

# ネットワーク運用を楽にする チュートリアル(実践編)



Kaoru Kitauchi  
DWANGO Co.,Ltd.

# 自己紹介

北内 薫 @kaorukit

ドワンゴから来ました

# 今日話すこと

- 運用コード化あるある
- APIとは？
- WebAPI叩いてみた

## 今日話すこと

- 運用コード化あるある
- APIとは？
- WebAPI叩いてみた

# NW運用コード化あるある

- NW機器のconfig / showを取り込んでゴ`ニョゴ`ニョ
  - Ansibleで叩いたり、自分で叩いたり
- 台帳(例:Excel)を取り込んでゴ`ニョゴ`ニョ
  - csvにして使ったり
- 管理システム(例:Zabbix)を参照してゴ`ニョゴ`ニョ
  - ファイルエクスポートしたり
  - ???

# NW運用コード化あるある

- NW機器のconfig / showを取り込んでゴニョゴニョ
  - Ansibleで叩いたり、自分で叩いたり
- 台帳(例:Excel)を取り込んでゴニョゴニョ
  - csvにして使ったり
- 管理システム(例:Zabbix)を参照してゴニョゴニョ
  - ファイルエクスポートしたり
  - **WebAPIを使ってみよう!**

# 今日話すこと

- 運用コード化あるある
- APIとは？
- WebAPI叩いてみた

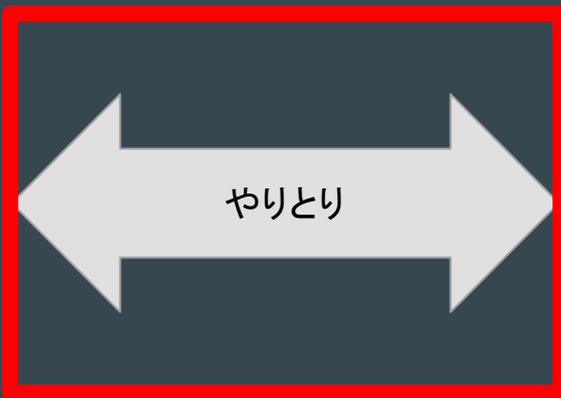
# APIとは？

Wikipedia曰く

「アプリケーションプログラミングインタフェース(API、英: Application Programming Interface)とは、広義の意味ではソフトウェアコンポーネントが互いにやりとりするのに使用するインタフェースの仕様である。」

なるほど？つまり？

あるシステム



あるシステム

# WebAPIとは？



# WebAPIは、大体お作法が公開されています

郵便番号検索API <http://zip.cgis.biz/>

## 仕様

リクエストメソッド

GET

リクエスト送信先URL

[XMLで受け取りたい場合] <http://zip.cgis.biz/xml/zip.php>

[CSVで受け取りたい場合] <http://zip.cgis.biz/csv/zip.php>

# ために叩いてみる

```
~ $ curl "http://zip.cgis.biz/xml/zip.php?zn=1030000"
<?xml version="1.0" encoding="utf-8" ?>
<ZIP_result>
  <result name="ZipSearchXML" />
  <result version="1.01" />
  <result request_url="http%3A%2F%2Fzip.cgis.biz%2Fxml%2Fzip.php%3Fzn%3D1030000" />
  <result request_zip_num="1030000" />
  <result request_zip_version="none" />
  <result result_code="1" />
  <result result_zip_num="1030000" />
  <result result_zip_version="0" />
  <result result_values_count="1" />
  <ADDRESS_value>
    <value state_kana="トウキョウト" />
    <value city_kana="チュウオウク" />
    <value address_kana="none" />
    <value company_kana="none" />
    <value state="東京都" />
    <value city="中央区" />
    <value address="none" />
    <value company="none" />
  </ADDRESS_value>
</ZIP_result>
~ $ █
```

これは便利

# 今日話すこと

- 運用コード化あるある
- APIとは？
- WebAPI叩いてみた

# 本日のWebAPI



FEATURES

TRY

DOWNLOADS

SCREENSHOTS

SUPPORT

THANKS

DOCS

BLOG

*Welcome to LibreNMS, a fully featured network monitoring system that provides a wealth of features and device support.*

