SYMPHONY

The Sidekick Bot

July 24, 2016 cft@symphony.com

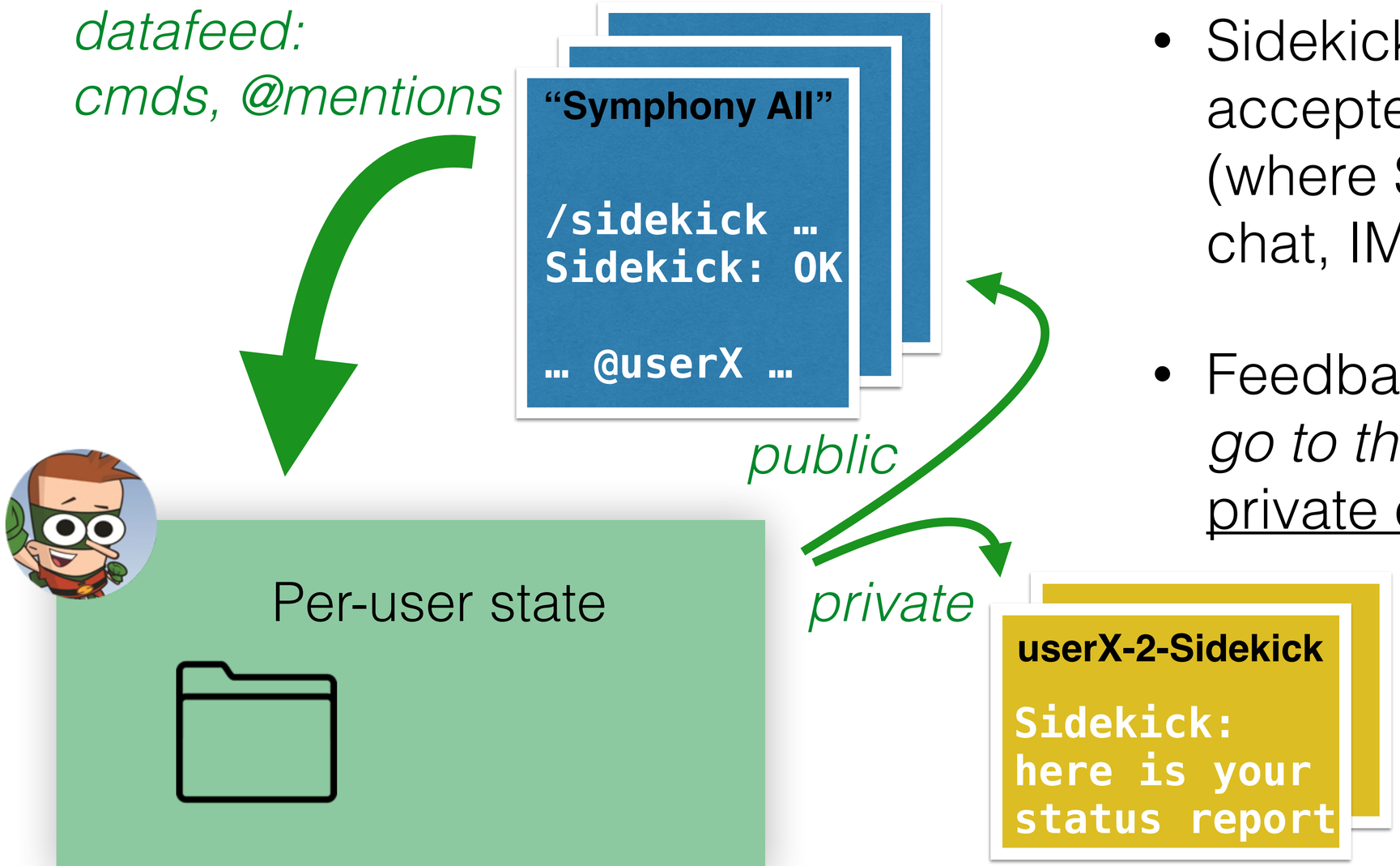
Sidekick in a nutshell

- Sidekick = a stateful bot written in Python
- Demonstrated services, PoC for:
  - automatic **out-of-office messages**
    (sent the first time the absent user is mentioned, once a day, per room)
  - **pre-scheduled announcements** (e.g. repetitive reminders, single+all rooms)
  - user-logging and **content-watch** using arbitrary regex
    (forwarding content of interest to subscribing user, also as a daily digest)
  - **bulk room management** (e.g. promote all room members with single cmd line)

