

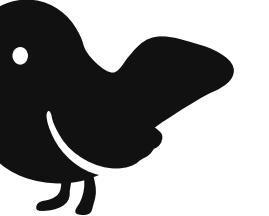
Automate the Boring Stuff with Slackbot

Takanori Suzuki

PyCon / 2020 Apr 17

Agenda

- Background and Motivation for Slackbot
- How to create Simple bot
- How to create Interactive bot
- How to Extend bot using libs and APIs

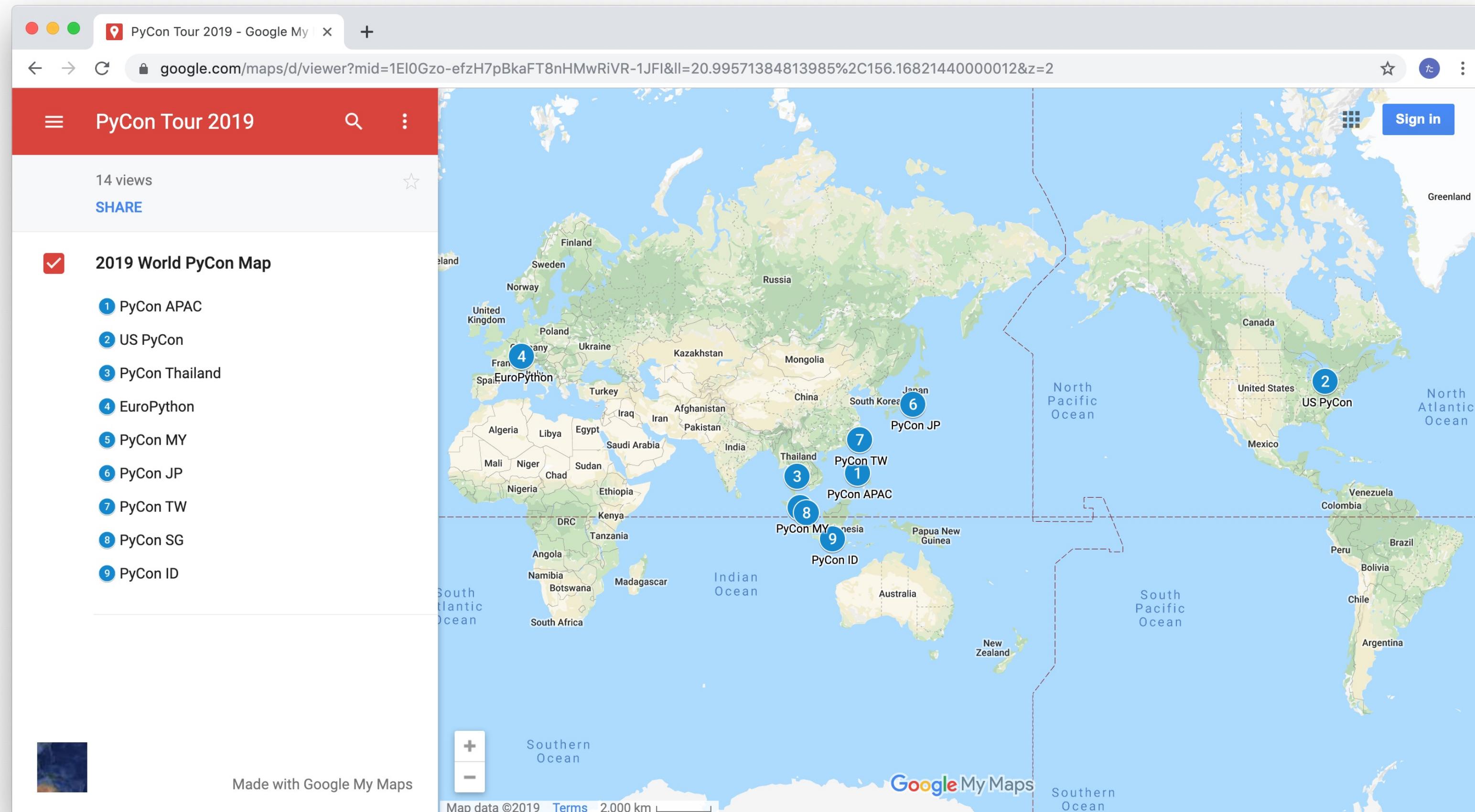
Tweets  

#pycon2020 / @takanory

Who am I?

- Takanori Suzuki / 鈴木 たかのり
-  [@takanory](https://twitter.com/takanory)
- PyCon JP Association
- BeProud Inc.
- Python Boot Camp, Python mini Hack-a-thon,
Python Bouldering Club





www.google.com/maps/d/viewer

@takanory

Background and Motivation

Conference Tasks

- I held PyCon JP(2014-2016) as **Chair**
- Conference tasks:
 -  Keynotes, Talks and Trainings arrangement
 -  Ticket sales and reception
 -  Venue and facility(WiFi, Video...) management
 -  Foods,  Coffee,  Snacks and  Beers

Staff ask me the same things

- 40+ staff
-  NEW staff :  OLD staff = 50 : 50

Programmer is Lazy

Let's create a secretary!!

Goal

- How to create **Simple** bot
- How to create **Interactive** bot
- How to **Extend** bot using libs and APIs

Why Slackbot

- Slack running all the time on  
- Easy to access Slack
- To do **Everything** in Slack

#pycon-overseas ★
5 | 世界のPyCon情報 <https://tinyurl.com/r54v7w4>

takanory 00:13
20分からじゃねーのかよ!!! (edited)

terapyon 00:13
画像の話？

canzawa 00:13
ですねw

canzawa 00:18
取り敢えず啓蒙?しました。
<https://twitter.com/yutakanzawa/status/1393224062713798656>

Yuta Kanzawa @yutakanzawa
Keynote of #PyConUS now! #PyCon
<https://pbs.twimg.com/media/E1W5uszVUAEP589.jpg>
Twitter | May 15th (86 kB) ▾

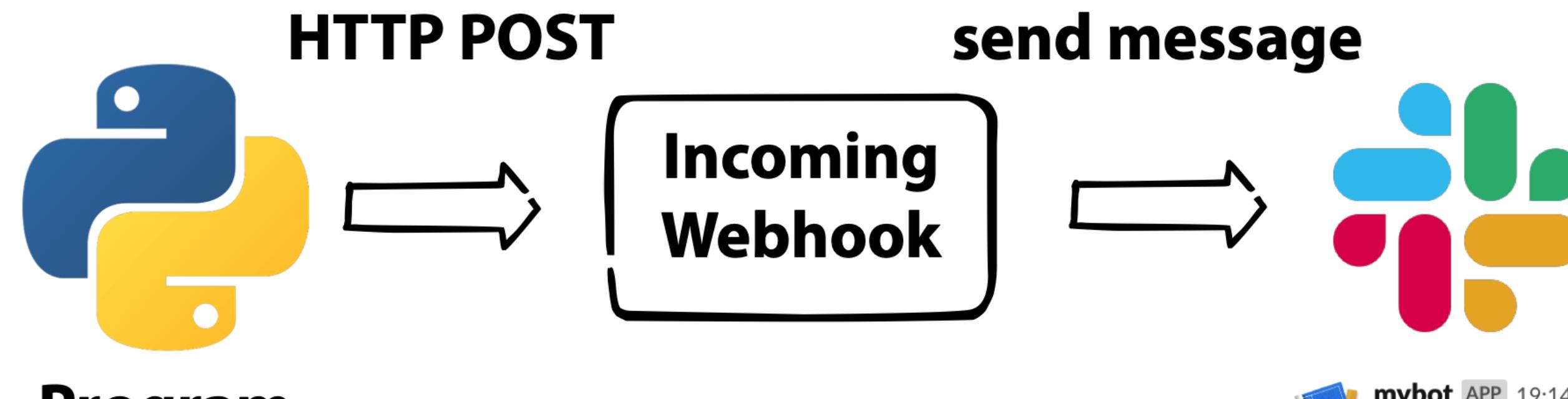
Saturday, May 15th ▾

106   

@takanory

Simple integration with Incoming Webhooks

System overview



Create Incoming Webhooks Integration

- Generate Webhook URL
 - 1. Create a Slack app
 - 2. Enable Incoming Webhooks in the app
 - 3. Create an Incoming Webhook
- Webhook URL:

