OSSとAPIで作る、AI”風”NWマネジメント/オペレーションツールとネットワーク自動化

Juniper Networks
有村淳矢
jarimura@juniper.net

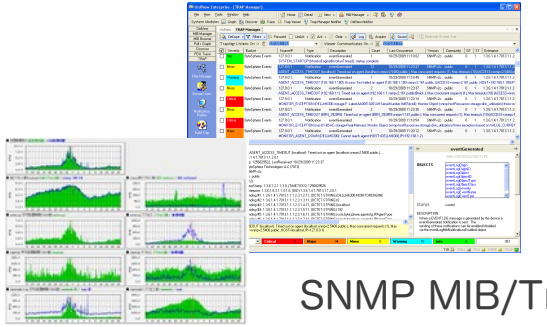
Feb-2018 JANOG 41 @広島

有村 淳矢
Juniper Networks株式会社
SPチーム所属 Sr.Systems Engineer

Backbone NW
Data Center Fabric
SDN/OverLay、仮想化基盤, NFV
自動化
SD-WAN
Packet Optical/DWDM
MPLS, PCEP, SR/SPRING, 等…

システム&ネットワークオペレーションのUI

パトランプ

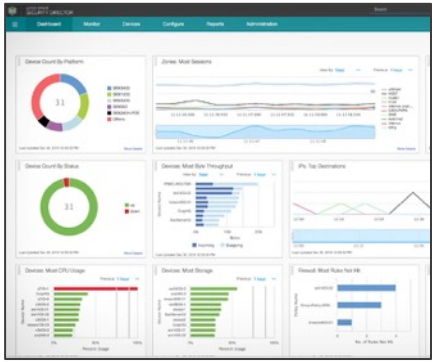


MRTG

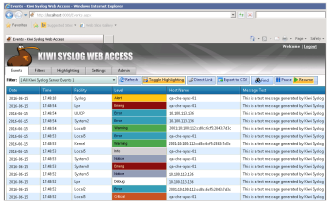
SNMP MIB/Trap

Terminal

専用ツール・WebUI



メール通知



Syslog

通知、状況把握
システム/ネットワーク可視化

コマンド、スクリプト等の実行

新しいユーザーインターフェース

slack

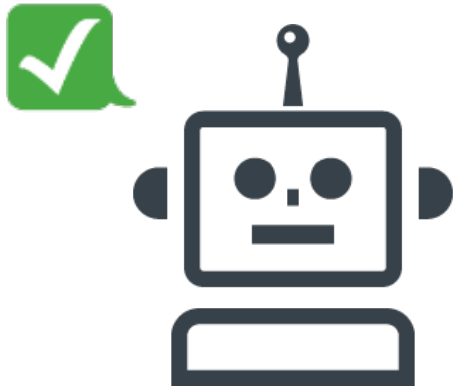
チャットツール

音声アシスタント

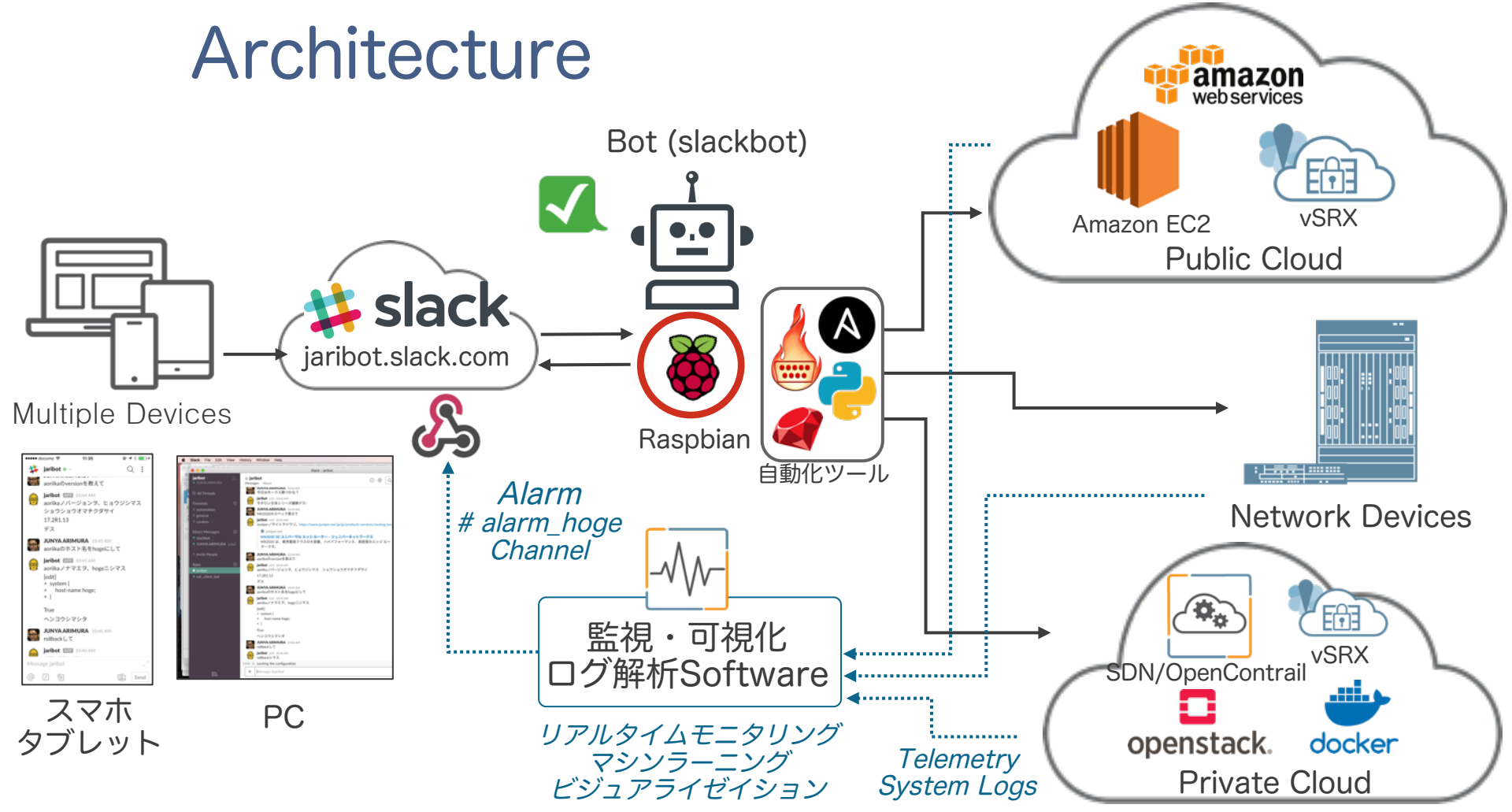
と、自動化ツールを使って、
”AI風な”ネットワークオペレーションをしてみよう！

