

# 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](mailto: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...



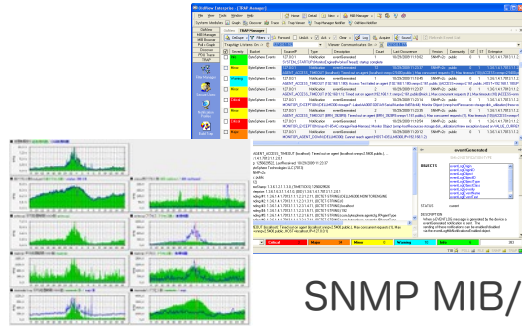
# システム&ネットワークオペレーションのUI



パトランプ

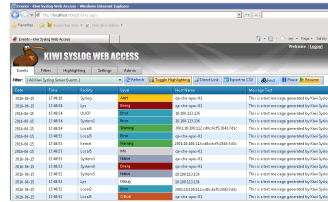


メール通知



MRTG

SNMP MIB/Trap

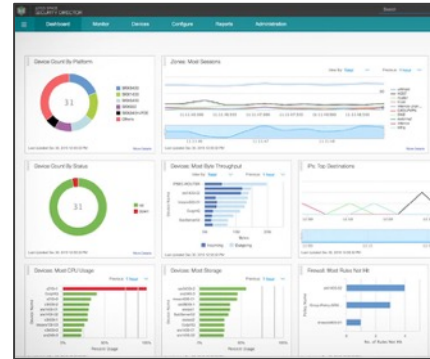


Syslog

Terminal



専用ツール・WebUI



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



チャットツール

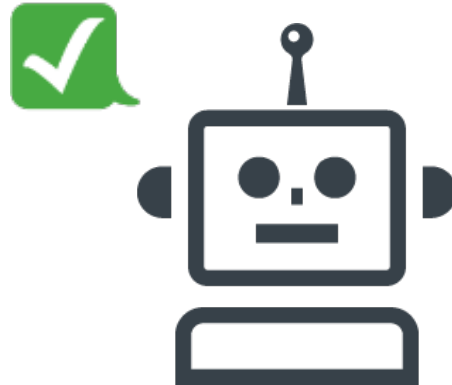


音声アシスタント

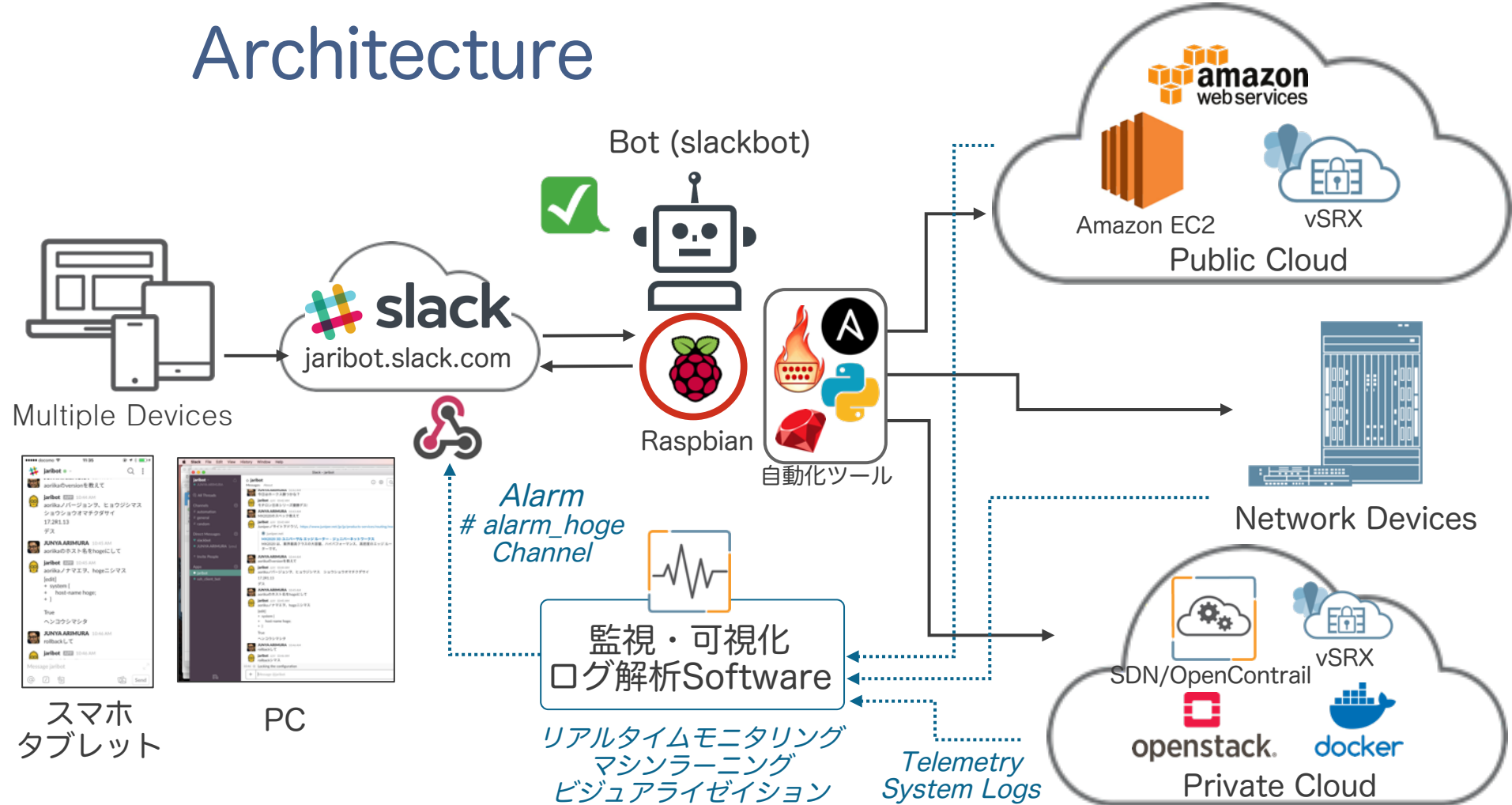
と、自動化ツールを使って、  
”AI風な”ネットワークオペレーションをしてみよう！

# Use Case ①

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



# Architecture



スマホ  
タブレット

PC

リアルタイムモニタリング  
マシンラーニング  
ビジュアライゼーション

Telemetry  
System Logs

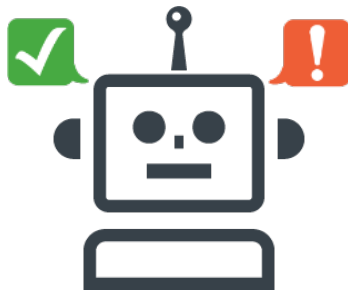
# オペレータの皆さんや、エンドユーザーに 新しいUXを提供

ルーターの設定を変更して！

マルチベンダー ネットワーク&サーバー  
マルチプラットフォームの管理

サーバーhoge上の  
vSRXのCPU使用率が  
高沸してます！

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！



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！

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？

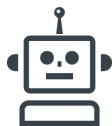
ルータhogeのInterfaceの  
トラフィックが80%を  
超えました！

AWS上にEC2  
インスタンスを作って！

サーバーのリソースが  
不足そうです！



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、決められたコメントを返す



The screenshot shows a chat interface for a bot named 'jaribot'. At the top, there is a header with the bot's name 'jaribot' (with a star icon), 'Messages', and 'About'. To the right are icons for information (i), settings (gear), and a search bar. The chat history is dated 'Yesterday'.

The conversation consists of four messages:

- JUNYA ARIMURA** (6:26 PM): こんにちは
- jaribot** (APP) (6:26 PM): こんにちは！何かご用ですか？
- JUNYA ARIMURA** (6:27 PM): MX2020のスペックを教えてください。
- jaribot** (APP) (6:27 PM): Juniperのサイトをどうぞ。 <https://www.juniper.net/jp/jp/products-services/routing/mx-series/mx2020/>

Below the last message, there is a link preview for 'juniper.net' with the title 'MX2020 3D ユニバーサル エッジ ルーター - ジュニパーネットワークス'. The preview text reads: 'MX2020 は、業界最高クラスの大容量、ハイパフォーマンス、高密度のエッジ ルーターです。'