<https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXX>

- see: [Incoming Webhooks](#)

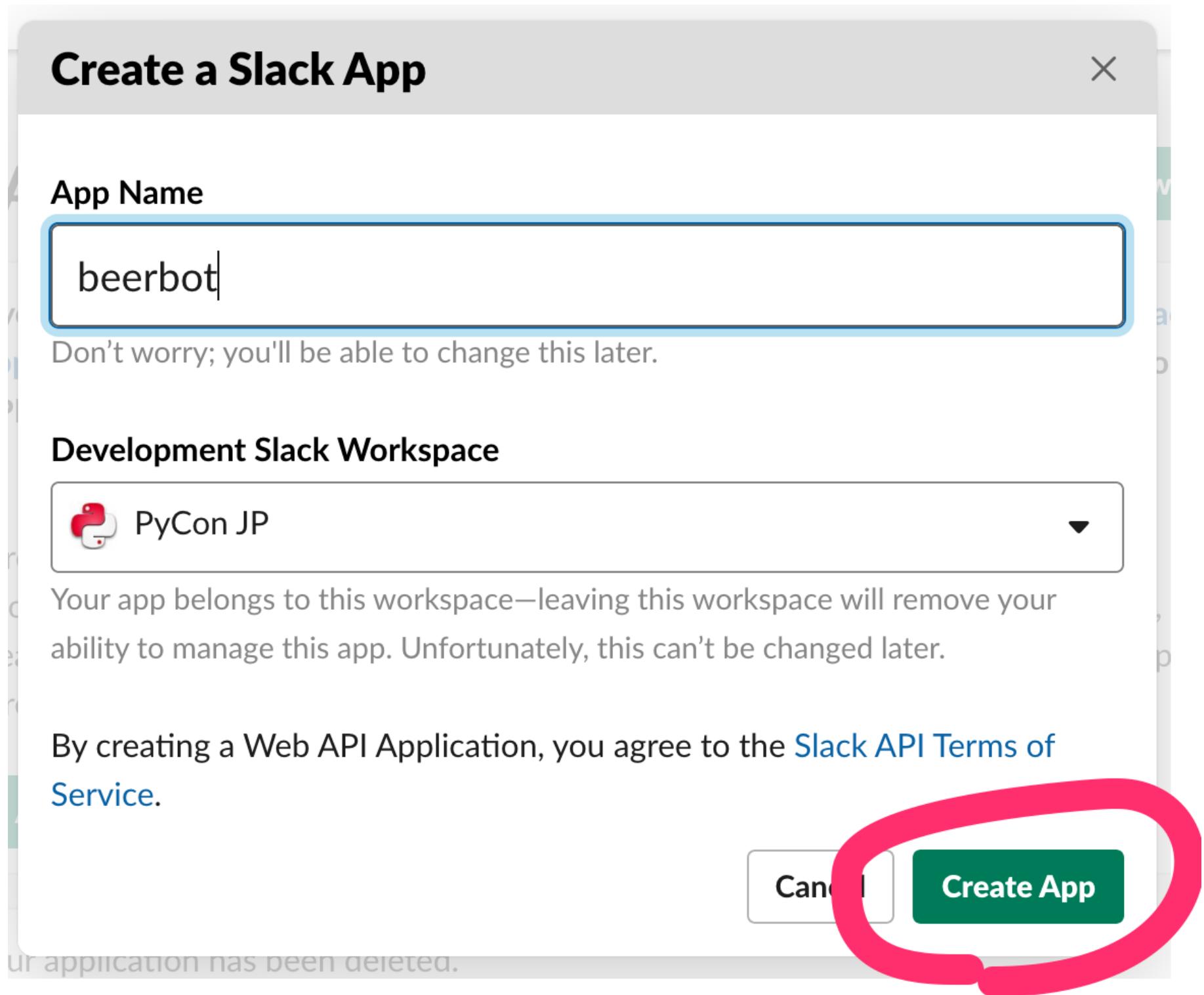
Your Apps

[Create New App](#)

-  If your app is (or will be) listed in the Slack App Directory, please review our [Slack App Directory Agreement](#). These terms are in addition to the existing Developer Policy, API TOS, and Brand Guidelines.

By keeping your app in the App Directory or review process, you're confirming your agreement to the Slack App Directory Agreement and to providing additional information for security review, if requested. If you don't agree with this Agreement, please send an email to feedback@slack.com, and we'll remove your app from the App Directory or the review process.

[I Agree](#)



Display Information

This information will be shown in the Slack App Directory and in the Slack App For more information, view our [App Detail Guidelines](#).

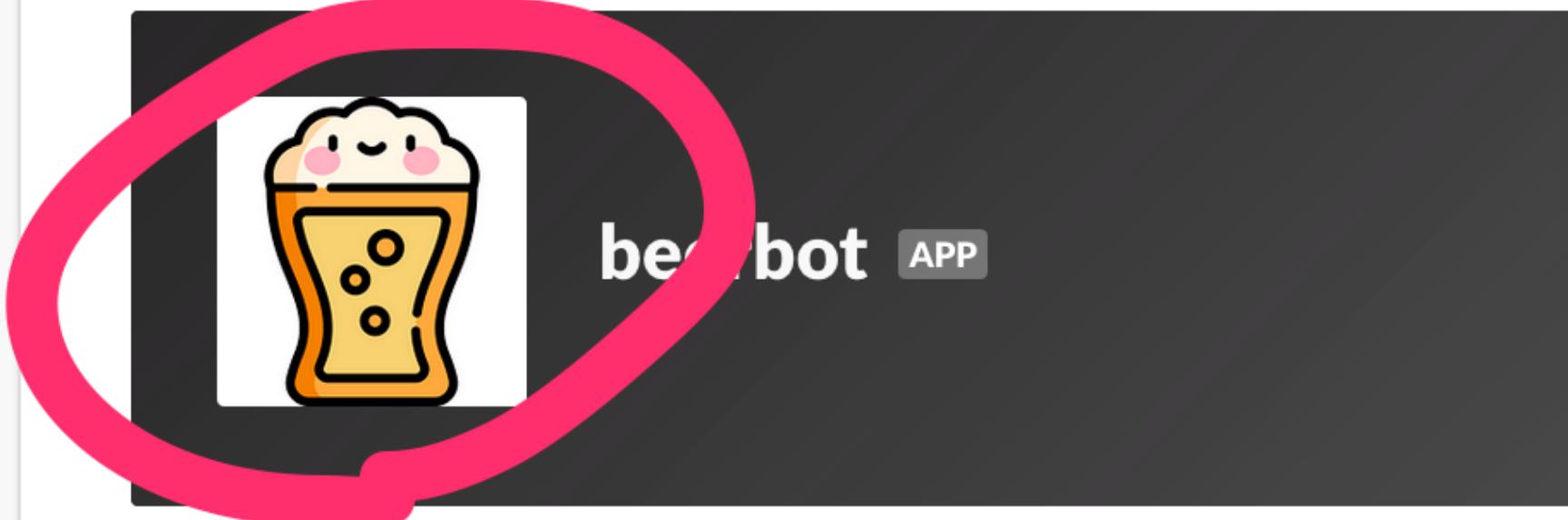
App name

beerbot

Short description

Add a short description...

App icon & Preview



Background color



#2C2D30

- credit: Icons made by [Freepik](#) from [www.flaticon.com](#)

Basic Information

Building Apps for Slack

Create an app that's just for your workspace (or build one that can be used by any workspace) by following the steps below.