Use Case ①

Botを使った、IC (Infrastructure as Code)の実現



Architecture



オペレータの皆さんや、エンドユーザーに新しいUXを提供

ルーターの設定を変更して！

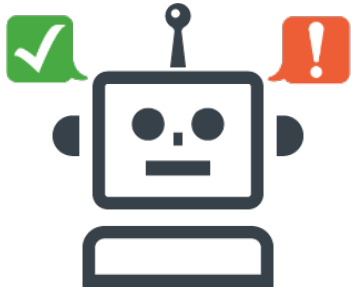
プライベートクラウド環境に仮装セキュリティデバイスを作って！

設定変更後のOSPFのトポロジー変わってないかな？

スイッチの筐体の温度を教えて？

AWS上にEC2インスタンスを作って！

マルチベンダー ネットワーク&サーバー
マルチプラットホームの管理



サーバーhoge上のvSRXのCPU使用率が高沸してます！

ログ解析より、セキュリティインシデントの可能性があります！

ルータhogeのInterfaceのトラフィックが80%を超えました！

サーバーのリソースが不足しそうです！

マルチデバイス

Slackbot

https://github.com/lins05/slackbot

- Features
  - Based on slack Real Time Messaging API
  - Simple plugins mechanism
  - Messages can be handled concurrently
  - Automatically reconnect to slack when connection is lost
  - Python3 Support
- Slackにもともと備わっているSlackbotとは異なる、GitHubで公開されているOSS
- PythonベースのChat Bot フレームワーク

@respond_to('hoge') もしくは @listen_to('hoge') で定義した文字列にヒットした場合、用意されたスクリプトを実行させる事で自動化ツールを動かし、message.reply('hoge') でその結果もしくはBotの発言をSlack UI に表示させる。

BotによるNetwork Operation

通常のBotとの会話
my_mentione.pyで定義した、ユーザーの発言に対して、決めれたコメントを返す

ユーザーの問いかけに対してコメントを返すスクリプト例

```
pi@raspberrypi:~/slackbot/plugins $ cat my_mention.py
from slackbot.bot import respond_to

@respond_to('こんにちは')
def cheer(message):
    message.reply('こんにちは！何かご用ですか？')


@respond_to('MX2020のスペック')
def cheer(message):
    message.reply('Juniperのサイトをどうぞ。https://www.juniper.net/jp/jp/products-services/routing/mx-series/mx2020/')


pi@raspberrypi:~/slackbot/plugins $
```

'こんにちは' というキーワードが来たら、def output(message) 以下を実行し message.reply()をテキストとして表示

BotによるNetwork Operation – cont

自動化ツールを使ったオペレーション
実行トリガーとなるキーワードとアクションを定義し、自動化ツール・スクリプトを実行させる

ansible_aoriika_version_grep.py
‘version’というキーワードに対し、
‘ansible-playbook /home/pi/ansible/version.yml’ を実行

python_aoriika_interface_change_xe-0-0-0.py
‘xe-0/0/0’というキーワードに対し、
‘python /home/pi/PyEZ/python/junos-pyez-config.py’ を実行

python_aoriika_rollback.py
‘rollback’というキーワードに対し、
‘python /home/pi/PyEZ/python/junos-pyez-config-rollback.py’ を実行

入力‘version’に対してansible-playbookを実行するスクリプト例

```
pi@raspberrypi:~/slackbot/plugins $ cat ansible_aoriika_version_grep.py
from slackbot.bot import respond_to
from slackbot.bot import listen_to
import subprocess
import re
import sys

@respond_to('version')
@listen_to('version')
def output(message):
    message.reply('aoriikaのJUNOS versionを表示します。')
    hoge = subprocess.run(['ansible-playbook' , '/home/pi/ansible/version.yml'], stdout=subprocess.PIPE)

        r = re.compile('\"([a-zA-Z1-9.]+)\"')
    for x in hoge.stdout.splitlines():
      if x.find(b'msg') > 0:
        line = x

    line = r.findall(str(line))[1]

    message.reply(line)
    message.reply('です。')
```

’version’ というキーワードが来たら、def output(message) 以下を実行

message.reply()の文字列を表示させたのち、
subprocessモジュールで、’ansible-playbook /home/pi/ansible/version.yml’を実行させる

ansible-playbokの実行結果がyaml形式で返ってくるので、
バージョン情報だけを切り抜き、’line’に格納

message.reply()で、lineに格納しているバージョン情報を表示させる

装置のOS Versionを取得する.ymlファイル例

version.yml ファイル

```
pi@raspberrypi:~/ansible $ cat version.yml
- name: Get Device Facts
  hosts: all
  roles:
   - Juniper.junos
  connection: local
  gather_facts: no

  tasks:
  - name: Checking NETCONF connectivity
    wait_for: host={{ inventory_hostname }} port=830 timeout=5

  - name: Retrieve information from devices running Junos OS
    junos_get_facts:
      host={{ inventory_hostname }}
    register: junos

  - name: version
    debug: msg="{{ junos.facts.version }}"

pi@raspberrypi:
```

