NeMo-efishery Documentation

Release stable

Aug 18, 2021

Contents

1 How it Work **3**

2 Requirements **5**

3 How to run **7**
- 3.1 Generate the whatsapp session . . . 7
- 3.2 Testing run the bot with example . . . 7
- 3.3 Run the nemo with docker . . . 8

4 Understanding Basic Coral Configuration **9**
- 4.1 Author . . . 9
- 4.2 Default Greeting . . . 9
- 4.3 Commands . . . 10
- 4.4 Message Format Parser . . . 10
- 4.5 Process . . . 11
- 4.6 Webhook . . . 12
- 4.7 Logging . . . 12
- 4.8 Expected Users . . . 12

Customizeable Whatsapp chatbot with simply creating a YML file to configure the bot, very easy to configure.

Why you need this ? imagine you want to automate your shop order or catalog your shop, or just simply don't want to reply some annoying friend

The tools is awesome it just the documentation that looks shitty, help me to fix the documentation :(

This tools is the study case research from the eFishery to solve problem "How might we gather data supply from farmer it self" so we create a bot whatsapp because farmer is very familiar with whatsapp and facebook rather than with new apps

Whatsapp NeMo

From the 87 registered farmer, in 2 days 80% farmer is easily input the data, to achieve 100% input from farmer need 5 days

	Blasted (Reach)	Read	Message to Bot	Try trigger	Trigger succes	Filling data	Complete filling
NeMo	88	47	13	10	6	6	2
Group community	19	-					
SMS blast	84	-					

++ 2 farmer unregistered chat to offer supply, but we can't follow this up because bot can't handle this terdaftar