Add features and functionality

Choose and configure the tools you'll need to create your app (or review all [our documentation](#)).

Incoming Webhooks

Post messages from external sources into Slack.

Interactive Components

Add buttons to your app's messages, and create an interactive experience for users.

Slash Commands

Allow users to perform app actions by typing commands in Slack.

Event Subscriptions

Make it easy for your app to respond to activity in Slack.

Bots

Add a bot to allow users to exchange messages with your app.

Permissions

Configure permissions to allow your app to interact with the Slack API.

Incoming Webhooks

Activate Incoming Webhooks

On

Incoming webhooks are a simple way to post messages from external sources into Slack. They make use of normal HTTP requests with a JSON payload, which includes the message and a few other optional details. You can include [message attachments](#) to display richly-formatted messages.

Each time your app is installed, a new Webhook URL will be generated.

If you deactivate incoming webhooks, new Webhook URLs will not be generated when your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to [Revoke All OAuth Tokens](#).

Webhook URLs for Your Workspace

To dispatch messages with your webhook URL, send your [message](#) in JSON as the body of an [application/json](#) POST request.

Add this webhook to your workspace below to activate this curl example.

Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}' YOUR_WEBHOOK_URL_HERE
```

Webhook URL	Channel	Added By
-------------	---------	----------

No webhook URLs have been added yet.

[Add New Webhook to Workspace](#)

This app was created by a member of your workspace, PyCon JP.



On PyCon JP, beerbot would like to:

Confirm your identity on PyCon JP

Post to #random ▾

Cancel

Allow



your app is installed to your team. If you'd like to remove access to existing Webhook URLs, you will need to [Revoke All OAuth Tokens](#).

Webhook URLs for Your Workspace

To dispatch messages with your webhook URL, send your [message](#) in JSON as the body of an [application/json](#) POST request.

Add this webhook to your workspace below to activate this curl example.

Sample curl request to post to a channel:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello,  
World!"}' https://hooks.slack.com/services/T024G2ZE8/BNAFX3QGN
```

[Copy](#)

Webhook URL	Channel	Added By
https://hooks.slack.com/services/T024G2ZE8/BNAFX3QGN	#pyconjpbot-test	Takanori Suzuki Sep 11, 2019
Add New Webhook to Workspace		

message with cURL

```
$ curl -X POST -H 'Content-type: application/json' \
> --data '{"text": "Hello Slack"}' \
> https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXX
```



beerbot APP 20:17
Hello Slack

message with Python

```
import json
from urllib import request

URL = 'https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXX'
data = {'text': 'Hello from Python!'}
jsoned = json.dumps(data)
req = request.Request(URL, data=jsoned, method='POST')
request.urlopen(req)
```



beerbot APP 20:19
Hello from Python!

message with Requests

```
import requests
```

```
URL = 'https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXX'  
data = {'text': 'Hello from Requests!'}  
requests.post(URL, json=data)
```



beerbot APP 20:21

Hello from Requests!

Complex message with Requests

```
fields = [ {'title': 'Love', 'value': 'Ferrets, :beer:, LEGO', 'short': True},  
          {'title': 'From', 'value': 'Japan :jp:', 'short': True}]  
data = { 'attachments': [{  
    'pretext': 'Nice to meet you!!',  
    'author_name': 'Takanori Suzuki',  
    'author_link': 'https://twitter.com/takanory/',  
    'text': '*THANK YOU* for coming to my talk !:tada: Please give me *feedback*!',  
    'fields': fields}]}  
requests.post(URL, json=data)
```

 **beerbot APP** 20:23
Nice to meet you!!

Takanori Suzuki
THANK YOU for coming to my talk ! Please give me feedback about this talk 
Love
Ferrets, , LEGO

From
Japan 

More complex message

- Block-Kit: new UI framework
- [Introducing Block Kit](#)
- [Block Kit Builder](#)

T Section

Image

Context

Divider

Actions

Section with...

Image

Button

Select

Multi-Select

Overflow

Datepicker

Fields

Message Preview ▾

Desktop ▾

Select a Template

Your App APP 2:37 AM

Hello, Assistant to the Regional Manager Dwight! **Michael Scott** wants to know where you'd like to take the Paper Company investors to dinner tonight.

Please select a restaurant:

Farmhouse Thai Cuisine

 1528 reviews

They do have some vegan options, like the roti and curry, plus they have a ton of salad stuff and noodles can be ordered without meat!! They have something for everyone here



Kin Khao

 1638 reviews

The sticky rice also goes wonderfully with the caramelized pork belly, which is absolutely melt-in-your-mouth and so soft.



Ler Ros

 2082 reviews

I would really recommend the Yum Koh Moo Yang - Spicy lime dressing and roasted quick marinated pork shoulder, basil leaves, chili & rice powder.



[Farmhouse](#)

[Kin Khao](#)

[Ler Ros](#)

```

1 { "blocks": [
2   {
3     "type": "section",
4     "text": {
5       "type": "mrkdwn",
6       "text": "Hello, Assistant to the Regional Manager Dwight! *Michael Scott* wants to
7       know where you'd like to take the Paper Company investors to dinner tonight.\n\n *Please select a
8       restaurant:*

```

@takanory

```
import requests

URL = 'https://hooks.slack.com/services/T0XXXXXX/YYYYYYYY/ZZZZZZZ'

places = [
    {
        'name': 'Masthead Brewing Co',
        'date': 'May 1, 2019',
        'address': '1261 Superior Ave E, Cleveland, OH',
        'tweet': 'https://twitter.com/takanory/status/1123746614768820224',
        'image': 'https://pbs.twimg.com/media/D5hZlOUwsAE8ipN?format=jpg',
        'beers': ['Tire Swing', 'Brute Force Double IPA'],
    },
    {
        'name': 'Southern Tier Brewery Cleveland',
        'date': 'May 2, 2019',
        'address': '811 Prospect Ave E, Cleveland, OH',
        'tweet': 'https://twitter.com/takanory/status/1124112560058503168'
    }
]
```

```
data = {'text': ''}
blocks = []
title = {
    'type': 'section',
    'text': {
        'type': 'mrkdwn',
        'text': ':beer: Beer memories from PyCon 2019',
    }
}
blocks.append(title) # add title
blocks.append({'type': 'divider'}) # add divider
```

```
for place in places: # create all place info
    s = {'type': 'section'}
    tweet = f"<{place['tweet']}>|Tweet"
    beers = ', '.join(place['beers'])
    s['text'] = {
        'type': 'mrkdwn',
        'text': f"*{place['name']}* {tweet}\n:beers:: {beers}",
    }
    s['fields'] = [ # fields
        {'type': 'mrkdwn', 'text': f"*Date*: {place['date']}"},
        {'type': 'mrkdwn', 'text': f"*Address*: {place['address']}"},
    ]
    s['accessory'] = { # image
        'type': 'image',
        'image_url': food['image'],
        'alt_text': food['name'],
    }
```



beerbot APP 15:45
Beer memories from PyCon 2019

Masthead Brewing Co ([Tweet](#))

🍺: Tire Swing, Brute Force Double IPA

Date: May 1, 2019

Address: 1261 Superior Ave E,
Cleveland, OH



Southern Tier Brewery Cleveland ([Tweet](#))

🍺: 8 Days A Week, Nu Juice IPA

Date: May 2, 2019

Address: 811 Prospect Ave E,
Cleveland, OH



Punch Bowl Social ([Tweet](#))

🍺: Columbus IPA, Cloud Cutter Hoppy Wheat Ale, Burning River Pale Ale,
Elvis Juice, Nano OG Lager

Date: May 3, 2019

Address: 1086 W 11th Pl,
Cleveland, OH



Noble Beast Brewing ([Tweet](#))

🍺: [Winter] Evil Motives, Sweet Amarillo

Date: May 5, 2019

Address: 1470 Lakeside Ave E (E.
13th St.), Cleveland, OH



Great Lakes Brewing Company ([Tweet](#))