その実行結果

```
pi@raspberrypi:~/ansible $ ansible-playbook version.yml

PLAY [Get Device Facts] ********************************************************

TASK [Checking NETCONF connectivity] *******************************************
ok: [172.27.114.7]

TASK [Retrieve information from devices running Junos OS] **********
ok: [172.27.114.7]
```

OS Version情報の表示

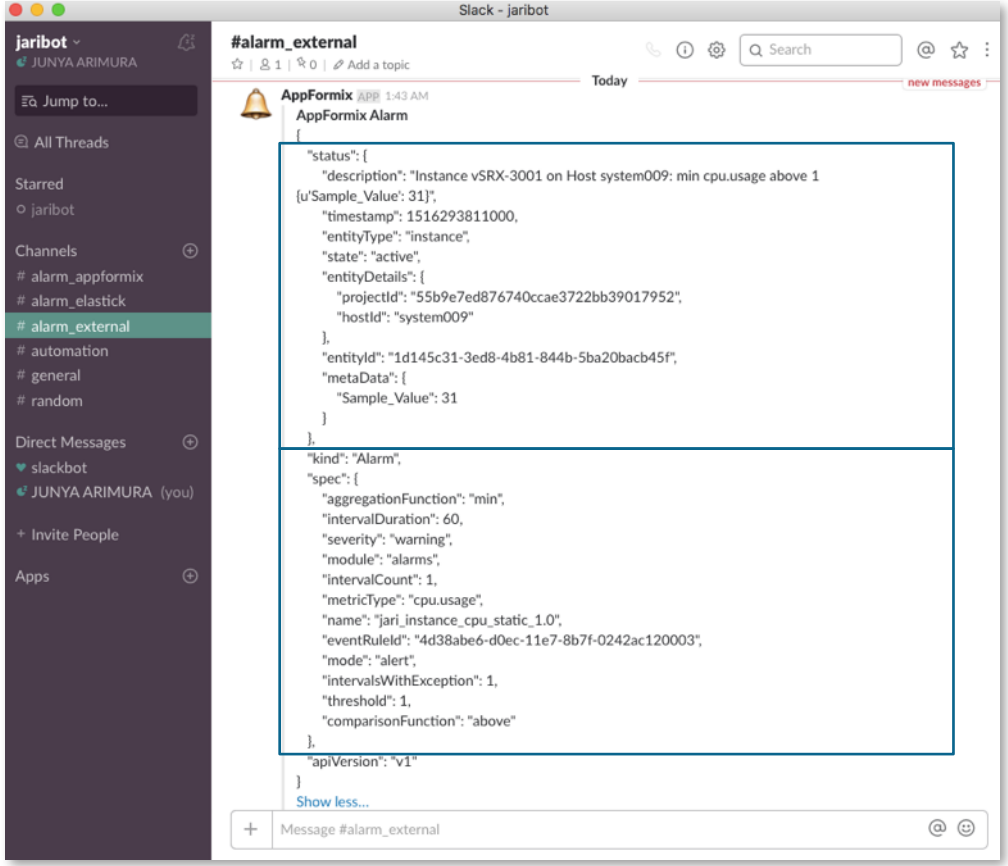
```
TASK [version] *****************************************************************
ok: [172.27.114.7] => {
    "msg": "17.2R1.13"
}

PLAY RECAP *********************************************************************
172.27.114.7               :
ok=3    changed=0    unreachable=0    failed=0

pi@raspberrypi:~/ansible $
```

Incoming WebHooksを使ったAlarm監視

https://api.slack.com/incoming-webhooks

- Send data into Slack in real-time.
- 外部ツールから、JSON形式のアラームをリアルタイムで受信
- マシンラーニング、振る舞い検知を備えたソフトウェア上で、監視項目に閾値を設定し、想定外の挙動があった場合、アラームをSlack Channelに送信
- 最近は様々なOSSやベンダー製品が Incoming WebHooksに対応
- マシンラーニング、振る舞い検知ということで、、、ちょっとだけ、AIっぽく

