Telegram

[telegram, bot]

channel

send message to channel from bot (BOT*API*KEY required):

1. add bot to channel as admin
2. send message

```
curl
https://api.telegram.org/bot[BOT_API_KEY]/sendMessage?chat_id=@[MY_CHANNEL_NAME]&text=[MY_MESSAGE_TEXT]
```

BOT

Create BOT with https://telegram.me/botfather

```
TOKEN=xyz
```

registrare un webhook (consigliato)

```
pip install python-telegram-bot ipython

ipython
import telegram
TOKEN="xyz"
bot = telegram.Bot(TOKEN)
bot.setWebhook("https://giano.comune.verona.it/giano")
```

registrare un webhook (non funziona)

```
curl -i -H "Accept: application/json" -H "Content-Type: application/json" \
  -X POST --data-urlencode '{"url":"https://giano.comune.verona.it/giano"}'
\
  https://api.telegram.org/bot${TOKEN}/setWebhook
```

```
curl -s -X POST https://api.telegram.org/bot${TOKEN}/sendMessage \
-d text="this is a message" \
-d chat_id=73496590 \
| jq .
```

```
curl -s -X POST https://api.telegram.org/bot${TOKEN}/getUpdates | jq .
```

certificato self signed

```
openssl req -newkey rsa:2048 -sha256 -nodes -keyout key.pem -x509 -days 365
-out cert.pem -subj "/C=US/ST=New York/L=Brooklyn/O=Example
Company/CN=example.com"

curl -F url="https://example.com:8443/<token>" -F certificate@=cert.pem
```

Add BOT to group

- With BotFather click /setjoingroups, choose BOT
- Add @BOT to group
- Send a message to group
- Give .result.message.chat.id from curl https://api.telegram.org/bot${TOKEN}/getUpdates (negative number) <code bash> TOKEN= curl https://api.telegram.org/bot${TOKEN}/getUpdates | jq . </code>

Send a message to group

```
TOKEN=
CHATID=
curl -X POST "https://api.telegram.org/bot${TOKEN}/sendMessage" -d
"chat_id=${CHATID}&text=my sample text"
```

To enable BOT to receive messages from group disable “group privacy” of BOT

curl

send message

```
#/bin/sh

API="112212222:XXX..."
CHATID=123456
TEXT="$*"

curl --data chat_id=$CHATID --data-urlencode "text=$TEXT"
"https://api.telegram.org/bot$API/sendMessage"
```

bot video

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""Basic example for a bot that uses inline keyboards.

# This program is dedicated to the public domain under the CC0 license.
"""
```

```
TOKEN="xxx:xxx"
AUTHORIZED=[73496590,483703779,534914573,536325022]

import logging
from telegram import InlineKeyboardButton, InlineKeyboardMarkup,
ReplyKeyboardMarkup
from telegram.ext import Updater, CommandHandler, CallbackQueryHandler
import subprocess

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s -
%(message)s',
                    level=logging.INFO)
logger = logging.getLogger(__name__)


def _isAuthorized(update):
    isOk = update.effective_user['id'] in AUTHORIZED
    if not isOk:
        update.message.reply_text('maybe another time, you are %s' %
update.effective_user['id'])
    return isOk


def reply(update, message):
    reply_keyboard = [['/show', '/motion', '/day']]
    update.message.reply_text(message,
reply_markup=ReplyKeyboardMarkup(reply_keyboard, resize_keyboard=True,
one_time_keyboard=False))


def start(bot, update):
    if _isAuthorized(update):
        reply(update, "welcome")
        #update.message.reply_text('comandi',
reply_markup=ReplyKeyboardMarkup(reply_keyboard, resize_keyboard=True,
one_time_keyboard=False))

def show(bot, update):
    print("show ...")
    if _isAuthorized(update):
        user_id = update.effective_user['id']
        subprocess.call(["/home/pi/preview.sh", str(user_id)], shell=False)
        reply(update, 'recording ... wait 15 seconds ...')

def motion(bot, update):
    print("show latest motion ...")
    if _isAuthorized(update):
        user_id = update.effective_user['id']
        reply(update, 'send latest motion video ...')
        subprocess.call(["/home/pi/latest.sh", str(user_id)], shell=False)
```

```
def error(bot, update, error):
    """Log Errors caused by Updates."""
    logger.warning('Update "%s" caused error "%s"', update, error)


def previewday(bot, update):
    if _isAuthorized(update):
        user_id = update.effective_user['id']
        message = update.message['text'][1:].split(" ")
        camera = message[0]
        try:
            speed = int(message[1])
        except:
            speed = 50
        reply(update, 'send latest 24 hours from camera %s at speed %s, wait
5 minutes ...' % (camera, speed))
        makedayvideo(user_id, camera, speed=speed)


def makedayvideo(user_id, camera, speed=50):
        print(["/home/pi/preview-day.sh", str(user_id), camera, str(speed)])
        subprocess.call(["/home/pi/preview-day.sh", str(user_id), camera,
str(speed)], shell=False)

def button(bot, update):
    query = update.callback_query
    chat_id=query.message.chat_id

    bot.edit_message_text(text='send latest 24 hours from camera %s at speed
50, wait 5 minutes ...'.format(query.data), chat_id=chat_id,
message_id=query.message.message_id)
    makedayvideo(chat_id, query.data, speed=50)


def buttonpreview(bot, update):
    if _isAuthorized(update):
        keyboard = [[InlineKeyboardButton("1", callback_data='1'),
            InlineKeyboardButton("2", callback_data='2'),
            InlineKeyboardButton("3", callback_data='3'),
            InlineKeyboardButton("4", callback_data='4'),
            InlineKeyboardButton("5", callback_data='5'),
            ]]
        reply_markup = InlineKeyboardMarkup(keyboard)
        update.message.reply_text('camera:', reply_markup=reply_markup)

def main():
    updater = Updater(TOKEN)

    updater.dispatcher.add_handler(CommandHandler('start', start))
    updater.dispatcher.add_handler(CommandHandler('show', show))
    updater.dispatcher.add_handler(CommandHandler('motion', motion))
```

```
    updater.dispatcher.add_handler(CommandHandler('1', previewday))
    updater.dispatcher.add_handler(CommandHandler('2', previewday))
    updater.dispatcher.add_handler(CommandHandler('3', previewday))
    updater.dispatcher.add_handler(CommandHandler('4', previewday))
    updater.dispatcher.add_handler(CommandHandler('5', previewday))
    updater.dispatcher.add_handler(CommandHandler('day', buttonpreview))
    updater.dispatcher.add_handler(CallbackQueryHandler(button))
    updater.dispatcher.add_error_handler(error)

    updater.start_polling()
    updater.idle()


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