🍺: Dortmunder, Eliot Ness

Date: May 7, 2019

Address: Concourse C (at CLE
Airport), Cleveland, OH



Goose Island Beer Company ([Tweet](#))

🍺: Green Line Pale Ale

Date: May 7, 2019

Address: Terminal 1, near Gate C10
(ORD Airport), Chicago, IL

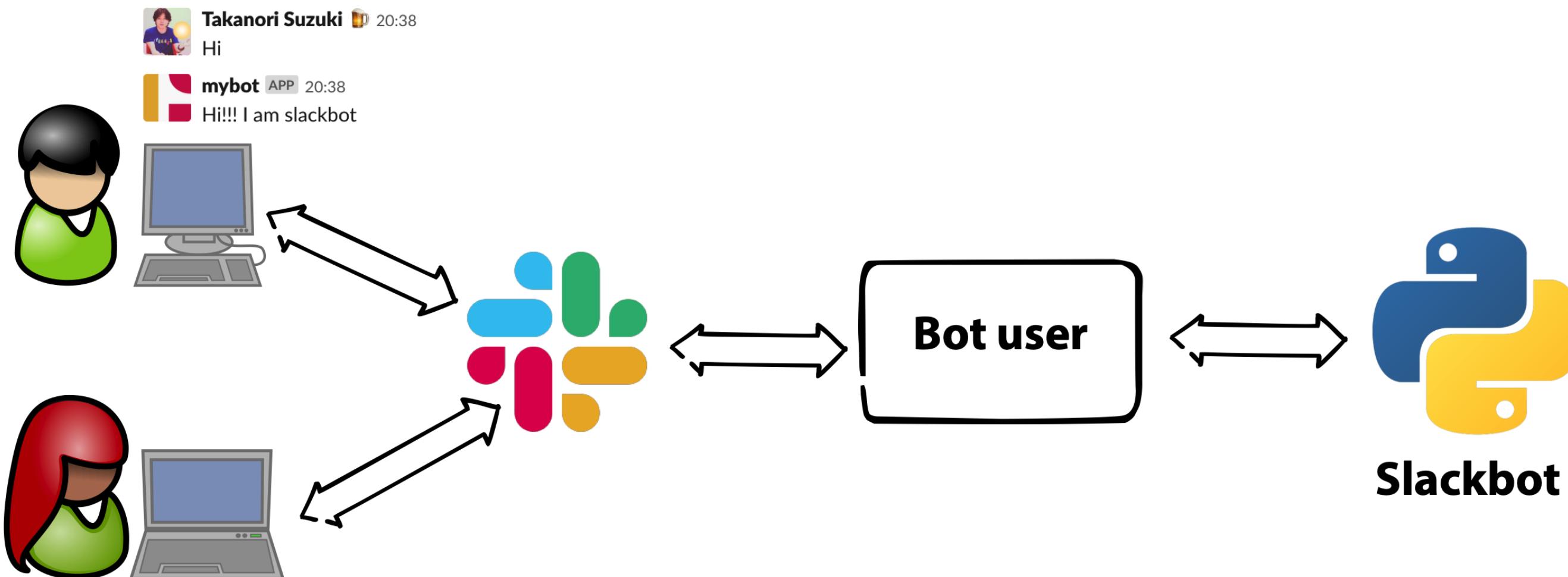


Summary of Incoming Webhooks

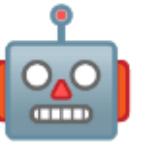
- Easy to send messages from programs 
- Create Complex messages 
- But One way (program  Webhook  Slack)

Interactive bot with slackbot library

System overview



Create bot user on Slack

- Create bot user 
- 1. Create a Slack app
- 2. Enable Bots
- 3. Add a "Bot User"
- 4. Install App to Workspace -> Authorize
- "Bot User OAuth Access Token": **xoxb-123467890-XXXXXX-XXXXXXXXXXXX**
- Invite Bot User to Slack channels
- see: [Creating a bot user](#)

Basic Information

Building Apps for Slack

Create an app that's just for your workspace (or build one that can be used by any workspace) by following the steps below.

Add features and functionality



Choose and configure the tools you'll need to create your app (or review all [our documentation](#)).

Incoming Webhooks

Post messages from external sources into Slack.

Interactive Components

Add buttons to your app's messages, and create an interactive experience for users.

Slash Commands

Allow users to perform app actions by typing commands in Slack.

Event Subscriptions

Make it easy for your app to respond to activity in Slack.

Bots

Add a bot to allow users to exchange messages with your app.

Permissions

Configure permissions to allow your app to interact with the Slack API.

Bot User

You can bundle a bot user with your app to interact with users in a more conversational manner. Learn more about [how bot users work](#).

[Add a Bot User](#)

Bot User

You can bundle a bot user with your app to interact with users in a more conversational manner. Learn more about [how bot users work](#).

Display name

beerbot

Names must be shorter than 80 characters, and can't use punctuation (other than apostrophes and periods).

Default username

beerbot

If this username isn't available on any workspace that tries to install it, we will slightly change it to make it work. Usernames must be all lowercase. They cannot be longer than 21 characters and can only contain letters, numbers, periods, hyphens, and underscores.

Always Show My Bot as Online

Off

When this is off, Slack automatically displays whether your bot is online based on usage of the RTM API.

Add Bot User

This app was created by a member of your workspace, PyCon JP.



On PyCon JP, beerbot would like to:

Confirm your identity on PyCon JP

Post to #random ▾

⚠ Add a bot user with the username @beerbot ➔

ncel

Allow

Installed App Settings

OAuth Tokens for Your Team

These tokens were automatically generated when you installed the app to your team.
You can use these to authenticate your app. [Learn more.](#)

OAuth Access Token

xoxp-2152101484-2152107267-756333156709

[Copy](#)

Bot User OAuth Access Token

xoxb-2152101484-756349000160-EKav7mWf

[Copy](#)

[Reinstall App](#)



Takanori Suzuki 🍺 20:40

てす



beerbot APP O

Not in channel



/invite @beer

bold

@takanory

slackbot library

- A simple chat bot for Slack
- Python 3.4+
- github.com/lins05/slackbot
- pip install slackbot

Install slackbot with venv

```
$ mkdir beerbot  
$ cd beerbot  
$ python3.7 -m venv env  
$ . env/bin/activate  
(env) $ pip install slackbot
```

Create a simple bot with slackbot

- `slackbot_settings.py`

```
API_TOKEN = "xoxb-123467890-XXXXXX-XXXXXXXXXXXXXX" # Bot token
PLUGINS = ['beerbott.plugins'] # Plugin packages
```

- `run.py`

```
from slackbot.bot import Bot

def main():
    bot = Bot()
    bot.run()

if __name__ == "__main__":
    main()
```

Simple Plugin

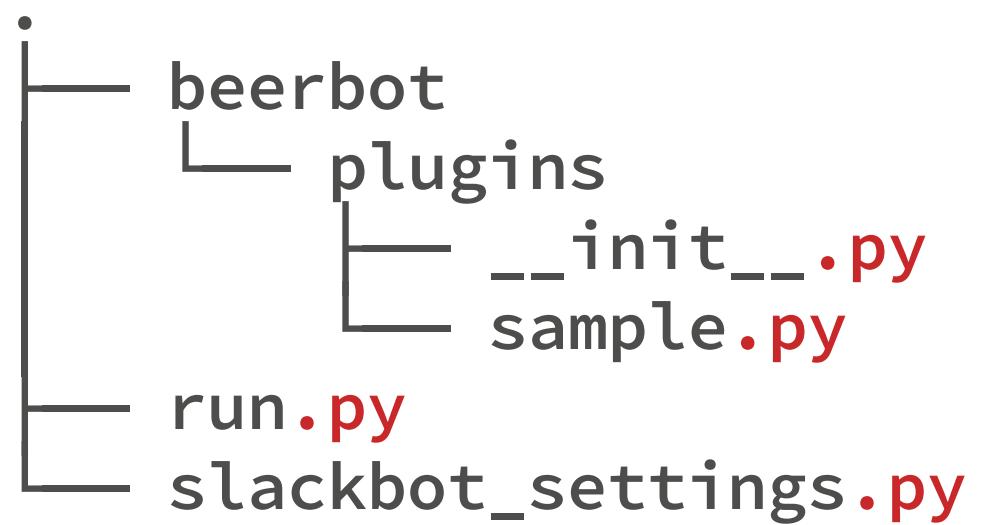
- **beerbot/plugins/__init__.py**

```
(env) $ mkdir -p beerbot/plugins  
(env) $ touch beerbot/plugins/__init__.py # empty file
```