受信アラーム例:
Openstack上の仮想FirewallのCPU閾値超えのアラーム

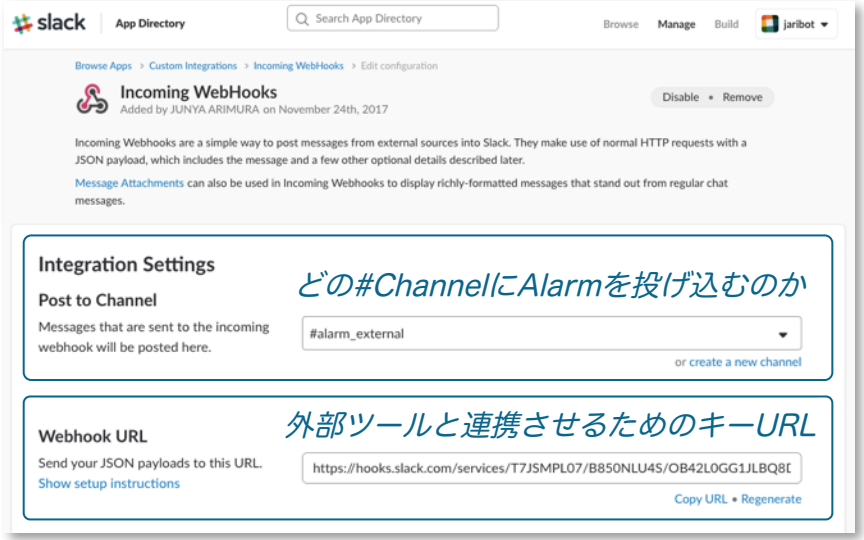


Host名:system009 サーバー上の vSRX-3001という仮想マシンで cpu.usageの値が高くなったというアラーム

cpu.usage 値の定義情報

Incoming Webhookと外部ツールとの連携例

jaribot.slack.comでの
Incoming WebHook設定画面



elastic

elastic stackでの設定例
yamlファイルにキーURLを記載し、Alarmを投げ込む

```
watcher.actions.slack.service:
    account:
      monitoring:                          キーURLの設定
        url:https://hooks.slack.com/services/T0A6BLEEA/B0A6D1PRD/76nxxxxxxxxxx
        message_defaults:
            from: Watcher
```

