

# REST-API/NetmikoによるNW管理 (Juniper VSRX編)

2021/03/07 第2版

# 目 的

Juniperネットワーク社のFirewallであるVSRXのREST APIを使用したNW管理の要領について説明します

# 内 容

- 1 VSRXとは？
- 2 REST APIとは？
- 3 REST APIによるVSRXの管理
  - (1) VSRXでの記述形式
  - (2) VSRXにおける事前設定
  - (3) APIエクスプローラによる確認
  - (4) CurlコマンドによるAPIアクセス
  - (5) PythonスクリプトによるAPIアクセス
- ア 環境準備
- イ スクリプト作成及び実行
- 4 NetmikoモジュールによるNW管理
  - (1) Netmikoとは？
  - (2) VSRXにおける事前設定

# 目 的

Juniperネットワーク社のFirewallであるVSRXのREST-APIを使用したNW管理の要領について説明します

# 内 容

## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

## 5 参考資料

○VSRXによるネットワーク構築(自宅PC編)

○Junos OS RESTAPI Guide

Juniper ネットワーク

▪ Juniper公式ドキュメントです。

○ネットワークAPIのあれこれ

<https://www.slideshare.net/kentaroebisawa/api-enog37>

○gitbub上でJunos REST APIアクセスのスクリプト例が公開されています！

[https://github.com/ksator/junos\\_automation\\_with\\_rest\\_calls](https://github.com/ksator/junos_automation_with_rest_calls)

○Netmikoを使用する際の関連資料がのってます

<https://github.com/ktbyers/netmiko>

1

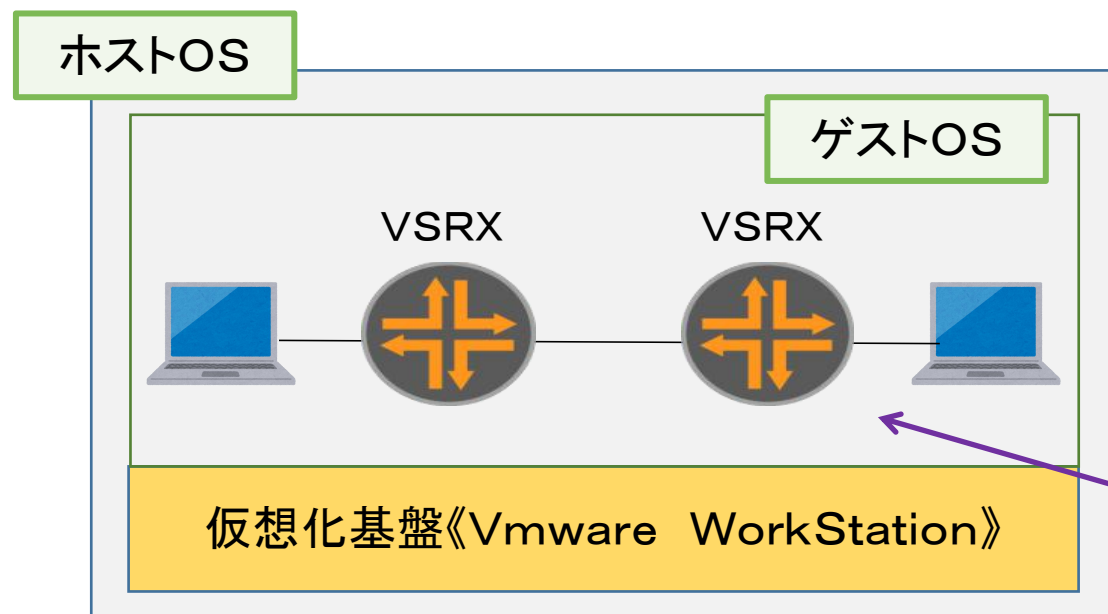
VSRXとは？

# 1 VSRXとは？

VSRXとはVmware／KVM上で動作する、ブランチ型SRXシリーズサービスゲートウェイを基にしたバーチャルアプライアンス型セキュアルーターです。

サポートする機能等については、リリースノートを参照してください。

(今回はWINDOWSホスト上でVmware Workstation 15 上で使用しています)



PC上のVMで  
仮想NWが構築可能

以下のドキュメントも参考になります

VSRX on Your Laptop PCで始めるVSRX

～JUNOSを触ってみよう！～

Juniper Network

## 2 REST APIとは？

## 2 REST APIとは？

○ REST APIは以下のようなものです

RESTful API(REST API)とは、Webシステムを外部から利用するためのプログラムの呼び出し規約(API)の種類の一つになります

リソースの操作はHTTPメソッドによって指定します。

(例：取得ならGETメソッド、書き込みならPOSTメソッド)

結果はXMLやHTML、JSONなどの形式で返信されます

また処理結果はHTTPステータスコードで通知するという原則が含まれることもある。

引用 IT用語辞典 e-Words

[https://e-words.jp/w/RESTful\\_API.html](https://e-words.jp/w/RESTful_API.html)

## 3 REST APIによるVSRXの管理

### (1) VSRXにおける記述形式



### 3 REST APIによるVSRXの管理

#### (1) VSRXにおける記述形式

##### 【JUNOS REST APIの表現方法】

- ・CLIに紐付いたRPC MethodをURIに記述
- ・インタフェース名などのパラメータもURI中に?で指定

##### 【記述形式】

○ `scheme;device-name:port/rpc/method[@attributes]?params`

- `scheme`: http or https
- `method`: rpc command （各コマンドに対してそれぞれRPC methodが定義）
- `params`: Optional parameter values(name[=value])
- `@attributes`で指定: `@format=json` （response Formatも指定可能）  
HTTP header “Accept:”で指定: application/xml ,application/json