**WHAT CAN IT DO?**

# やりたいこと

- Tracerouteで経由した(NW機器 / Interface)を知りたい！

```
traceroute 8.8.8.8
```

```
1  vlan100-router1
```

```
2  vlan200-router2
```

```
3  gil/0/1-router3
```

```
□
```

# やりたいこと

- Tracerouteで経由した(NW機器 / Interface)を知りたい！

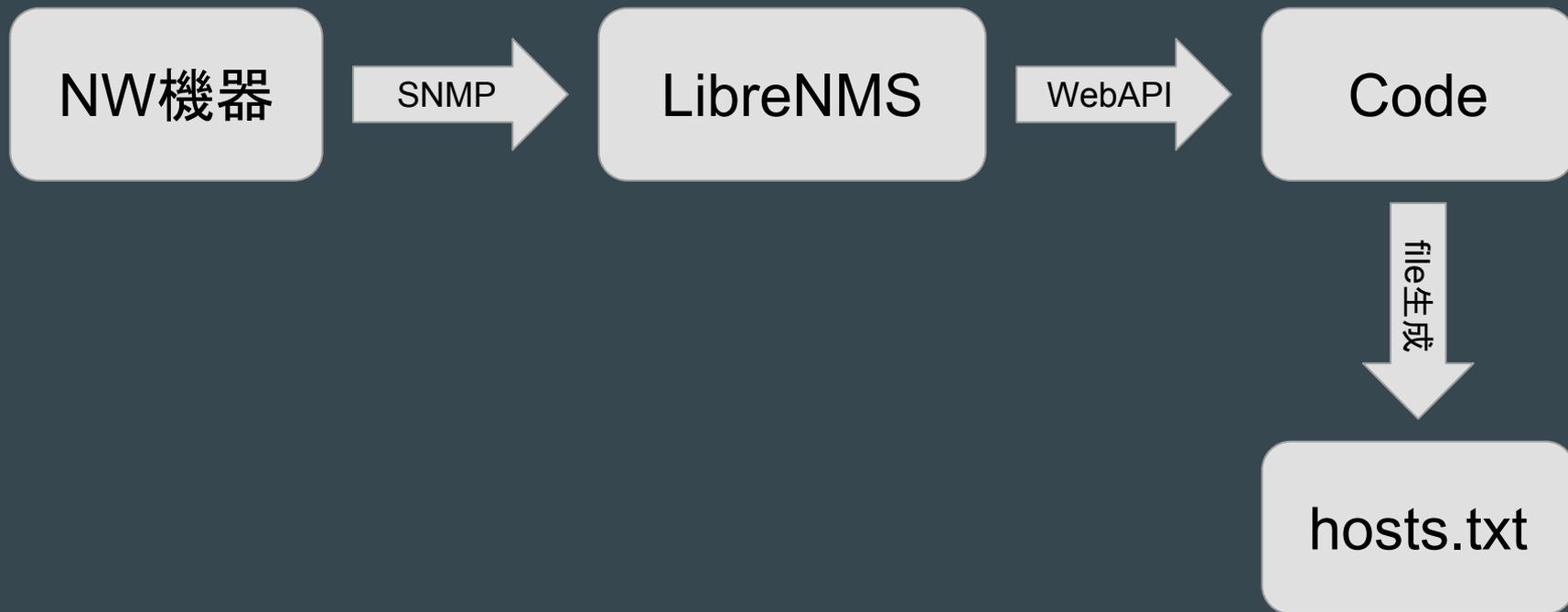
```
x.x.x.x vlan100-router1
```

```
y.y.y.y vlan200-router2
```

```
z.z.z.z gi1/0/1-router3
```

```
□
```