https://www.elastic.co/guide/en/x-pack/current/actions-slack.html

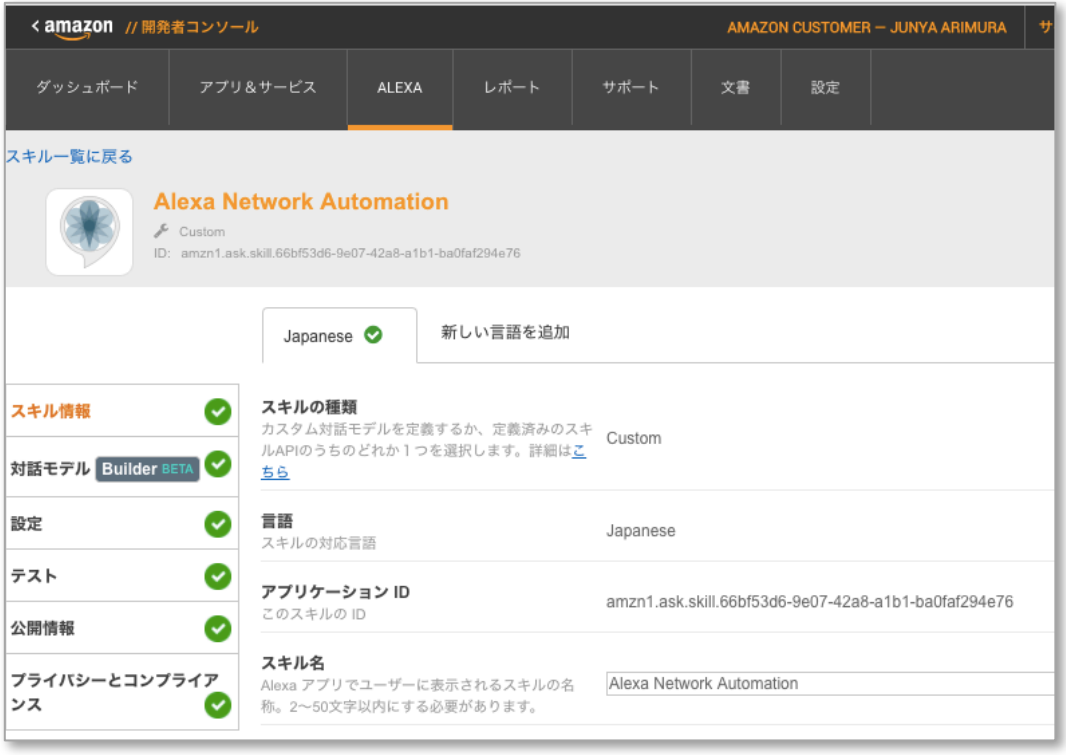
WebUIからキーURLを設定するツールも

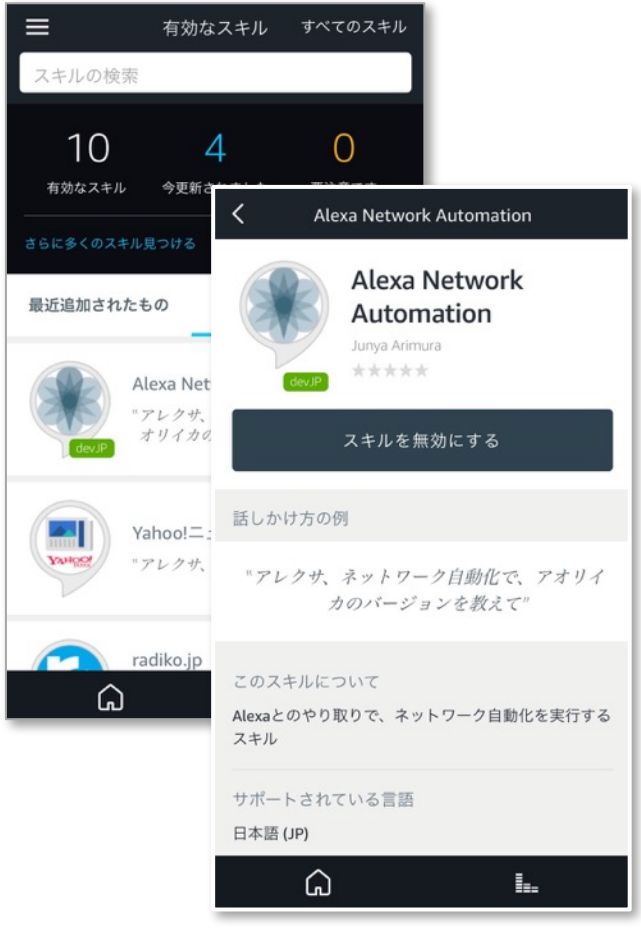
Use Case ②

音声アシスタントを使った自動化の実現

BotによるNetwork Operation

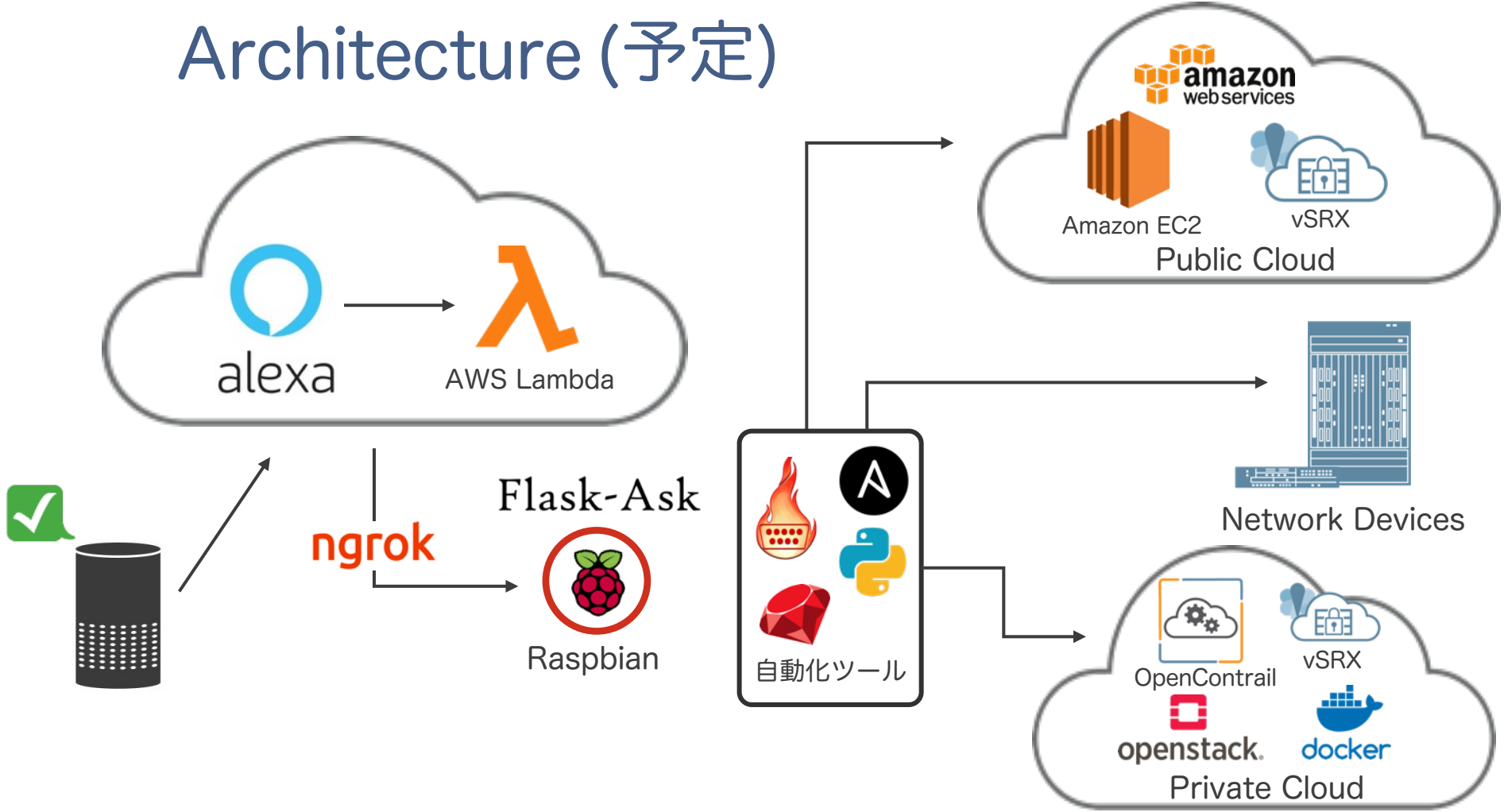
Alexa Skill Kitにてスキルを作成





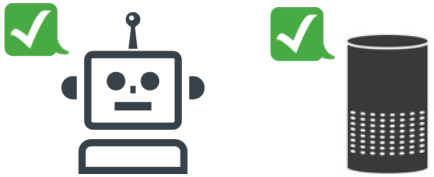
Alexa アプリにてスキルを有効化

Architecture (予定)



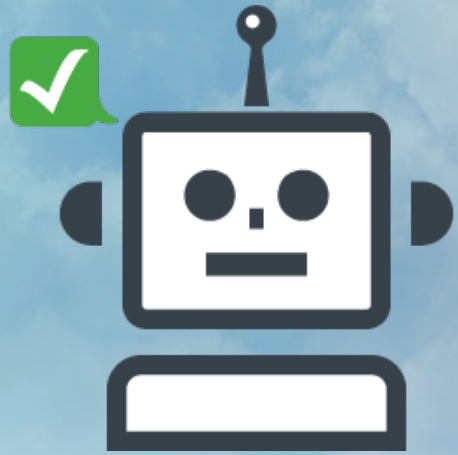
まとめ

- 新しいユーザーインターフェースと自動化ツールを組み合わせることで、オペレーター、エンドユーザーに新しいUser Experienceを提供
- マルチデバイスでの入力、マルチプラットホーム、マルチベンダーに対応
- 入力キーワードに対して、アクションと実行スクリプトを実行することでIC (Infrastructure as Code)を実現
- あくまでも、AI”風”



Special Thanks!!
Motoshi Nakagawa - monaka@juniper.net
Kazuki Shimizu - kshimizu@juniper.net