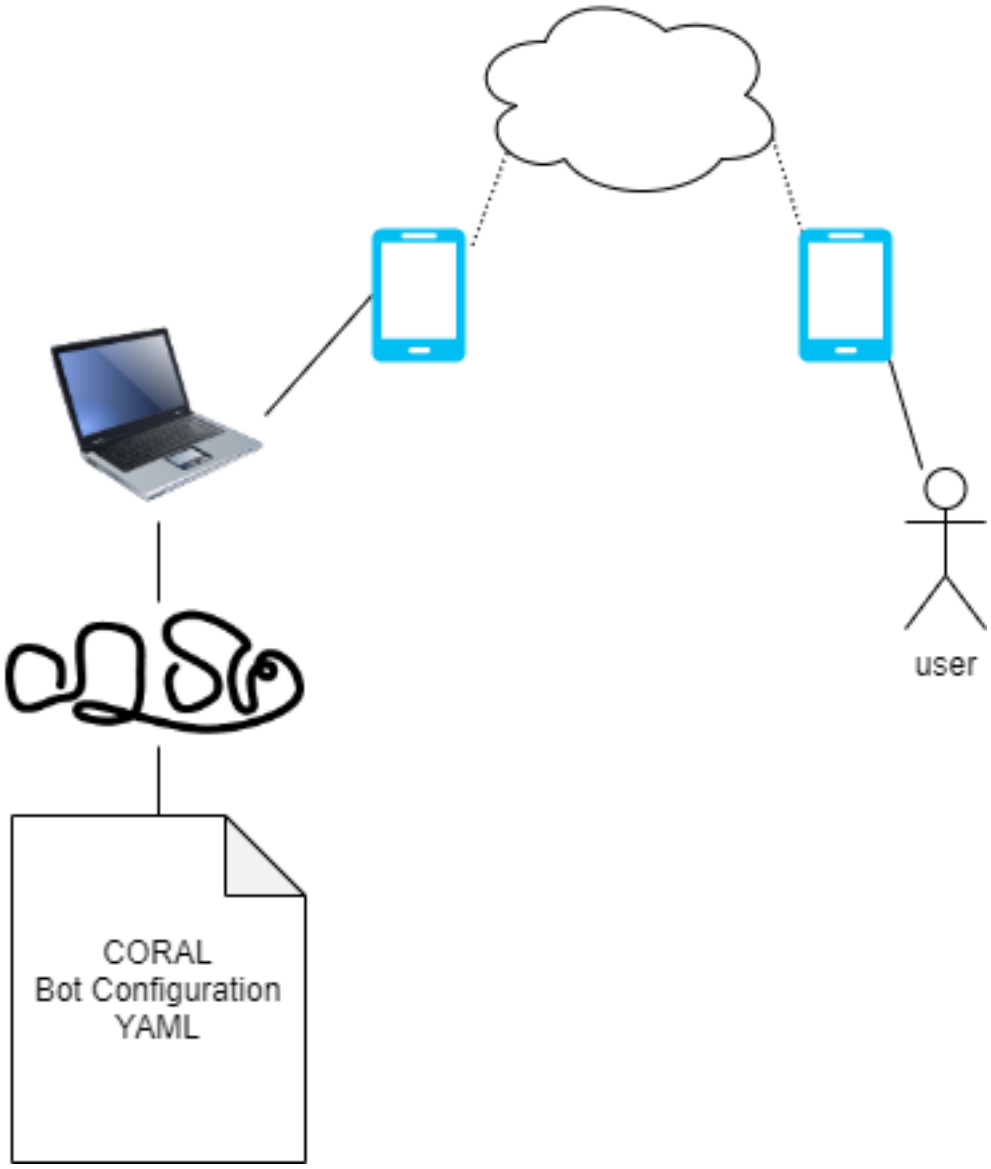
CHAPTER 1

How it Work

You need to test the NeMo is connected with your device and then you create the YAML file as the Bot Configuration and then test the bot by triggering the bot from the keyword registered at configuration.

NeMo is using the reverse engineered whatsapp web from here https://github.com/sigalor/whatsapp-web-reveng, this is the only reason why the requirement need the screen is stay awake and focus on whatsapp application, we are using https://github.com/Rhymen/go-whatsapp for the golang library

Nemo is also using the randomizer pause chat depends on how many word is typed, its like how typical human type the word 1-3 secon perword, also NeMo will sent the “typing” mode so the whatsapp will thinking that we are as the person


user
CORAL
Bot Configuration
YAML

CHAPTER 2

Requirements

1. android phone connected to pc/laptop
2. the android phone need to enable the settings “Stay awake” (screen will never sleep while charging), “USB Debugging”, “Install via USB”

   you need to enable this so you can connect your device with your pc/laptop
3. the screen stay on whatsapp appplication

   you need to do this so the nemo will not interupted, because the nemo is using the *web.whatsapp.com* API so the device need to be always connected to internet

CHAPTER 3

How to run

to run you need to have enviroment variable as in *.env.example*, rename it to *.env*, by default you need to create *coral* dir or you can specify by env var *CORAL_DIR*. The *coral* dir is for the bot configuration written in YAML format, to create a *coral* directory and put all the coral file in this folder, also you need the WhatsApp session file in order to connect to the current phone

3.1 Generate the whatsapp session

Use this tools to generate the whatsapp session file https://github.com/k1m0ch1/WhatsappLogin, you can download the latest binary file from latest release ttps://github.com/k1m0ch1/WhatsappLogin/releases/

and run with command *./WhatsappLogin -p 08123123123* and scan the QR

or

run with docker

```
docker run --rm -v sessions:/go/src/github.com/k1m0ch1/WhatsappLogin/sessions k1m0ch1/
↪whatsapplogin:latest -p 08123123123
```

3.2 Testing run the bot with example

after you have file *08123123123.gob* and *coral* directory with *basic.yml* inside, run the NeMo with this command

```
./NeMo 08123123123
```

this will find the file *08123123123.gob* in the *SESSION_DIR* environment variable by default this will goes to current directory running file

3.3 Run the nemo with docker

you must prepare the *coral* directory include with yaml file and Whatsapp file session, after that mount volume in docker with example command like this

```
docker run \
--name NeMo -v $(pwd)/coral:/app/coral \
-v $(pwd)/.sessions/08123123123.gob:/app/08123123123.gob \
k1m0ch1/nemo 08123123123
```