# 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？')
```

*'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*

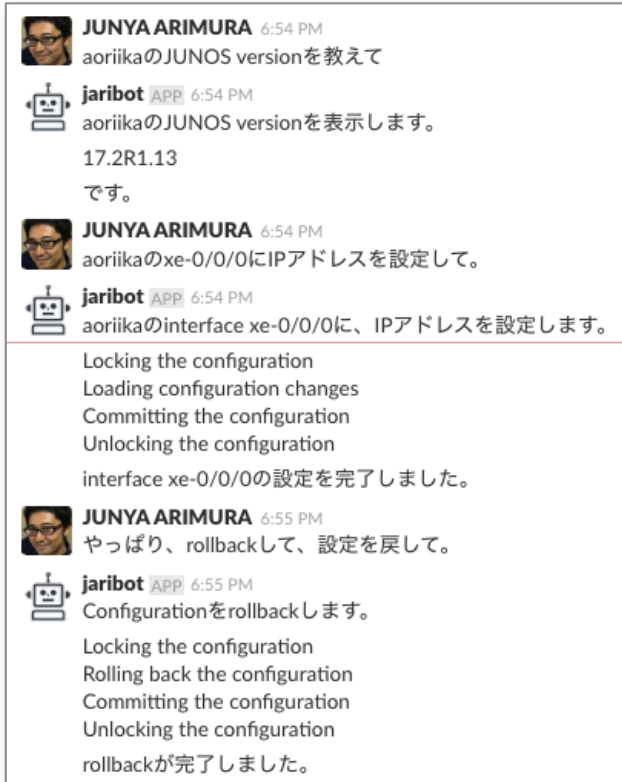
```
@respond_to('MX2020のスペック')
def cheer(message):
    message.reply('Juniperのサイトをどうぞ。https://www.juniper.net/jp/jp/products-services/routing/mx-series/mx2020/')
```

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

# 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



**JUNYA ARIMURA** 6:54 PM  
aoriikaのJUNOS versionを教えて

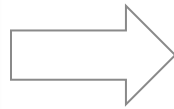
**jaribot** APP 6:54 PM  
aoriikaのJUNOS versionを表示します。  
17.2R1.13  
です。

**JUNYA ARIMURA** 6:54 PM  
aoriikaのxe-0/0/0にIPアドレスを設定して。

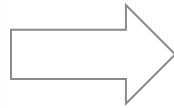
**jaribot** APP 6:54 PM  
aoriikaのinterface xe-0/0/0に、IPアドレスを設定します。  
Locking the configuration  
Loading configuration changes  
Committing the configuration  
Unlocking the configuration  
interface xe-0/0/0の設定を完了しました。

**JUNYA ARIMURA** 6:55 PM  
やっぱり、rollbackして、設定を戻して。

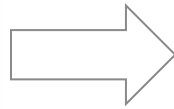
**jaribot** APP 6:55 PM  
Configurationをrollbackします。  
Locking the configuration  
Rolling back the configuration  
Committing the configuration  
Unlocking the configuration  
rollbackが完了しました。



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_ioriika_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):
```

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

```
message.reply('ioriikaのJUNOS versionを表示します。')
hoge = subprocess.run(['ansible-playbook', '/home/pi/ansible/version.yml'], stdout=subprocess.PIPE)
```

*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*

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

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

*ansible-playbook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(line)
message.reply('です。')
```

*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 $
```