- **beerbot/plugins/sample.py**

```
from slackbot.bot import listen_to  
  
@listen_to('Hi')  
def hello(message):  
    message.send('Hi!!! I am beerbot! :beer:')
```

File structure



Run slackbot

- (env) \$ python run.py



Takanori Suzuki 🍺 20:54

Hi



beerbot APP 20:54

Hi!!! I am beerbot! 🍺

Extend slackbot

decorator

- `listen_to`: message sent on a channel
- `respond_to`: message sent to the bot

```
from slackbot.bot import listen_to, respond_to
```

```
@listen_to('Hi')
def hello(message):
    message.send('Hi!!! I am beerbot! :beer:')
```

```
@respond_to('cheers') # mention
def ping(message):
    message.send('Cheers! :beers:')
```



Takanori Suzuki 14:40

Hi



beerbot APP 14:40

Hi!!! I am beerbot!



Takanori Suzuki 14:40

cheers

@beerbot cheers!



beerbot APP 14:40

Cheers!

mention

- **message.reply()**

```
@listen_to('morning')
def morning(message):
    message.reply('Good morning!') # reply to me
```



Takanori Suzuki 13:35

Good morning



beerbot APP 13:35

@takanory: Good morning!

emoji reaction

- `message.react()`

```
@listen_to('beer')
def hungry(message):
    message.react('beer') # react beer emoji
```



Takanori Suzuki 15:55

I want to drink craft beer in Pittsburgh!!!



1



Extract parameters on message

- Use regular expressions

```
@respond_to('choice (.*)') # choice pizza beer sushi
def choice(message, words): # -> words='pizza beer sushi'
    word = random.choice(words.split())
    message.send('I chose *{}*'.format(word))
```

```
@listen_to('(\d+)\s*beers') # 3 beers, 100beers
def bees(message, num): # num='3' or '100'
    beers = ':beer:' * int(num)
    if beers:
        message.send(beers)
```



Takanori Suzuki 13:39

3 beers



beerbot APP 13:39



Takanori Suzuki 13:39

100 beers



beerbot APP 13:39



slackbot_settings.py

```
ALIASES = '$' # Prefix instead of mention (@mybot ping -> $ping)
ERRORS_TO = 'mybot-error' # Error reporting channel
DEFAULT_REPLY = "Sorry but I didn't understand you"
PLUGIN = ['mybot.plugins', 'other.plugins',] # Pluing packages
```



Takanori Suzuki 15:31

\$choice beer wine sake



beerbot APP 15:31

I chose beer



Takanori Suzuki 15:32

\$ping



beerbot APP 15:32

@takanory: Sorry but I didn't understand you

Attachments support

```
import json

@respond_to('follow me')
def followme(message):
    attachments = [{  
        'author_name': 'Takanori Suzuki',  
        'text': 'Follow me! <https://twitter.com/takanory|@takanory>',  
        'color': '#59afe1'  
    }]
    message.send_webapi('', json.dumps(attachments))
```



Takanori Suzuki 15:34

\$follow me



beerbot APP 15:34

Takanori Suzuki

Follow me! @takanory

Summary of Slackbot

- We can **communicate** with Slackbot
- Slackbot can handle **parameters** in messages
- Slackbot can send messages in **various formats**

Case studies

Calculator function using SymPy

Calculator function using SymPy

- Motivation
 - I feel heavy to call a calculator app on my smartphone
 - It seems useful if Slack as a calculator

about SymPy

- SymPy: Python library for symbolic mathematics
 - www.sympy.org
- `$ pip install sympy`

• calc.py

```
from slackbot.bot import listen_to
from sympy import sympify, SympifyError

@listen_to(r'^([-+*/^%!().\d\s]+)$') # Formula like pattern
def calc(message, formula):
    try:
        result = sympify(formula) # Simplifies the formula
        if result.is_Integer:
            answer = int(result) # Convert to interger value
        else:
            answer = float(result) # Convert to float value
        message.send(f'{answer},')
    except SympifyError:
        pass
```



Takanori Suzuki 15:45

$1 + 1$



beerbot APP 15:45

2



Takanori Suzuki 15:45

$9500 / 3$



beerbot APP 15:45

3,166.6666666666665



Takanori Suzuki 15:45

$10^2 + 11^2 + 12^2$



beerbot APP 15:45

365



Takanori Suzuki 15:45

$10!$



beerbot APP 15:45

3,628,800

Plusplus function using Peewee ORM

Plusplus function using Peewee ORM

- Motivation
 - In PyCon JP, I want to make a culture that **appreciates** each other staff 

about Peewee

- Simple and small **ORM**.
 - a small, expressive ORM
 - python 2.7+ and 3.4+ (developed with 3.6)
 - supports sqlite, mysql and postgresql
- docs.peewee-orm.com
- `$ pip install peewee`

plusplus_model.py

```
from pathlib import Path
from peewee import SqliteDatabase, Model, CharField, IntegerField

dbpath = Path(__file__).parent / 'plusplus.db' # db file
db = SqliteDatabase(dbpath)

class Plusplus(Model):
    name = CharField(primary_key=True) # fields
    counter = IntegerField(default=0)

    class Meta:
        database = db

db.connect()
db.create_tables([Plusplus], safe=True)
```

plusplus.py

```
from slackbot.bot import listen_to

from .plusplus_model import Plusplus

@listen_to(r'^(\w+)\+\+')
def plusplus(message, name):
    target = name.lower()
    # Get or create object
    plus, created = Plusplus.get_or_create(
        name=target, defaults={'counter': 0})
    plus.counter += 1
    plus.save()
    message.send(f'Thank you {name}! (count: {plus.counter})')
```



Takanori Suzuki 15:56

staff++



beerbot APP 15:56

Thank you staff! (count: 1)



Takanori Suzuki 15:56

Staff++



beerbot APP 15:56

Thank you Staff! (count: 2)



Takanori Suzuki 15:56

beer++



beerbot APP 15:56

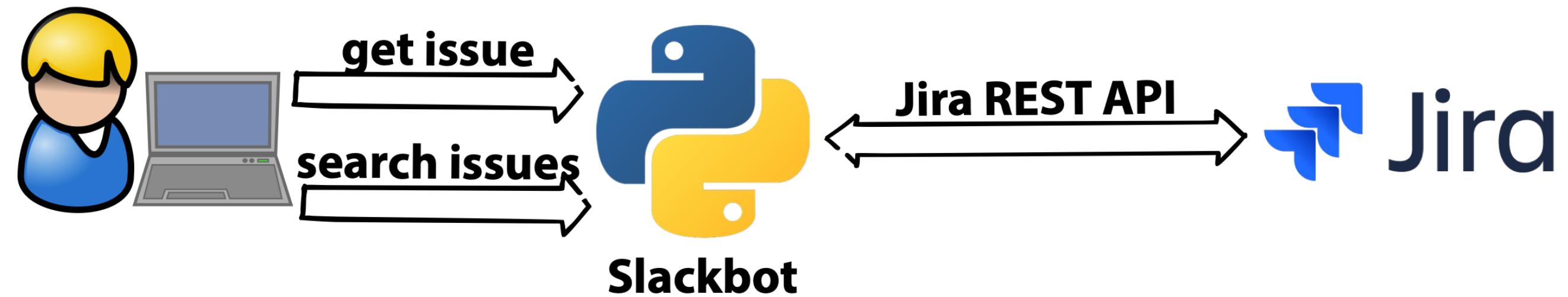
Thank you beer! (count: 1)

