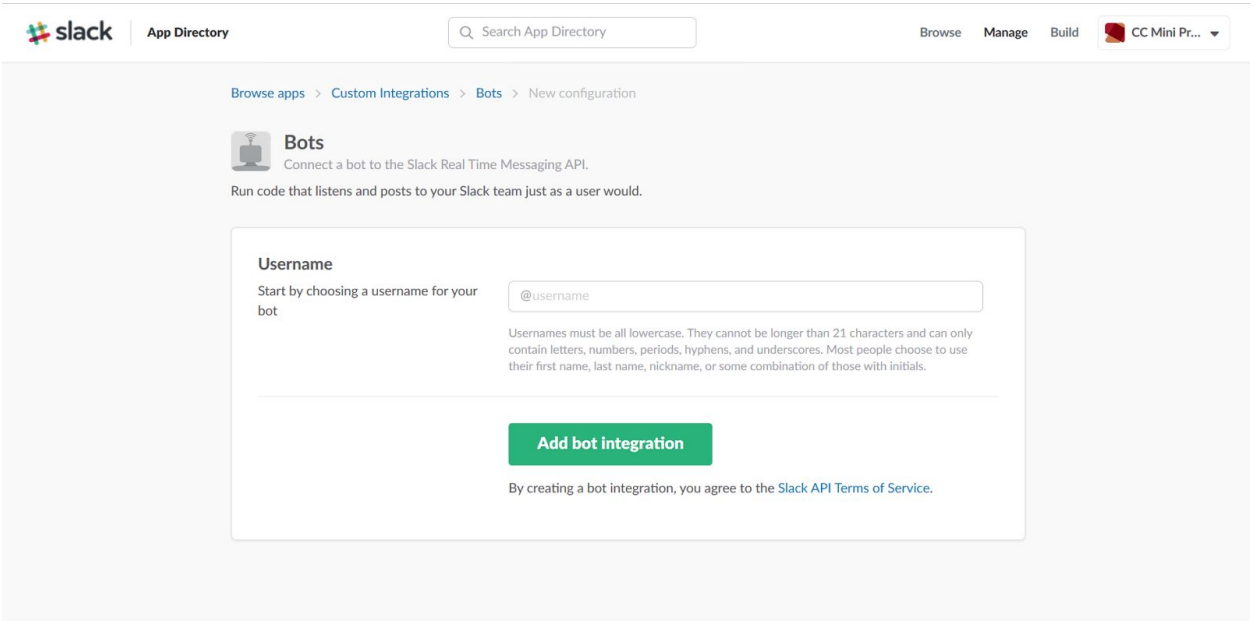
Result

The Heroku cloud platform has been studied. It's API has been used to create a chatbot for Slack that allows the manipulation of the instances under an account. The bot has the following functionalities :

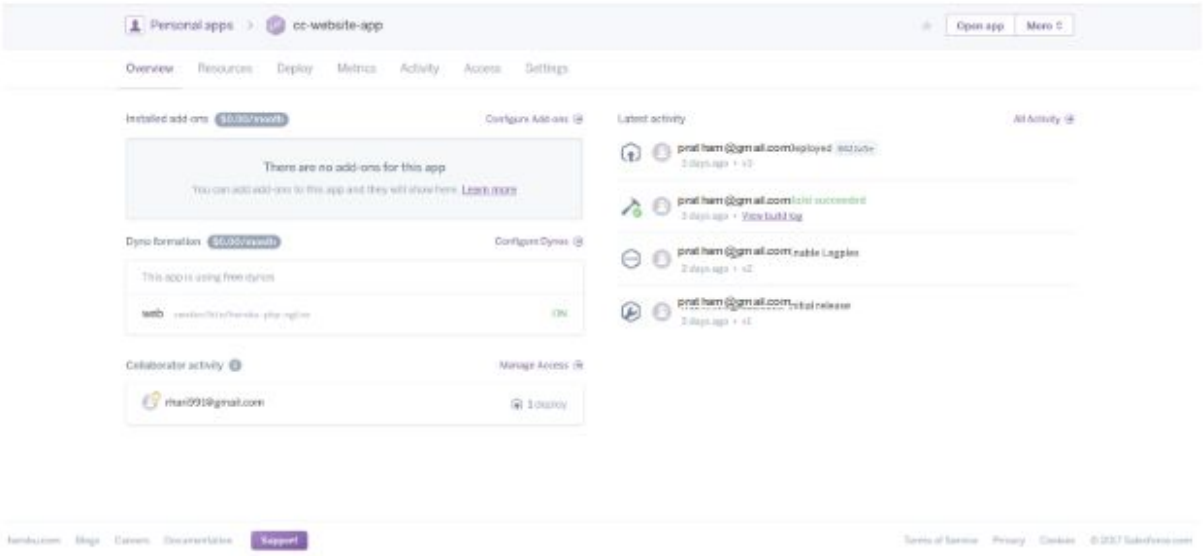
- Creating a new instance
- Renaming an existing instance
- Deleting an existing instance
- Listing all running instances
- Forking an existing instance
- Obtaining the logs of a live instance

A sample single page web application written in PHP has been deployed to a Heroku instance. The above functionalities have been tested against this instance.