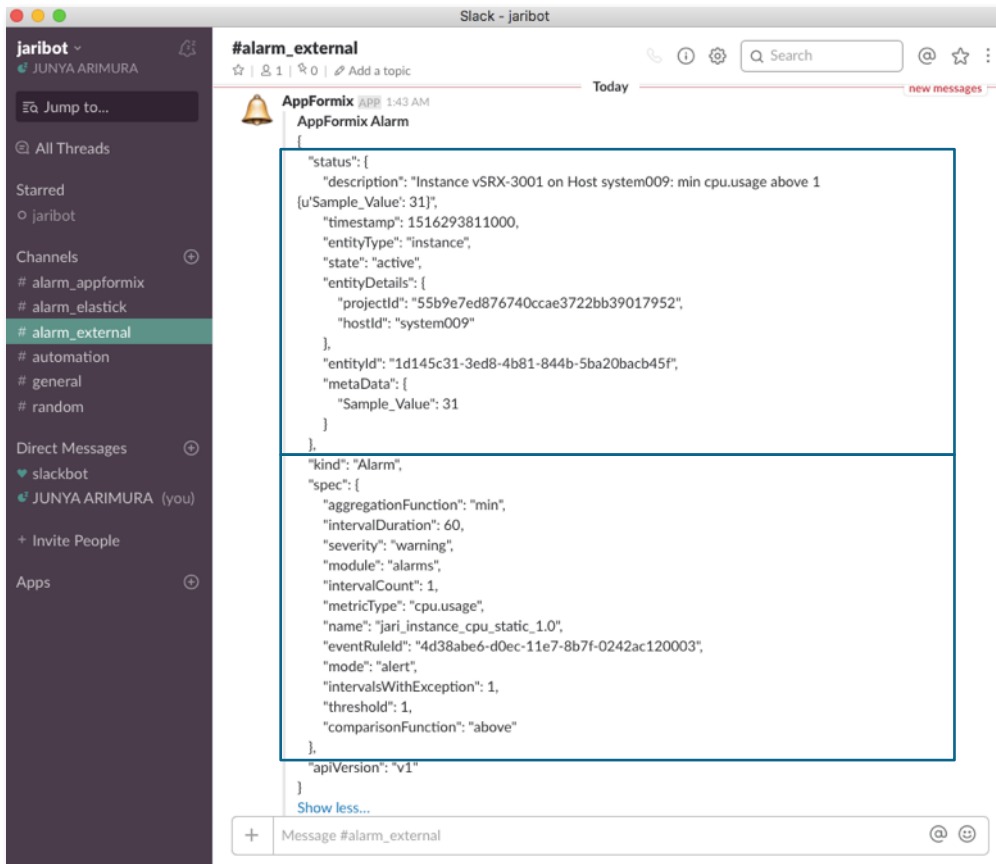


# Incoming WebHooksを使ったAlarm監視

<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



Slack - jaribot

#alarm\_external

AppFormix APP 1:43 AM

AppFormix Alarm

```
{
  "status": {
    "description": "Instance vSRX-3001 on Host system009: min cpu.usage above 1
    {u'Sample_Value': 31}",
    "timestamp": 1516293811000,
    "entityType": "instance",
    "state": "active",
    "entityDetails": {
      "projectId": "55b9e7ed876740ccae3722bb39017952",
      "hostId": "system009"
    },
    "entityId": "1d145c31-3ed8-4b81-844b-5ba20bacb45f",
    "metaData": {
      "Sample_Value": 31
    }
  },
  "kind": "Alarm",
  "spec": {
    "aggregationFunction": "min",
    "intervalDuration": 60,
    "severity": "warning",
    "module": "alarms",
    "intervalCount": 1,
    "metricType": "cpu.usage",
    "name": "jari_instance_cpu_static_1.0",
    "eventRuleId": "4d38abe6-d0ec-11e7-8b7f-0242ac120003",
    "mode": "alert",
    "intervalsWithException": 1,
    "threshold": 1,
    "comparisonFunction": "above"
  },
  "apiVersion": "v1"
}
```

Show less...

+ Message #alarm\_external

*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

slack App Directory Search App Directory Browse Manage Build jaribot

Browse Apps > Custom Integrations > Incoming Webhooks > Edit configuration

### Incoming WebHooks

Added by JUNYA ARIMURA on November 24th, 2017 Disable Remove

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 described later.

[Message Attachments](#) can also be used in Incoming Webhooks to display richly-formatted messages that stand out from regular chat messages.