Appendix: AI”風” Automation Botの作り方



使用したハードウェア、OS Version

■ ハードウェア

RSPBERRY PI3 MODEL B

https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

•Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
•1GB RAM
•BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
•40-pin extended GPIO
•4 USB 2 ports
•4 Pole stereo output and composite video port
•Full size HDMI
•CSI camera port for connecting a Raspberry Pi camera
•DSI display port for connecting a Raspberry Pi touchscreen display
•Micro SD port for loading your operating system and storing data
•Upgraded switched Micro USB power source up to 2.5A

■ OS Version

Raspbian 9.1
Debianベースのオペレーティングシステム

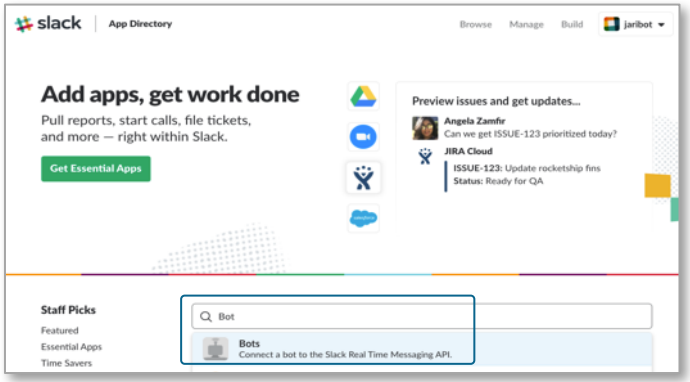
Raspbian ダウンロードはこちら https://www.raspberrypi.org/downloads/raspbian/

※ Raspberry Piである必要はありません。馴染みのオペレーティングシステムを使用してください

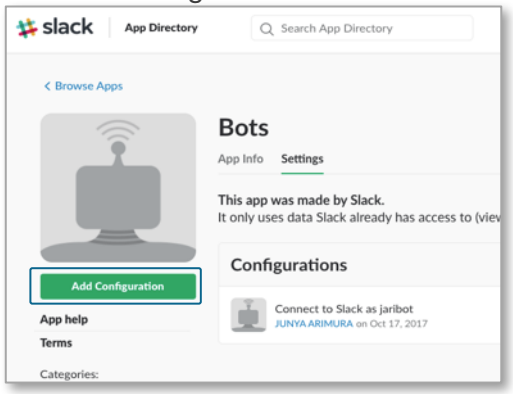
Botの作成

■ Botの作成

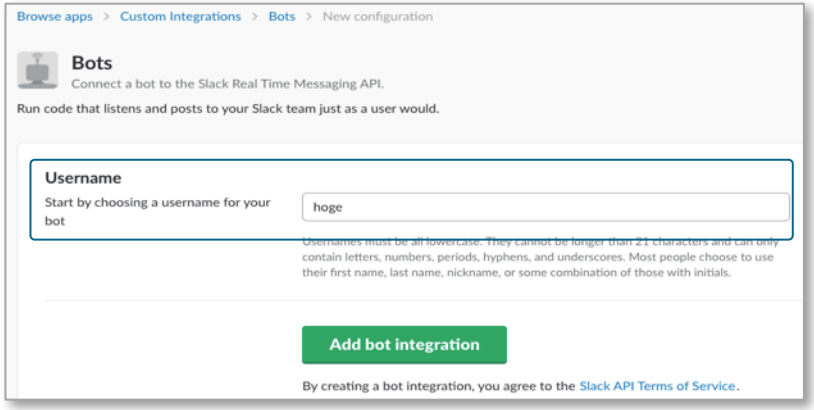
1. <workspace>.slack.jp で、App Directoryのページで'Bot'と検索



2. 'Add Configuration'を選択



3. 好きなBotの名前を決め、'Add bot integration'でBotを作成



slackbotのインストールとSlackとの連携

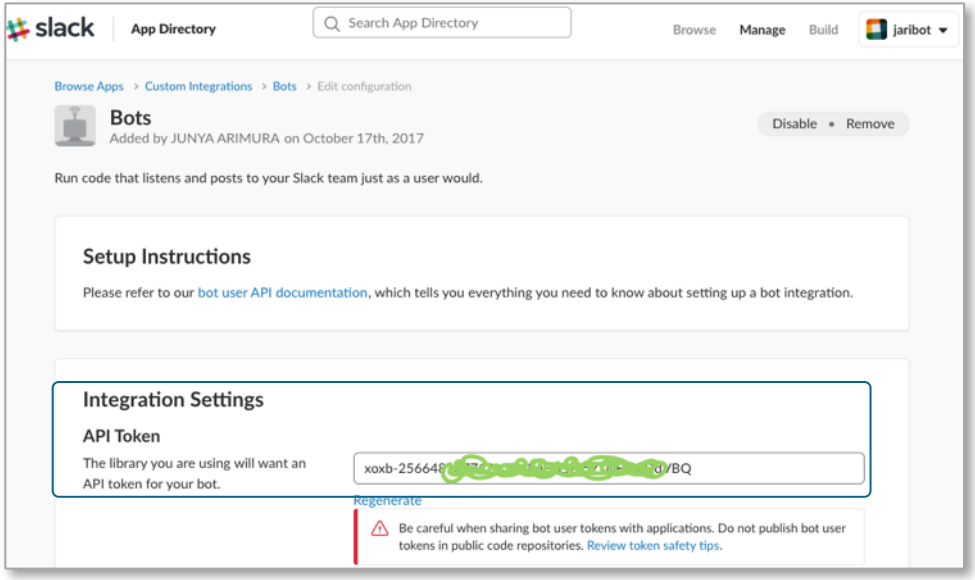
参照：https://github.com/lins05/slackbot

■ slackbotのインストール

```
sudo apt-get install python3-pip
sudo pip3 install slackbot
```

参考にさせていただいたサイト： PythonのslackbotライブラリでSlackボットを作る (https://qiita.com/sukesuke/items/1ac92251def87357fdf6)