# やりたいこと



## 今回必要なのは・・・

- ipアドレス
- interface名
- device名

## 今回必要なのは・・・

- ipアドレス
- interface名
- device名
- 上記を相互に紐付ける何か←大事！

やってみた

# deviceのlistを作る

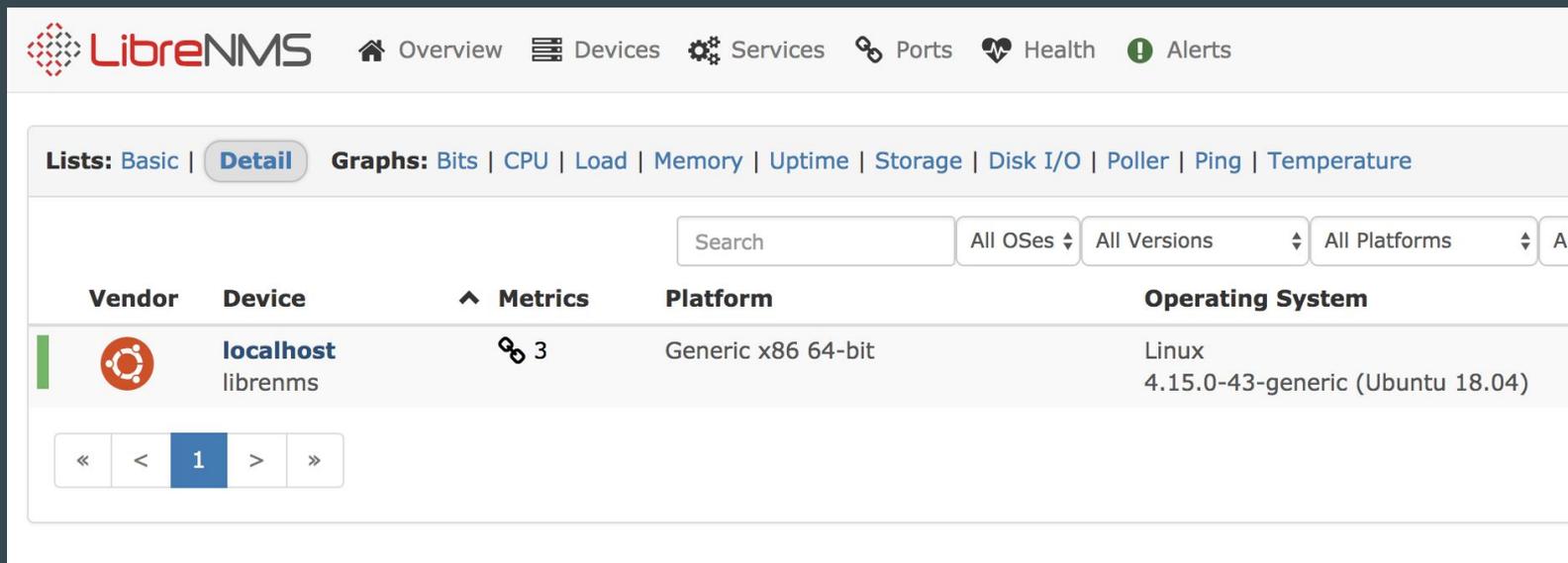
```
list_devices
```

Return a list of devices.

Route: `/api/v0/devices`

```
{
  "status": "ok",
  "count": 1,
  "devices": [
    {
      "device_id": "1",
      "hostname": "localhost",
      ...
      "serial": null,
      "icon": null
    }
  ]
}
```

# deviceのlistを作る



The screenshot displays the LibreNMS web interface. At the top, the LibreNMS logo is on the left, and navigation links for Overview, Devices, Services, Ports, Health, and Alerts are on the right. Below the navigation bar, there are tabs for Lists (Basic and Detail) and Graphs (Bits, CPU, Load, Memory, Uptime, Storage, Disk I/O, Poller, Ping, Temperature). A search bar and filter dropdowns for OSes, Versions, and Platforms are present. The main content area shows a table with one device entry: localhost (librenms), which has 3 metrics, is on a Generic x86 64-bit platform, and runs Linux 4.15.0-43-generic (Ubuntu 18.04). A pagination bar at the bottom shows page 1 of 1.