#### Integration Settings

**Post to Channel** *どの#ChannelにAlarmを投げ込むのか*

Messages that are sent to the incoming webhook will be posted here.  or create a new channel

**Webhook URL** *外部ツールと連携させるためのキーURL*

Send your JSON payloads to this URL.  [Show setup instructions](#) [Copy URL](#) [Regenerate](#)



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/76nxxxxxxxxxxx

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

### Notification Settings

PagerDuty

Service Now

Slack

Notification Services

Slack Webhook Name	Webhook URL	Edit	🗑️
jaribot	https://hooks.slack.com/services/T7JSMPL07/B850...	✏️	🗑️
appformix_error	https://hooks.slack.com/services/T7JSMPL07/B863...	✏️	🗑️

*キーURLの設定*

+ Add Service

Directions:



## 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

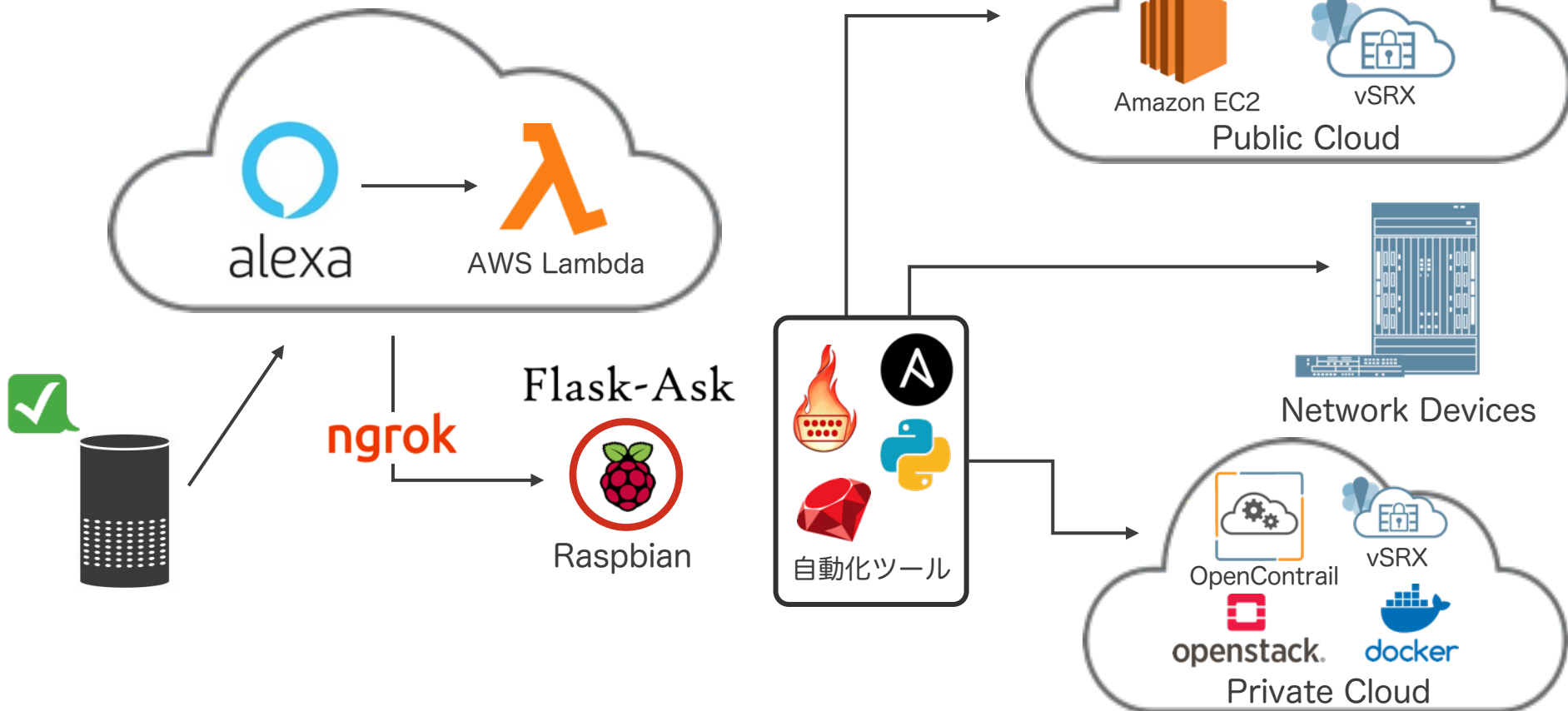
The screenshot shows the Amazon Developer Console for the 'Alexa Network Automation' skill. The skill is currently in 'Custom' mode and supports Japanese. The configuration table is as follows:

項目	設定
スキルの種類	Custom
言語	Japanese
アプリケーション ID	amzn1.ask.skill.66bf53d6-9e07-42a8-a1b1-ba0faf294e76
スキル名	Alexa Network Automation