■ <workspace>.slack.com とslackbotの連携



xoxbから始まるAPI Tokenをもとに、Slackワークスペースとslackbotが API連携するので、このToken キーをコピー

slackbotのインストールとSlackとの連携 - cont

■ slackbotに取得したTokenキーの登録

slackbot_settings.py に、取得したキーを登録する

```
pi@raspberrypi:~/slackbot $ cat slackbot_settings.py
# -*- coding: utf-8 -*-
import os

API_TOKEN = "xoxb-25664xxxxxxxxxxxxxxxxxxxxxxxxBQ”

default_reply = "おっしゃっている事がわかりません。"
PLUGINS = [ 'plugins' ]

pi@raspberrypi:~/slackbot $
```

API Token

併せて、Defaultの返事(PLUGINSサブフォルダ配下のpythonスクリプト中の、キーワードにヒットしなかった場合の返事)を設定

PLUGINS = []で、python スクリプトを置いてあるサブフォルダを設定

キーワードと、応答コメントの設定

■ PLUGINS サブフォルダ配下のPythonスクリプトの設定例

my_mention.py に、やり取りしたい、キーワードと応答コメントを設定

```
pi@raspberrypi:~/slackbot/plugins $ cat my_mention.py
from slackbot.bot import respond_to

@respond_to('こんにちは')
def cheer(message):
    message.reply('こんにちは！何かご用ですか？')

@respond_to('MX2020')
def cheer(message):
    message.reply('Juniperのサイトをどうぞ。https://www.juniper.net/jp/jp/products-services/routing/mx-series/mx2020/')

pi@raspberrypi:~/slackbot/plugins $
```

slackbotの実行

■ slackbotの実行

```
pi@raspberrypi:~/slackbot $ python3 run.py
start slackbot
```

■ 動作確認 – スマホでの例

Bot名横のアイコンがグリーンに変われば<workspace>.slack.comとの連携が完了です

my_mention.py に登録されたキーワードに対してコメントを返します

慰めてもくれます

次スライド以降で、入力キーワードに対して、特定スクリプトを実行するpythonスクリプト例を記載します

特定スクリプトを実行するpythonスクリプト例
インターフェースの設定を変更するパターン

※ 本スクリプトはJUNOSの自動化ツールを前提に作成しています (申し訳ありません！！)
※ このスクリプトを試す場合は、あらかじめ、PyEZ (Python Library for JUNOS Automation)をインストールしてください
Python library for JUNOS Automation
https://github.com/Juniper/py-junos-eznc
※ もしくは、皆さんがすでにお使いのAutomation Toolsを起動するコマンドを、'subprocess' に記載してください

■ slackbot pythonスクリプト

python_aoriika_interface_change_xe-0-0-0.py で、'python /home/pi/PyEZ/python/junos-pyez-config.py' を実行

```
pi@raspberrypi:~/slackbot/plugins $ cat python_aoriika_interface_change_xe-0-0-0.py
from slackbot.bot import respond_to
from slackbot.bot import listen_to
import subprocess

@respond_to('xe-0/0/0')
@listen_to('xe-0/0/0')
def output(message):
    message.reply('aoriikaのinterface xe-0/0/0に、IPアドレスを設定します。')
    hoge = subprocess.run(['python' , '/home/pi/PyEZ/python/junos-pyez-config.py'], stdout=subprocess.PIPE)

    message.reply(hoge.stdout)

    message.reply('interface xe-0/0/0の設定を完了しました。')

pi@raspberrypi:~/slackbot/plugins $
```

特定スクリプトを実行するpythonスクリプト例
インターフェースの設定を変更するパターン

■ JUNOS PyEZ pythonスクリプト

```
pi@raspberrypi:~/PyEZ/python $ cat junos-pyez-config.py
from jnpr.junos import Device
from jnpr.junos.utils.config import Config
from jnpr.junos.exception import ConnectError
from jnpr.junos.exception import LockError
from jnpr.junos.exception import UnlockError
from jnpr.junos.exception import ConfigLoadError
from jnpr.junos.exception import CommitError

host = '172.27.114.7'
conf_file = '/home/pi/PyEZ/config/xe-0-0-0.conf'

def main():
    # open a connection with the device and start a NETCONF
session
    try:
        dev = Device(host=host)
        dev.open()
    except ConnectError as err:
        print ("Cannot connect to device: {0}".format(err))
        return

    dev.bind(cu=Config)

    # Lock the configuration, load configuration changes, and
commit
    print ("Locking the configuration")
    try:
        dev.cu.lock()
    except LockError as err:
        print ("Unable to lock configuration: {0}".format(err))
        dev.close()
        return

    print ("Loading configuration changes")
    try:
        dev.cu.load(path=conf_file, merge=True)
    except (ConfigLoadError, Exception) as err:
        print ("Unable to load configuration changes:
{0}".format(err))
        print ("Unlocking the configuration")
        try:
                dev.cu.unlock()
        except UnlockError:
            print ("Unable to unlock configuration: {0}".format(err))
        dev.close()
        return

    print ("Committing the configuration")
    try:
        dev.cu.commit(comment='Loaded by example.')
    except CommitError as err:
        print ("Unable to commit configuration: {0}".format(err))
        print ("Unlocking the configuration")
        try:
            dev.cu.unlock()
        except UnlockError as err:
            print ("Unable to unlock configuration: {0}".format(err))
        dev.close()
        return

    print ("Unlocking the configuration")
    try:
        dev.cu.unlock()
    except UnlockError as err:
        print ("Unable to unlock configuration: {0}".format(err))


    # End the NETCONF session and close the connection
    dev.close()

if __name__ == "__main__":
    main()
pi@raspberrypi:~/PyEZ/python $
```