LibreNMS

Overview Devices Services Ports Health Alerts

Lists: Basic | **Detail** | Graphs: Bits | CPU | Load | Memory | Uptime | Storage | Disk I/O | Poller | Ping | Temperature

Search All OSes All Versions All Platforms All

Vendor	Device	Metrics	Platform	Operating System
	localhost librenms	 3	Generic x86 64-bit	Linux 4.15.0-43-generic (Ubuntu 18.04)

« < 1 > »

# deviceのlistを作る

```
headers: dict = {"X-Auth-Token": token}
devices_dict = {}

res = requests.get(f"{url}api/v0/devices", headers=headers, verify=False,)
devices_list = json.loads(res.text)["devices"]

for device in devices_list:
    devices_dict[device["device_id"]] = device["sysName"]
```

# deviceのlistを作る

```
device_id  sysName
```

```
{1: 'librenms'}
```

# portのlistを作る

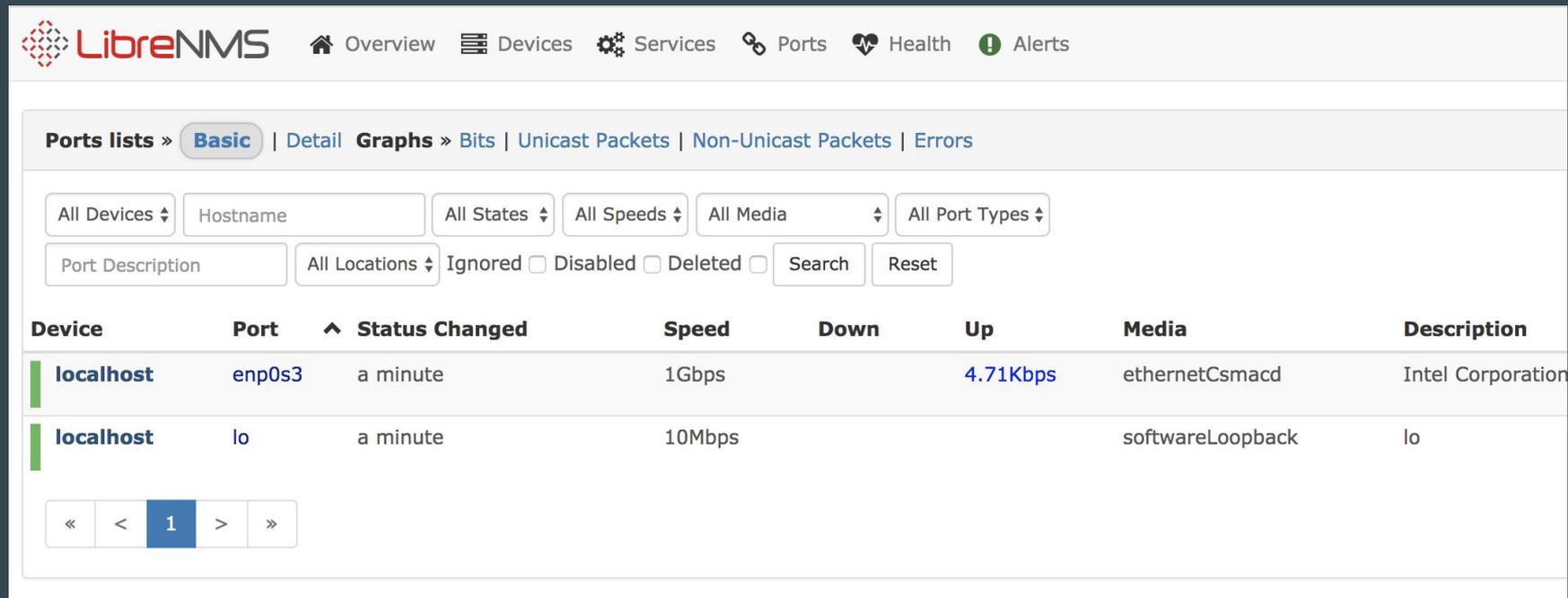
get\_all\_ports

Get info for all ports on all devices. Strongly recommend that you use the `columns` parameter to avoid pulling too much data.