The screenshot shows the Alexa mobile app interface. At the top, there are statistics: 10有効なスキル (Active Skills), 4今更新されたスキル (Skills updated today), and 0未更新のスキル (Skills not updated). Below this is a search bar and a list of recently added skills. The 'Alexa Network Automation' skill is highlighted, showing its icon, name, developer 'Junya Arimura', and a 5-star rating. A 'スキルを無効にする' (Disable Skill) button is visible. Below the skill card, there is a section for '話しかけ方の例' (Example of how to talk) with the text: "アレクサ、ネットワーク自動化で、アオリイカのバージョンを教えてください" (Alexa, please tell me the version of the network automation with the octopus). Another section titled 'このスキルについて' (About this skill) states: "Alexaとのやり取りで、ネットワーク自動化を実行するスキル" (A skill that executes network automation through interaction with Alexa). At the bottom, it lists supported languages: サポートされている言語 (Supported languages) and 日本語 (JP) (Japanese).

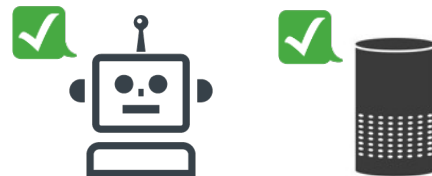
## Alexa アプリにてスキルを有効化

# 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、マルチプラットフォーム、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”

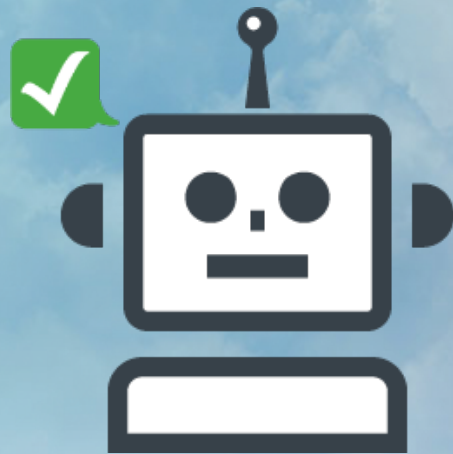


Thank you.



Special Thanks!!  
Motoshi Nakagawa - [monaka@juniper.net](mailto:monaka@juniper.net)  
Kazuki Shimizu - [kshimizu@juniper.net](mailto: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

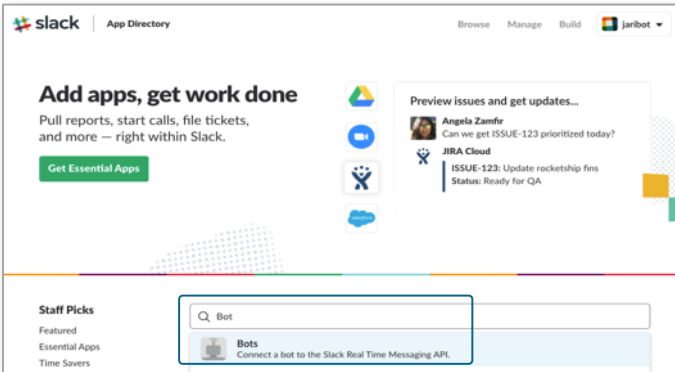
Raspbian ダウンロードはこちら <https://www.raspberrypi.org/downloads/raspbian/>