Display JIRA issue and Search issues

Display JIRA issue and Search issues

- Motivation
 - JIRA is very useful
 - JIRA Web is **very heavy**
 - I want to check issue details **without JIRA Web**

System Overview



about Python JIRA

- Python library to work with JIRA APIs
- jira.readthedocs.io
- `$ pip install jira`

Authentication

```
from jira import JIRA  
  
URL = 'https://jira.atlassian.com/'  
jira = JIRA(URL, basic_auth=('user', 'pass'))
```

- see: [2.1.2. Authentication](#)

Get Issue object

```
@listen_to(r'(ISSHA-[\d]+)') # Issue id pattern
def jira_issue(message, issue_id):
    issue = jira.issue(issue_id)
    summary = issue.fields.summary
    status = issue.fields.status.name
    assignee = issue.fields.assignee.name
    issue_url = issue.permalink()
    attachments = [
        'pretext': f'<{issue_url}|{issue_id}> {summary}',
        'fields': [{ 'title': 'Assignee', 'value': assignee},
                   { 'title': 'Status', 'value': status}],
    ]
    message.send_webapi('', json.dumps(attachments))
```

- see: 2.1.3 Issues



Takanori Suzuki 🍺 11:06

ISSHA-1484



mybot APP 11:06

ISSHA-1484 bot: amesh command

Assignee

takanory

Status

オープン

Search issues

```
@respond_to('jira (.*)')
def jira_search(message, keywords):
    # make JQL query string
    jql = f'project=ISSHA and text ~ "{keywords}" order by created desc'
    text = ''
    # get 5 recent issues
    for issue in jira.search_issues(jql, maxResults=5):
        id = issue.key
        url = issue.permalink()
        summary = issue.fields.summary
        text += f'- <{url}|{id}> {summary}\n'
    message.send(text)
```

- see: [2.1.5 Searching](#)
- JQL: JIRA Query Language
- see: [Advanced searching - Atlassian Documentation](#)



Takanori Suzuki 🍺 19:31

\$jira pycon apac



mybot APP 19:31

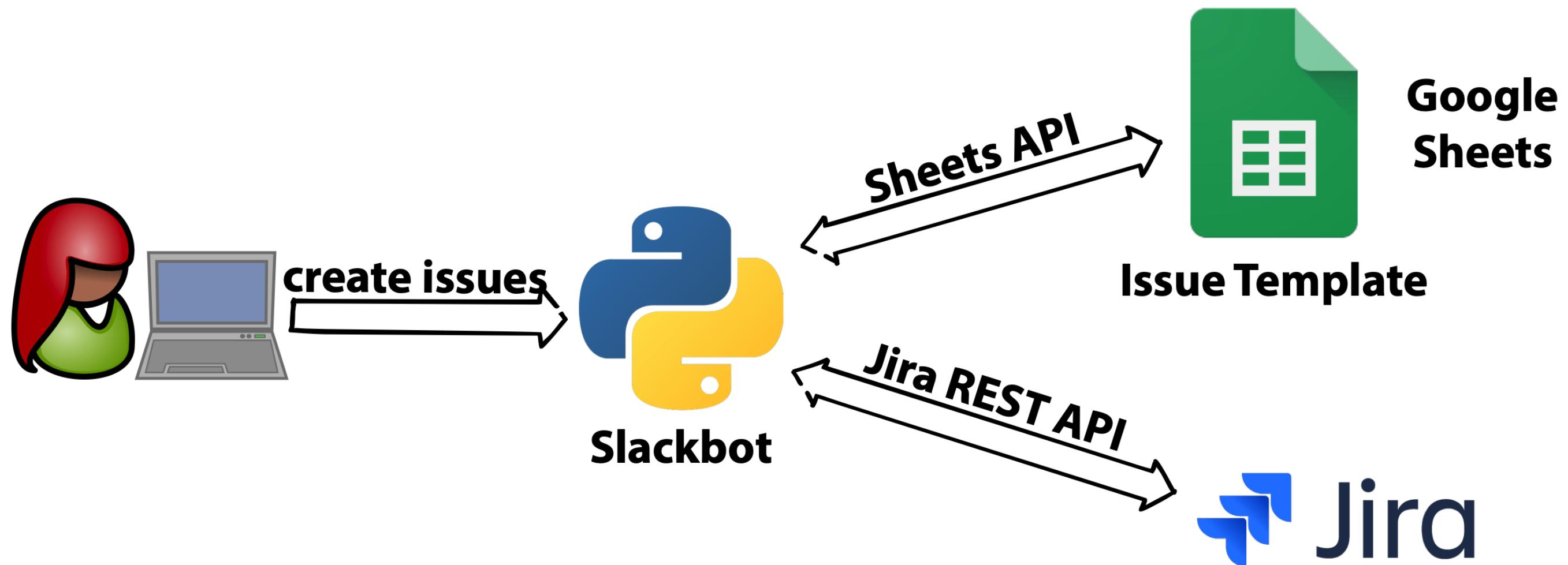
- [ISSHA-1549](#) OSC沖縄申込み
- [ISSHA-1457](#) 運営会議#26 (2018年12月) の準備
- [ISSHA-1301](#) PyCon JP 2019 座長募集を行う
- [ISSHA-1031](#) PyCon APAC 2018 シンガポール参加ツアーの段取り
- [ISSHA-1024](#) 一般社団法人PyCon JPの2017年の会計について報告する

Create multiple issues from a template

Create multiple issues from a template

- Motivation
 - In pycamp event, **20+ issues** are required for each event
 - Copying issues **by hand** is **painful**
 - JIRA Web is **very Heavy** (again)

System Overview



Google Authorization is VERY Complex(1/2)

- create Google Cloud Platform project
 - enable API(in this case: Google Sheets API)
 - download `credentials.json`
- Install Google Client Library
 - `pip install google-api-python-client google-auth-oauthlib`

Google Authorization is VERY Complex(2/2)

- download [quickstart.py](#) on GitHub
- run `quickstart.py`
 - select your Google account in Web browser
 - Click the Accept button
 - get `token.pickle` file(finish!!)

```
(env) $ python quickstart.py
Please visit this URL to authorize this application: https://accounts.goc
Name, Major:
Alexandra, English
:
```

- see: [Python Quickstart | Sheets API | Google Developers](#)

Get Spreadsheet Data

```
from googleapiclient.discovery import build
SHEET_ID = '1LEtpNewhAFSXXXXXXXXXXXXXX'

creds = None
if os.path.exists('token.pickle'):
    with open('token.pickle', 'rb') as token:
        creds = pickle.load(token)
# build service
service = build('sheets', 'v4', credentials=creds)
# get data from Spreadsheet
result = service.spreadsheets().values().get(
    spreadsheetId=SHEET_ID, range='sheet_name!A:G').execute()
for row in result.get('value', []):
    print(row[0], row[1]) # Cell A and B
```

- see: Method: spreadsheets.values.get

Create JIRA Issue

```
duedate = datetime.date.today + datetime.timedelta(days + 14)
issue_dict = {
    'project': {'key': PROJECT},
    'components': [{'name': COMPONENT}],
    'summary': SUMMARY_TEXT,
    'description': LONG_DESCRIPTION,
    'assignee': {'name': NAME_OF_ASSIGNEE},
    'reporter': {'name': NAME_OF_REPORTER},
    'duedate': f'{duedate:%Y-%m-%d}'}
issue = jira.create_issue(fields=issue_dict) # create issue
```

- see: [2.1.3 Issues](#)