CHAPTER 4

Understanding Basic Coral Configuration

Coral as in the house of the clown fish, is the configuration of the bot in order to specific give the operation to NeMo, you can see the example from the *basic.yml* or *example.yml* file

4.1 Author

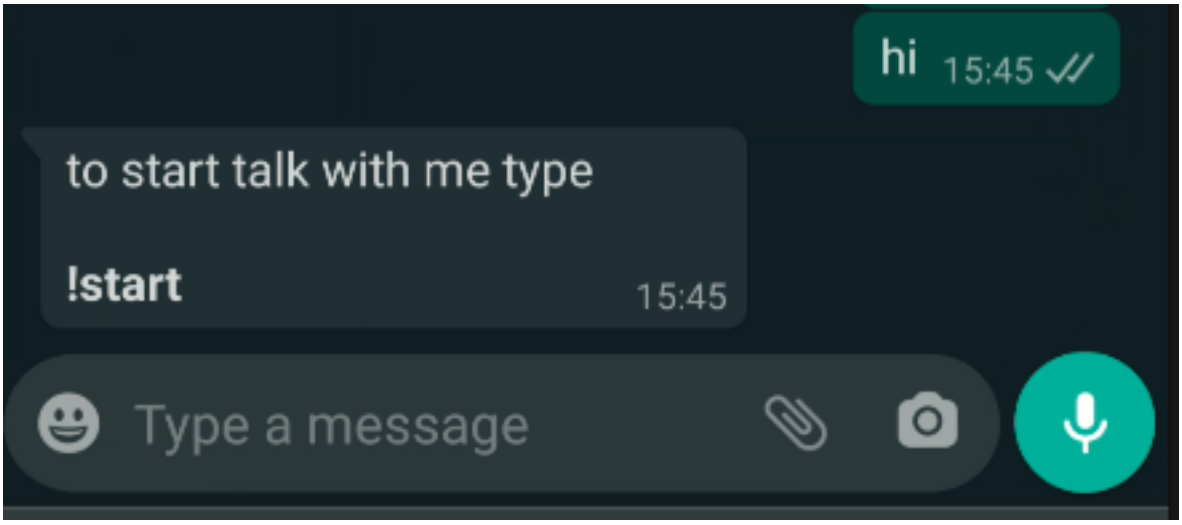
Specifically to define the author of the configuration

```
author:
    name: name
    phone: "08123123123"
    email: email@email.com
```

4.2 Default Greeting

This will be triggered when users chat any text to NeMo and start to send *message* and expected by the variable *expected_users*

```
default_greeting:
    message: "to start talk with me type\n\n*!start*"
```

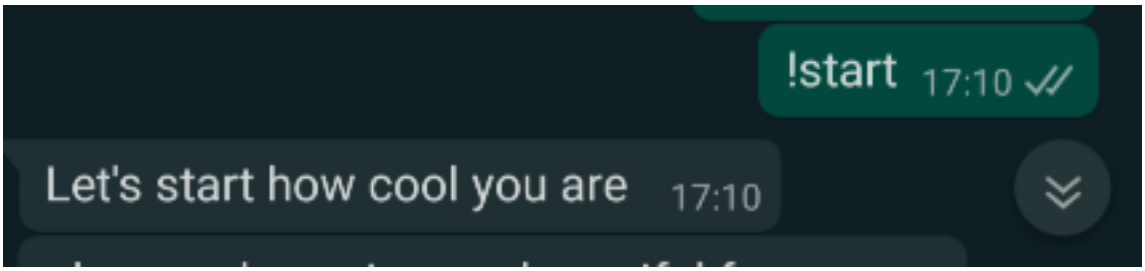


**note: default greeting will never sent to group

4.3 Commands

just remember if you want to record or in run process after triggering the commands, but the *True* value for *record* and *run_process*

```
commands:
    prefix: "!"
    command: "start"
    run_process: True
    message: "Let's start how cool you are"
```