  Feature: define personalized shortcuts to trigger the bot, complex cmds

  Speciality: IM chat between user and bot becomes a private control channel
- Lessons learned (see 2nd part of slide deck)

General Sidekick Architecture



- Sidekick commands are accepted from any room (where Sidekick is a member): chat, IM, MIM
- Feedback and public output *go to the same room*, private output *goes to the IM*

Example of a stateful service: OOO

datafeed: @mentions

Per-user state

“OOO msg on behalf of userX: absent till 8/1/16 (in NYC)”

- Sidekick scans all @mentions. If matching with a user having an OOO task: notify this room at most once a day.
- OOO-task can be for a single room **or** for all rooms shared with Sidekick
- Per-user state for the OOO service:

```
’ooo’ : [
    { ‘room’ : ‘*’,
      ’till’ : ‘m/d/y’,
      ‘msg’  : ‘in NYC’,
      ‘notified’ : {
           roomId1 : ‘m/d/y’,
           roomId2 : ‘m/d/y’
      }
    },
    { ‘room’ : ‘ES | internal’,
      …
```

Screenshots

Christian Tschudin 19:23
/sidekick status

Sidekick 19:23
@Christian Tschudin A status report will be sent to your Sidekick chat room shortly.

Christian Tschudin 19:31
/sidekick ooo -all 8/20/16 in NYC

Sidekick 19:31
@Christian Tschudin
Your new OOO message is active until 8/20/16: in NYC (in all rooms)

and please talk to @Christian Tschudin regarding X

Sidekick 19:32
This is an out-of-office message on behalf of Christian Tschudin:
away until 8/20/16 because of "in NYC"

Sidekick

Christian Tschudin 19:22
/sidekick help

Sidekick 19:22
General command format:
/sidekick [COMMAND [PARAMETERS]]
or
/sk [COMMAND [PARAMETERS]]
Commands are: ?, alias, announce, intro, manage, ooo, status, watch.
Try "/sidekick examples" or "/sidekick help CMD" for more details.

Christian Tschudin 19:22
/sidekick examples

Sidekick 19:22
Sidekick examples
See your definitions stored with Sidekick:
/sidekick status
Define an out-of-office auto-reply (posted at most once a day):
/sidekick ooo 8/20/2016 in NYC
Define an out-of-office msg for all rooms:
/sidekick ooo -all 8/20/2016 in NYC
Cancel an ooo entry, or all of them
/sidekick ooo cancel 2
/sidekick ooo cancel
Define a shorter Sidekick trigger for you:
/sidekick alias ./=/sidekick
and then call Sidekick with "./ help"

Lessons Learned 1: Bot Etiquette

- Bots are intrusive and can easily become a nuisance:
  - collect user-behavior
  - compete for “keyword real-estate”
  - react on each other

- Need rules, e.g.:

  - **opt-in:** a bot MUST obtain each user’s approval. Could be a user-2-bot IM chat, but:

  - Better when bot subscription is managed by Symphony and enforced (cleansed datafeed)

Lessons Learned 2: little added value

- The agent/bridge does not provide much added value:
  - delicate trust relationship (against the all-in-the-cloud principle)
  - no improvement on the API (mostly pass-through)

- Recommendation:
  - retire bridge
  - replace agent by a bot-side library
  - separate protocols: “cache replication” vs “data access” (see appendix)
  - back-port datafeed to main Symphony product (instead of letting mobile clients use PubNub)

Appendix: vertical vs horizontal protocol

- See SymphonyOSF Git: “session-less-client”

- Vertical: offers “cooked data” using a library (instead of an agent)

- Symphony 2.0?

An experiment with a “replica” approach

- Symphony = cloud storage for e2e-encrypted messages
- Client SW talks to ***local* message store**, treats Symphony as a backup
- Vertical protocol (client to local store): traverse and read from a “cooked tree”
- Horizontal protocol (local store to cloud): “raw tree” replication, fetch-on-demand
- Logical names for all nodes