Sample of template command

- `$pycamp create`: Create issues for pycamp event

Place **Target date** **Staff name**

 **Tomohiro Kobayashi** 21:58
\$pycamp create 和歌山 4/20 kobatomo ottidododo massa142

 **pyconjpbot** APP 22:00
チケットを作成しました: <https://pyconjp.atlassian.net/browse/ISSHA-1521>

Bot created parent issue

pycampチケットテンプレート

ファイル 編集 表示 挿入 表示形式 データ ツール アドオン ヘルプ 最終編集: 2018年11月27日 (Ryuji Tsutsui さん)

fx | カテゴリー Due date

Category	Assignee	Summary	Description
1	2	3	4
1.事前準備	コアスタッフ	-30 connpassイベント公開(現地スタッフ)	<p>h2. 目的</p> <ul style="list-style-type: none"> * 参加者を募るために、イベント本体、懇親会のconnpassイベントを作成して公開する * イベント開催の一ヶ月前くらいには公開したい <p>h2. 内容</p> <ul style="list-style-type: none"> * connpassイベントを過去イベントからコピーしてベースを作成(コアスタッフが実施) ** タイトルを変更する ** 日時を変更する ** 参加人数を消す ** 場所を未設定状態にする ** ロゴ画像を削除する ** アンケートを過去イベントからコピーする * 現地スタッフは以下を記述 ** ロゴ設定 ** 会場設定 ** 説明文の修正(場所、講師など) * 公開前にコアスタッフに内容のレビュー依頼をSlackでする * 公開のタイミングは平日の昼間が狙い目(多くの人に見てもらえる) * 懇親会のconnpassは場所未定で公開でもOKです。公開後に早めに場所を決めてください <p>以下、イベント公開に関するマニュアルも参照してください。 http://bootcamp-text.readthedocs.io/organize/1_manual.html#id8</p>
1.事前準備	講師	-30 ホテル、移動の手配(講師)	<p>h2. 目的</p> <ul style="list-style-type: none"> * 現地往復の交通手段を事前に予約する * ホテルの予約をする <p>h2. 内容</p> <ul style="list-style-type: none"> * 予約した内容と金額の記載をする * 支払いは、Python Boot Camp 終了後に別チケットで行います
1.事前準備	コアスタッフ	-21 事前打ち合わせ(主: コアスタッフ)	<p>h2. 目的</p> <ul style="list-style-type: none"> * 現地スタッフ、コアスタッフ、講師のやること認識合わせのために事前打ち合わせを実施する * Slack の Call を使う想定 <p>h2. 内容</p> <ul style="list-style-type: none"> * pyconjp-fellow.slack.comに#地域名(県単位)のチャンネルを作っておく * 作ったらpyconjp.slack.comの#pycampで現地スタッフ・TA・講師に報告する
			□ 未登録
			+ 三 親タスク サブタスク

ISSHA-1521

Python Boot Camp in 和歌山を開催

編集 コメント 割り当て 処理中止 課題の解決 課題のクローズ 管理 ▾

タイプ: タスク
優先度: 重度
コンポーネント: Python Boot Camp
ラベル: なし

ステータス: 進行中 (ワークフローを表示)
解決状況: 未解決

説明

目的

- Python Boot Camp in 和歌山を開催するのに必要なタスクとか情報をまとめる親タスク

内容

- 日時: 2019-04-20(Sat)
- 会場: T-LABO
- 現地スタッフ: 落合大輔
- 講師: [Masataka Arai](#)
- TA: 堀口日向, Shimasan
- イベントconnpass: <https://pyconjp.connpass.com/event/120116/>
- 懇親会connpass: <https://pyconjp.connpass.com/event/120117/>

JIRAの使い方については以下マニュアルを参照してください。

http://bootcamp-text.readthedocs.io/organize/1_manual.html#id28

1.事前準備

- [ISSHA-1522](#) CLOSED connpassイベント公開(現地スタッフ)
- [ISSHA-1523](#) OPEN ホテル、移動の手配(講師)
- [ISSHA-1524](#) IN PROGRESS 事前打ち合わせ(主: コアスタッフ)

2.広報

- [ISSHA-1525](#) CLOSED 事前ブログ執筆(現地スタッフ)
- [ISSHA-1526](#) IN PROGRESS メディアスポンサー経由での告知(コアスタッフ)
- [ISSHA-1527](#) CLOSED Twitter定期通知(コアスタッフ)
- [ISSHA-1528](#) IN PROGRESS 現地メディア経由での告知(現地スタッフ)

3.イベント直前、当日

- [ISSHA-1529](#) OPEN 参加者への事前連絡(現地スタッフ)
- [ISSHA-1530](#) OPEN お茶、お菓子購入(現地スタッフ)
- [ISSHA-1531](#) OPEN 案内スライド作成(現地スタッフ)
- [ISSHA-1532](#) OPEN 座席表作成、印刷(現地スタッフ)
- [ISSHA-1533](#) IN PROGRESS ランチミーティング(主: 現地スタッフ)
- [ISSHA-1534](#) OPEN 参加者アンケート実施(現地スタッフ)

4.事後処理

- [ISSHA-1535](#) OPEN 精算処理(現地スタッフ)
- [ISSHA-1536](#) OPEN 精算処理(講師)
- [ISSHA-1537](#) OPEN 事後ブログ/現地スタッフ

Source code of pycamp command

- see:

github.com/pyconjp/pyconjpbot/blob/master/pycamp.py

I never have to copy issues again

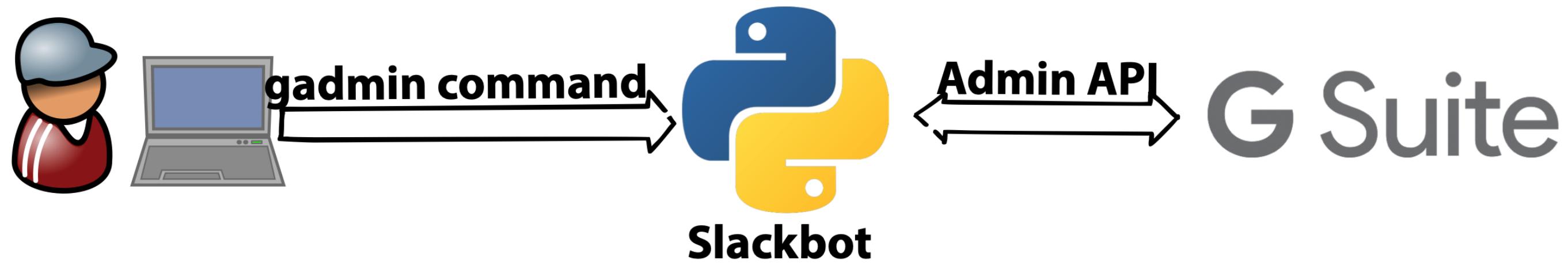


Account management of G Suite

Account management of G Suite

- Motivation
 - PyCon JP use `pycon.jp` domain with G Suite
 - I only use Google Admin web occasionally
 - I forgot to use admin screen

System Overview



Update Google Authorization

- update Google Cloud Platform project
 - add G Suite Admin API
 - re-download `credentials.json`
- re-run `quickstart.py`
 - get new `token.pickle`

```
(env) $ python quickstart.py
Please visit this URL to authorize this application: https://accounts.goc
Name, Major:
Alexandra, English
:
```

Get user list

```
# build service
DOMAIN = 'pycon.jp'
service = build('admin', 'directory_v1', credentials=creds)

# get user list
users_list = service.users().list(orderBy='email', domain=DOMAIN).execute()
for user in users_list.get('users', []):
    email = user['primaryEmail']
    fullname = user['name']['fullName']
    print(f'{email} {fullname}')
```

- see: [Users: list | Directory API](#)