Route: `/api/v0/ports`

```
{
  "status": "ok",
  "message": "",
  "ports": [
    {
      "ifName": "Gi0/0/0",
      "port_id": "1"
    },
    {
      "ifName": "Gi0/0/1",
      "port_id": "2"
    },
    ...
    {
      "ifName": "Vlan 3615",
      "port_id": "5488"
    }
  ]
}
```

# portのlistを作る



The screenshot shows the LibreNMS interface. At the top, there is a navigation bar with the LibreNMS logo and menu items: Overview, Devices, Services, Ports, Health, and Alerts. Below the navigation bar, there is a breadcrumb trail: Ports lists > Basic | Detail Graphs > Bits | Unicast Packets | Non-Unicast Packets | Errors. The main content area contains several filter controls: All Devices (dropdown), Hostname (text input), All States (dropdown), All Speeds (dropdown), All Media (dropdown), All Port Types (dropdown), Port Description (text input), All Locations (dropdown), Ignored (checkbox), Disabled (checkbox), Deleted (checkbox), Search (button), and Reset (button). Below the filters is a table with the following columns: Device, Port, Status Changed, Speed, Down, Up, Media, and Description. The table contains two rows of data. The first row shows a port named enp0s3 on localhost with a speed of 1Gbps and an up speed of 4.71Kbps. The second row shows a port named lo on localhost with a speed of 10Mbps. At the bottom of the table, there is a pagination control showing page 1 of 1.

Ports lists > **Basic** | Detail Graphs > Bits | Unicast Packets | Non-Unicast Packets | Errors

All Devices ▾ Hostname All States ▾ All Speeds ▾ All Media ▾ All Port Types ▾

Port Description All Locations ▾ Ignored  Disabled  Deleted  Search Reset

Device	Port	▲ Status Changed	Speed	Down	Up	Media	Description
localhost	enp0s3	a minute	1Gbps		4.71Kbps	ethernetCsmacd	Intel Corporation
localhost	lo	a minute	10Mbps			softwareLoopback	lo

« < 1 > »

# portのlistを作る

```
headers: dict = {"X-Auth-Token": token}
```

```
You, a day ago • commit
```

```
res = ""
```

```
res = requests.get(  
    f"{url}api/v0/ports?columns=ifName%2Cport_id%2Cdevice_id", headers=headers, verify=False)
```

```
ports_list = json.loads(res.text)["ports"]
```

```
if json.loads(res.text)["status"] == "ok":
```

```
    for port in ports_list:
```

```
        if ports_dict.get(port["device_id"]):
```

```
            ports_dict[port["device_id"]].update({port["port_id"]: port["ifName"]})
```

```
        else:
```

```
            ports_dict[port["device_id"]] = {port["port_id"]: port["ifName"]}
```

## portのlistを作る

```
{  
  1: {  
    1: 'lo',  
    3: 'enp0s3'  
  }  
}
```

**device\_id**

port\_id: ifName

port\_id: ifName

# addressのlistを作る

get\_device\_ip\_addresses

Get a list of IP addresses (v4 and v6) associated with a device.

Route: /api/v0/devices/:hostname/ip

- hostname can be either the device hostname or id

```
{
  "status": "ok",
  "message": "",
  "addresses": [
    {
      "ipv4_address_id": "290",
      "ipv4_address": "192.168.99.292",
      "ipv4_prefixlen": "30",
      "ipv4_network_id": "247",
      "port_id": "323",
      "context_name": ""
    }
  ]
}
```

# addressのlistを作る

get\_device\_ip\_addresses

Get a list of IP addresses (v4 and v6) associated with a device.