4.4 Message Format Parser

you can use the message with the URL POST with format *{{URL}}* with POST method

the NeMo will handle this body Response

```
{
    "message": "response",
    "images": [
        {
        "URL": "link to image",
        "caption": "caption"
        }
```

(continues on next page)

(continued from previous page)

```
    ]
}
```

The *message* key will sent a normal response The *images* key will stored all the image information, and will send the image regarding the length of the array The *URL* key will parse the image from internet and sent this with a new response along side with the *caption*

also you can get the indexing of the parameter when user using the command and add some argument, for example user type this command

!start hello world this will trigger this command to having a argument, and whenever the message include the message indexing like *{{1}}* or *{{2}}* this will return the argument index, if the message with the format like this *everyone {{1}}, I love the {{2}}* this will return message *everyone hello, I love the world*

if the argument is using number you can use function *sum(argumentindex,argumentindex,...)* to calculate between number, for example the message following format *we have {{1}} {{2}} {{3}} and sum of {{1}} and {{3}} is {{sum(1,3)}}* with user type the command like this *!start 3 2 1* this will return *we have 3 2 1 and sum of 3 and 1 is 4*

for example if you type *!start hello* in the chat, the NeMo will sent the body like in the Webhook part and the *hello* will be inserted in the question part.

4.5 Process

This configuration define the question after commands triggered, for a

in order to use validation rule *image*, you need to specify *AWS S3* configuration in *.env* file

```
process:
  record: True
  log: False
  timeout: 300
  exit_command:
      prefix: "!"
      command: stop
      message: "alright I'll stop asking"
  end_message: "Hey its done, thank you"
  questions:
      - question:
          slug: first
          asking: please take a pic your beautiful face
          validation:
            rule: image
        message: "must a pic dude!"
      - question:
          slug: second
          asking: tell me some random number
          validation:
            rule: ^[0-9]*$
            message: you can't read that ? I ask you to write some number
```

if you set the key *record* to *True* you need to add the *Webhook* part

when you set the *log* key into True, you need to add *log* parent key

4.6 Webhook

In order to save all the input data, you can rely on webhook that built in this ChatBot

```
webhook:
  service: WEBHOOK
  url: https://url.com/webhook
```

the data will be sent with *POST* method to *url* key with body *JSON* like this

```
{
    "phone_number": "628123123123",
    "current_process": "basic",
    "current_question_slug": 1,
    "process_status": "DONE",
    "data": [
        {
            "slug": "first",
            "question": "please take a pic your beautiful face",
            "answer": "https=//public-tools.s3.ap-southeast-1.amazonaws.com/fresh/
↪nemo/645C4CC4141DB97B2A29BBE33725B1BE.jpeg",
            "created": "2020-09-25T13:54:56+07:00"
        },
        {
            "slug": "second",
            "question": "tell me some random number",
            "answer": "1200",
            "created": "2020-09-25T13:55:17+07:00"
        }
    ],
    "sent": "",
    "sent_to": "",
    "created": "2020-09-25T13:54:19+07:00",
    "expired": "2020-09-25T13:59:19+07:00",
    "finished": "2020-09-25T13:55:17+07:00"
}
```

4.7 Logging

In order to log every user input when the process start, you can rely on log that built in this ChatBot

```
webhook:
  service: WEBHOOK
  url: https://url.com/webhook
```

the data will be sent with *POST* method to *url* key with body *JSON* like above on webhook part

4.8 Expected Users

is a list of a phone number that expected by ChatBot, technically a whitelist that chatbot will hear, currenlt I made this for a specific phone number, because I'm not making this for "any" users, currently I want to made that, but I need more validation to implement that

```
expected_users:
  - 628123123123
  - 628321321312
```

you can set the *expected_users* to *any* so this will targeted the *any* user for *command* and *default_greeting* but the *schedules* will be disabled due to *any* users.

to activate bot in group you can just simply add the *phonenumber-groupid*

```
expected_users:
  - any
```