Insert user

```
body = {  
    'primaryEmail': EMAIL_ADDRESS,  
    'password': PASSWORD,  
    'name': {  
        'givenName': FIRST_NAME,  
        'familyName': LAST_NAME,  
    },  
}  
  
try:  
    service.users().insert(body=body).execute()  
except:  
    pass # fail
```

- see: [Users: insert | Directory API](#)

Suspend, Resume, Delete user

```
suspend = {'suspended': True}  
service.users().update(userKey=EMAIL, body=suspend).execute()
```

```
resume = {'suspended': False}  
service.users().update(userKey=EMAIL, body=resume).execute()
```

```
service.users().delete(userKey=email).execute()
```

- see: [Users: update | Directory API](#)
- see: [Users: delete | Directory API](#)



Takanori Suzuki 🍺 11:25

\$gadmin help



pyconjpbot APP 11:25

ユーザー管理

- `$gadmin user list`: ユーザーの一覧を返す
- `$gadmin user insert (ユーザー) (名前) (名字)`: ユーザーを追加する
- `$gadmin user delete (ユーザー)`: ユーザーを削除する(停止中のみ削除可)
- `$gadmin user reset (ユーザー)`: ユーザーのパスワードをリセットする
- `$gadmin user suspend (ユーザー)`: ユーザーを停止する(停止中にする)
- `$gadmin user resume (ユーザー)`: ユーザーを再開する(アクティブにする)

メールのエイリアス管理

- `$gadmin alias list (ユーザ)`: ユーザーのエイリアスの一覧を返す
- `$gadmin alias insert (ユーザ) (エイリアス)`: ユーザーにエイリアスを追加する
- `$gadmin alias delete (ユーザ) (エイリアス)`: ユーザーからエイリアスを削除する

グループ管理

- `$gadmin group list`: グループの一覧を返す
- `$gadmin group insert (グループ) (グループ名)`: 指定したグループを追加する
- `$gadmin group delete (グループ)`: 指定したグループを削除する

グループのメンバー管理

- `$gadmin member list (グループ)`: 指定したグループのメンバー一覧を返す
- `$gadmin member insert (グループ) (メール1) [(メール2...)]`: 指定したグループにメンバーを追加する
- `$gadmin member delete (グループ) (メール1) [(メール2...)]`: 指定したグループからメンバーを削除する



Takanori Suzuki 🍺 11:27

\$gadmin user list



pyconjpbot APP 11:27

pycon.jp ドメインのユーザー一覧(39ユーザー)

- 2015@pycon.jp PyCon JP 2015事務局
- 2016@pycon.jp PyCon JP 2016
- 2017@pycon.jp PyCon JP 2017
- association@pycon.jp 一般社団法人会計
- event@pycon.jp イベント会計
- pycon.jp user admin
- araki@pycon.jp Ai Araki
- arakawa@pycon.jp Masataka Arakawa
- cc@pycon.jp JP Project
- hanabusa@pycon.jp Yuki Hanabusa SHIMOTO
- ian@pycon.jp Ian
- iqbal@pycon.jp Iqbal Ahmad
- kikuchi@pycon.jp Makoto Kikuchi
- kirk@pycon.jp Fumiko Kirk
- kondo@pycon.jp Tomohiro Kobayashi
- kondo@pycon.jp Kaori Kondo
- m@pycon.jp 関根タマム
- manami.noda@pycon.jp Noda Manami
- matsuo@pycon.jp Ryosuke Matsuo
- matsuyama@pycon.jp Ryosuke Matsuyama
- matsu@pycon.jp メディアマツ
- pr@pycon.jp プロジェクトマネージャー
- pycon@pycon.jp Book Python
- pycon@pycon.jp pycon.jp
- ryukamiya@pycon.jp Ryukami Kamiya
- shimizukawa@pycon.jp Shizuka Shimizukawa
- sotomatsu@pycon.jp スオトマツ
- suyama@pycon.jp Suyama
- syo@pycon.jp Aoi Syo
- syn@pycon.jp Vasya Syn - 管理用2016以降用
- syushida@pycon.jp Shuji Shida
- ta@pycon.jp Takao Suzuki
- te@pycon.jp Yuya Tezuka
- temanami@pycon.jp Mana Tezuka
- tsmamikawa@pycon.jp Mamiko Shimizukawa



Takanori Suzuki 🍺 11:30

\$gadmin group list



pyconjpbot APP 11:30

pycon.jp ドメインのグループ一覧(21グループ)

- 20 [REDACTED]@pycon.jp 2015グループ(2ユーザー)
- 20 [REDACTED]@pycon.jp 2016グループ(9ユーザー)
- 20 [REDACTED]@pycon.jp 2017グループ(6ユーザー)
- 20 [REDACTED]day-staff@pycon.jp 2018当日スタッフDocs権限用(16ユーザー)
- 20 [REDACTED]staff@pycon.jp 2019コアスタッフDocs権限用(11ユーザー)
- acc [REDACTED]nin@pycon.jp 一般社団法人会計責任者グループ(2ユーザー)
- acc [REDACTED]ociation-group@pycon.jp 一般社団法人会計用グループアドレス(5ユーザー)
- acc [REDACTED]ent-group@pycon.jp イベント会計用グループアドレス(3ユーザー)
- app [REDACTED]on.jp AppleDeveloperProgram管理(2ユーザー)
- boar [REDACTED]on.jp 法人理事会(7ユーザー)
- cor [REDACTED]up@pycon.jp contactグループ(4ユーザー)
- des [REDACTED]on.jp デザイン(5ユーザー)
- hir [REDACTED]pycon.jp PyCon mini Hiroshima(1ユーザー)
- ma [REDACTED]it-group@pycon.jp 事務局グループ(8ユーザー)
- me [REDACTED]o@pycon.jp メディアグループ(6ユーザー)
- prc [REDACTED]oup@pycon.jp プログラムグループ(6ユーザー)
- pycamp-[REDACTED]@pycon.jp Python Boot Camp(4ユーザー)
- pycc [REDACTED]up@pycon.jp pyconjp@pycon.jp の転送用グループ(3ユーザー)
- spor [REDACTED]up@pycon.jp スポンサーグループ(4ユーザー)
- sym [REDACTED]group@pycon.jp 2016(以降)のWebsiteのアラート送付先(4ユーザー)
- venu [REDACTED]o@pycon.jp 会場グループ(15ユーザー)



Takanori Suzuki 🍺 11:30

\$gadmin member list pycamp-[REDACTED]



pyconjpbot APP 11:30

pycamp-[REDACTED]@pycon.jp グループのメンバー(4ユーザー)

- kol [REDACTED]@pycon.jp
- o6z [REDACTED]@pyconjp.slack.com
- ry [REDACTED]@gmail.com
- tal [REDACTED]@pycon.jp

I can completely forget Google
Admin



Source code of gadmin command

- see:

github.com/pyconjp/pyconjpbot/blob/master/pyconjpbot/gadmin.py

Summary

- Simple bot with Incoming Webhooks
- Interactive bot with `slackbot`
- `slackbot` with libraries and apis

Next steps

- Let's make **your own bot**
- Let's use libraries and APIs in your bot
- Automate your Boring Stuff with Slackbot



Takanori Suzuki 17:14

\$trans ありがとうございました。ピツツバーグで来年会いましょう！ en



beerbot APP 17:14

Thank you very much. See you next year in Pittsburgh!

-  [@takanory](https://twitter.com/takanory)
-  github.com/takanory/slides

\$trans command

- pip install googletrans

```
from googletrans import Translator
from slackbot.bot import respond_to

@respond_to('^\$trans\s+(.+)\s+([\w-]+)$')
def trans(message, text, dest):
    translator = Translator()
    t = translator.translate(text, dest=dest)
    message.send(t.text)
```



Takanori Suzuki 17:14

\$trans ありがとうございました。ピツツバーグで来年会いましょう！ en



beerbot APP 17:14

Thank you very much. See you next year in Pittsburgh!

-  [@takanory](https://twitter.com/takanory)
-  github.com/takanory/slides