※ 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

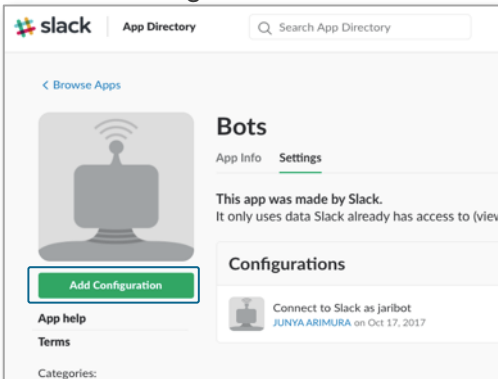
# Botの作成

## ■ Botの作成

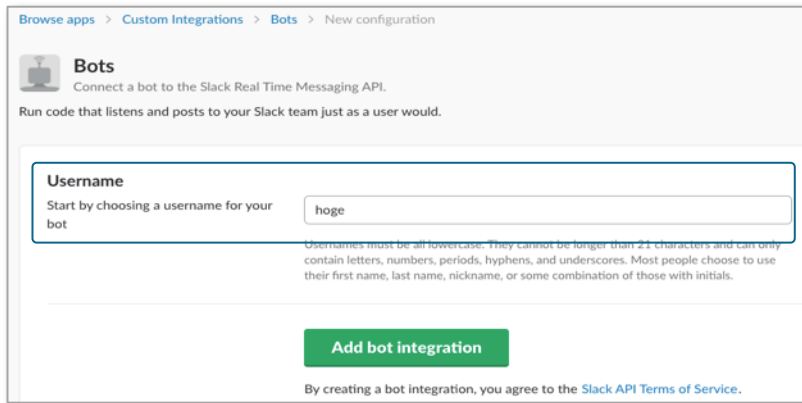
1. <workspace>.slack.jp で、App Directoryのページで'Bot'と検索



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 - 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-25664xxxxxxxxxxxxxxxxxxxxxxxxxBQ"
```

*API Token*

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

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

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\_auriika\_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_auriika_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('auriika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/xs-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: {}".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: {}".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:
    {}".format(err))
    print ("Unlocking the configuration")
    try:
        dev.cu.unlock()
    except UnlockError:
        print ("Unable to unlock configuration: {}".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: {}".format(err))
    print ("Unlocking the configuration")
    try:
        dev.cu.unlock()
    except UnlockError as err:
        print ("Unable to unlock configuration: {}".format(err))
        dev.close()
    return

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

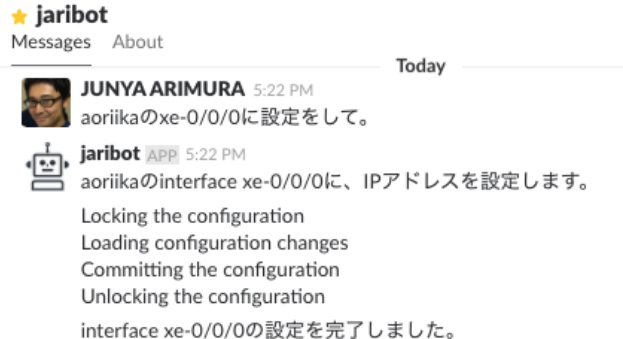
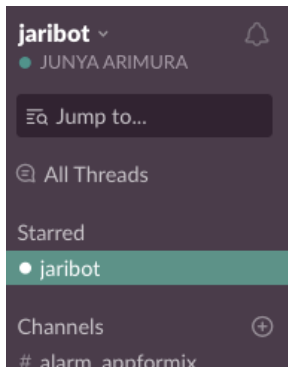
*Interface xe-0/0/0のIPアドレス未設定*

```
{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



特定キーワード  
xe-0/0/0をポスト



slackbot

python\_ioriika\_interface\_change\_xe-0-0-0.py  
@respond\_to('xe-0/0/0') にヒット!

Subprocessで'python junos-pyez-config.py'の呼び出し

python\_ioriika\_interface\_change\_xe-0-0-0.py



Python  
PyEZ

Subprocess: python junos-pyez-config.py

junos-pyez-config.pyを実行  
config/xe-0-0-0.conf に記載されたconfig ファイルを172.27.114.7/ioriika  
に Netconf で投入

Netconf の実行

- Configuration Datastore のLock
- Configurationのロード
- Candidate DatastoreのConfigurationをcommit
- Configuration DatastoreのUnlock



Netconf セッションの終了

Netconf上でエラーが無い限り、print ()の値を slackbot に返す



slackbot

hoge.stdout に返された値をmessage.replyで Slack 上に表示  
同じく、messge.reply('interface xe-0/0/0の設定を完了しました。')を表示

message.reply()が  
表示される



An aerial view of a city at dusk, with a glowing network overlay of white lines and nodes. The text "Thank you." is overlaid in white. The city features a mix of high-rise buildings and dense residential areas, with lights beginning to glow as the sun sets. The network overlay consists of several bright white nodes connected by thin white lines, forming a complex web across the sky.

Thank you.