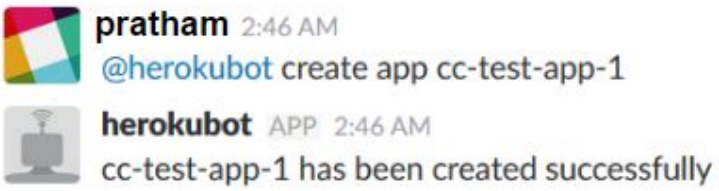
Snapshots



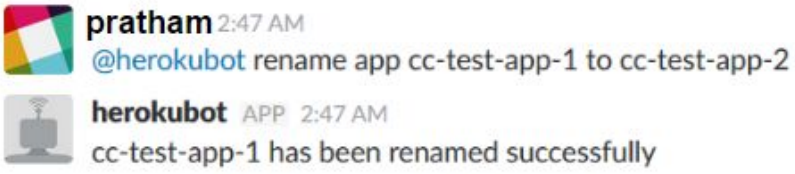
Adding a bot user to a Slack channel



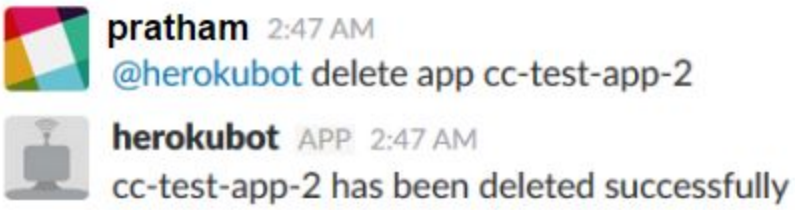
The Heroku dashboard for a deployed instance



Creating an app

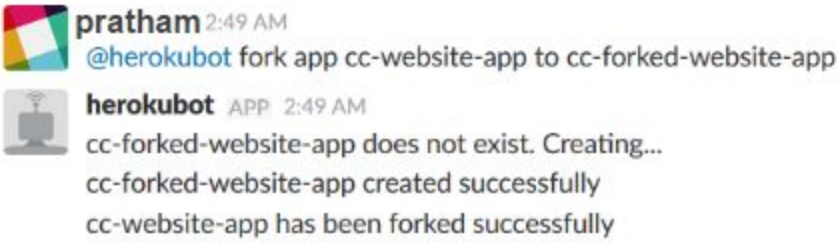


Renaming an app

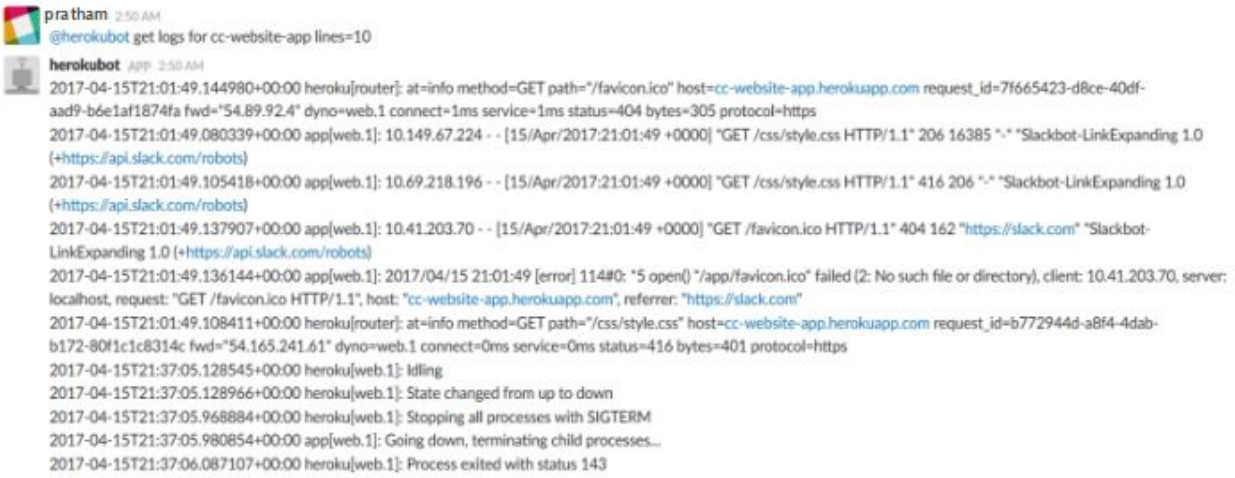


Deleting an app

Listing all live apps



Forking a live app



Fetching logs

Conclusion

The Heroku cloud platform has been studied. It uses slugs to package code in reusable containers and allows users to modify their working using dynos and add-on packs. The Heroku API has been used to create an interactive chatbot for Slack.

References

1) https://devcenter.heroku.com/articles/platform-api-reference
2) https://api.slack.com/bot-users
3) https://en.wikipedia.org/wiki/Heroku
4) https://en.wikipedia.org/wiki/Slack_(software)