■ 設定するためのconfiguration file

```
pi@raspberrypi:~/PyEZ/config $ cat xe-0-0-0.conf
interfaces {
    xe-0/0/0 {
        unit 0 {
            family inet {
                address 192.168.100.1/30;
            }
            family iso;
            family mpls {
                maximum-labels 5;
            }
        }
    }
}
pi@raspberrypi:~/PyEZ/config
```

特定スクリプトを実行するpythonスクリプト例
実行例

■ 作業前の interface xe-0/0/0 の設定状況

```
{master}[edit]
lab@aoriika-re0# run show interfaces terse xe-0/0/0
Interface               Admin Link Proto    Local                 Remote
xe-0/0/0                up    up
```

Interface xe-0/0/0のIPアドレス未設定

```
{master}[edit]
lab@aoriika-re0# show interfaces xe-0/0/0
```

Interface xe-0/0/0設定なし

```
{master}[edit]
lab@aoriika-re0#
```

■ Botにxe-0/0/0の設定変更を依頼

特定スクリプトを実行するpythonスクリプト例
実行例 - cont

■ interface xe-0/0/0 の設定状況の再確認

```
{master}[edit]
lab@aoriika-re0# run show interfaces terse xe-0/0/0
Interface               Admin Link Proto    Local                 Remote
xe-0/0/0                up    up
xe-0/0/0.0              up    up   inet     192.168.100.1/30
                                  iso
                                  mpls
                                  multiservice
```

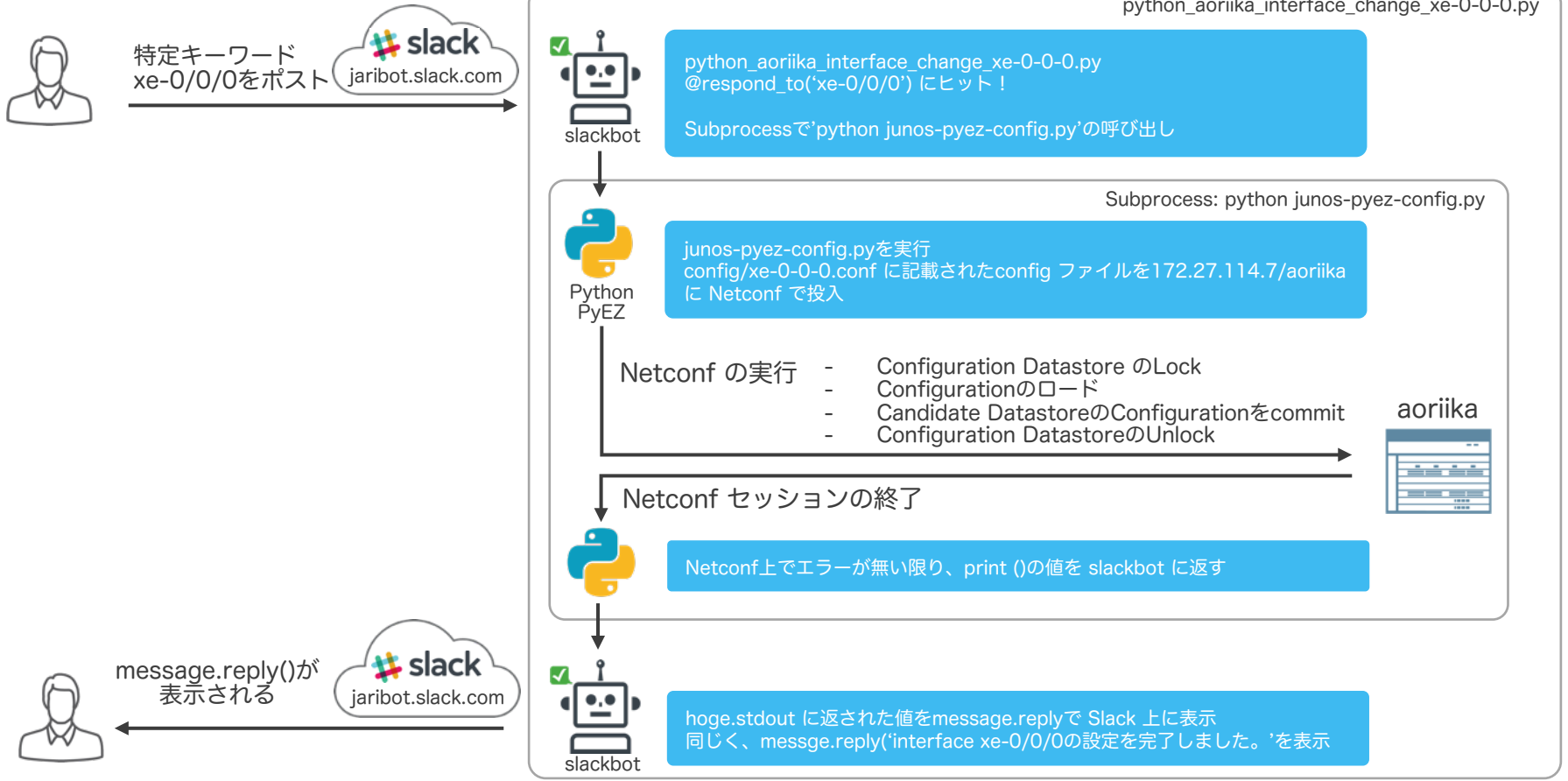
Interface xe-0/0/0に設定がされ、動作していることを確認

```
{master}[edit]
lab@aoriika-re0# show interfaces xe-0/0/0
unit 0 {
    family inet {
        address 192.168.100.1/30;
    }
    family iso;
    family mpls {
        maximum-labels 5;
    }
}
```

Interface xe-0/0/0設定ファイルを確認

```
{master}[edit]
lab@aoriika-re0#
```

特定スクリプトを実行するpythonスクリプト例
動作イメージ




Thank you.