Route: /api/v0/devices/:hostname/ip

- hostname can be either the device hostname or id

```
{
  "status": "ok",
  "message": "",
  "addresses": [
    {
      "ipv4_address_id": "290",
      "ipv4_address": "192.168.99.292",
      "ipv4_prefixlen": "30",
      "ipv4_network_id": "247",
      "port_id": "323",
      "context_name": ""
    }
  ]
}
```

# addressのlistを作る

The screenshot shows the LibreNMS interface for a device named 'localhost'. The main navigation bar includes 'Overview', 'Devices', 'Services', 'Ports', 'Health', and 'Alerts'. The sub-navigation bar includes 'Overview', 'Graphs', 'Health', 'Ports', 'Inventory', 'Services', 'Logs', and 'Alerts'. The 'Ports' section is active, showing a list of ports with columns for 'Port' and 'Traffic'.

Port	Traffic
<b>lo</b> lo 127.0.0.1/8	← 688 bps → 688 bps ← 1 pps → 1 pps
<b>enp0s3</b> Intel Corporation 82540EM Gigabit Ethernet Controller 10.0.2.15/24	← 296 bps → 576 bps ← 0 pps → 0 pps

# addressのlistを作る

```
headers: dict = {"X-Auth-Token": token}

for device_id in tqdm(devices_dict.keys()):

    addresses_list: list = []
    res = ""

    res = requests.get(
        f"{url}api/v0/devices/{device_id}/ip", headers=headers, verify=False)

    if json.loads(res.text)["status"] == "ok":
        addresses_list = json.loads(res.text)["addresses"]

        for address in addresses_list:
            if addr_dict.get(device_id):
                addr_dict[device_id].update({address["port_id"]: address["ipv4_address"]})
            else:
                addr_dict[device_id] = {address["port_id"]: address["ipv4_address"]}
```

## addressのlistを作る

```
{  
  1: {  
    1: '127.0.0.1',  
    3: '10.0.2.15'  
  }  
}
```

**device\_id**

port\_id: ipv4\_address

port\_id: ipv4\_address

# 出来上がった3つのlistを組み合わせてhosts出力

```
with open("raw_hosts.txt", "a") as raw_hosts:
    for device_id, address_dict in addr_dict.items():
        for port_id in address_dict.keys():
            raw_hosts.write(
                f"{addr_dict[device_id][port_id]} {ports_dict[device_id][port_id]}.{devices_dict[device_id]}\n")
```

```
127.0.0.1 lo.librenms
10.0.2.15 enp0s3.librenms
```

# 出来上がった3つのlistを組み合わせてhosts出力

```
with open("raw_hosts.txt", "a") as raw_hosts:
    for device_id, address_dict in addr_dict.items():
        for port_id in address_dict.keys():
            raw_hosts.write(
                f"{address_dict[device_id][port_id]} {ports_dict[device_id][port_id]}.{devices_dict[device_id]}\n")
```

```
127.0.0.1 lo.librenms
10.0.2.15 enp0s3.librenms
```

もうちょっと  
ゴニョゴニョしたい

# 自社内で使っているprefixだけ抽出したい

```
DEFINED_PREFIX = [  
    "10.0.0.0/8",  
    "172.16.0.0/12",  
    "192.168.0.0/16"  
]  
  
hosts_list: list = []  
  
with open(raw_hosts, "r") as raw_hosts:  
    for row in raw_hosts:  
        ipv4_addr = row.split()[0]  
  
        for ipv4_nw in DEFINED_PREFIX:  
            if ipaddress.IPv4Address(ipv4_addr) in ipaddress.IPv4Network(ipv4_nw):  
                hosts_list.append(row)
```

# 禁則文字とかdotを置換したい

```
PROHIBITION_CHARACTER = r"(/|@|_|\.| |#|\$)"

hosts_list: list = []

for row in raw_hosts:

    addr = row.split()[0]
    domain = re.sub(PROHIBITION_CHARACTER, "-", row.split()[-1])

    hosts_list.append(f"{addr} {domain.lower()}\n")
```

これがこうじゃ！

```
127.0.0.1 lo.librenms  
10.0.2.15 enp0s3.librenms
```



```
10.0.2.15 enp0s3-librenms
```

## さいごに

- WebAPIコワクナイヨ
  - ドキュメントはちゃんと読もう
- WebAPI最強、デハナイヨ
  - 適材適所を考えよう
- NW運用コード化タノシイヨ
  - Happy Hacking Hackathon!