## 3 REST APIによるVSRXの管理

### (2) VSRXにおける事前設定

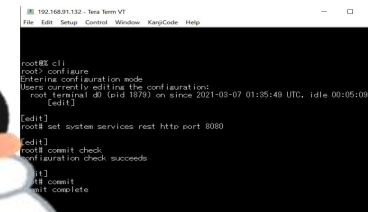
### 3 REST APIによるVSRXの管理

#### (2) VSRXにおける事前設定

VSRX



- ① APIにアクセスする設定を実施
- ② 取得する情報のメゾットを確認



#### ① APIでアクセスするための設定を実施

```
root#set system services rest http port 8080
```

→ APIアクセスするポート番号を8080に設定

#### ② 取得する情報のRPCを確認

(例) Junosバージョン情報をXML形式で確認

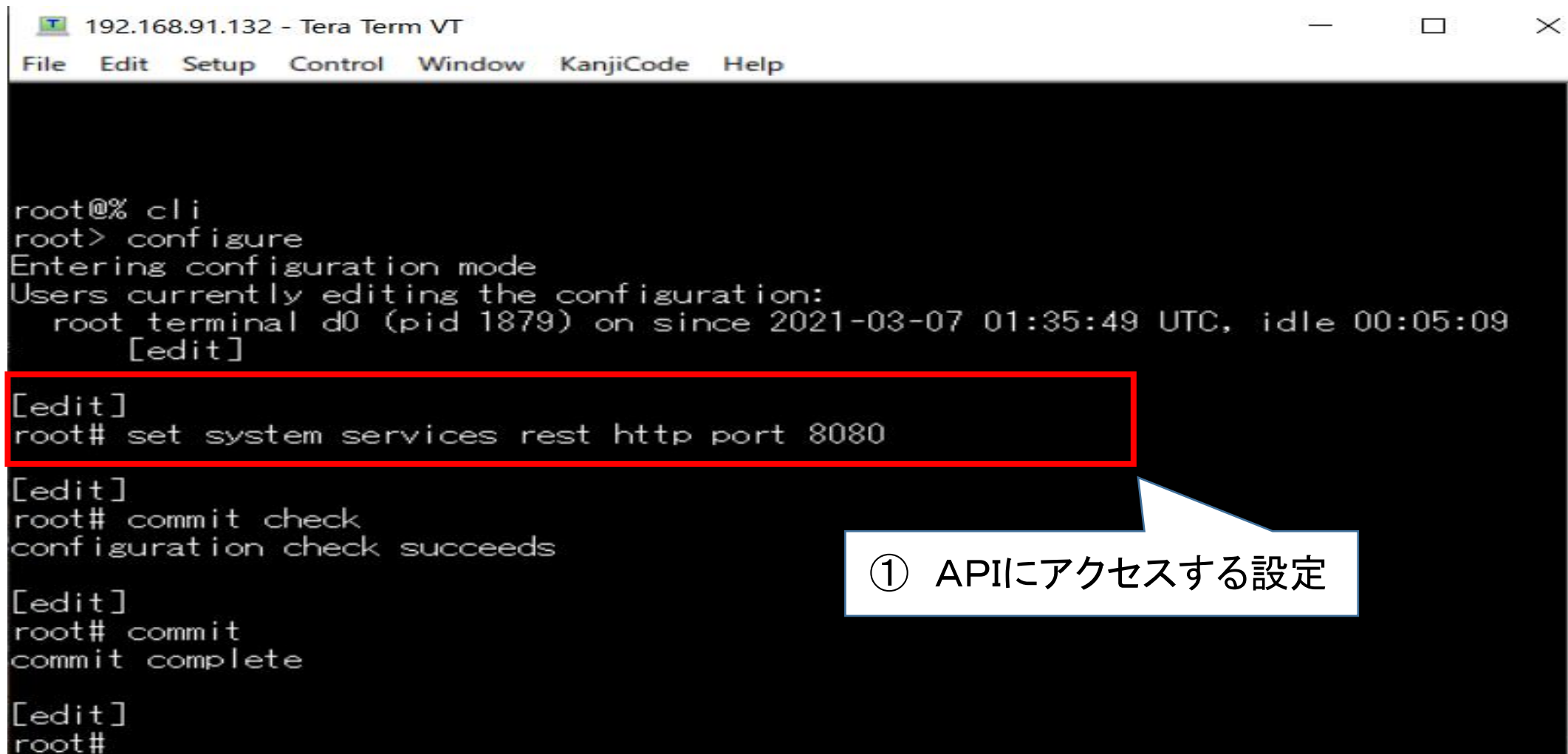
```
root#show version | display xml rpc
```

→ VSRXのバージョン情報のRPCコマンドを確認

### 3 REST APIによるVSRXの管理

#### (2) VSRXにおける事前設定

##### ① APIでアクセスするための設定を実施



```
192.168.91.132 - Tera Term VT
File Edit Setup Control Window KanjiCode Help

root@% cli
root> configure
Entering configuration mode
Users currently editing the configuration:
  root terminal d0 (pid 1879) on since 2021-03-07 01:35:49 UTC, idle 00:05:09
  [edit]

[edit]
root# set system services rest http port 8080

[edit]
root# commit check
configuration check succeeds

[edit]
root# commit
commit complete

[edit]
root#
```

① APIにアクセスする設定

### 3 REST APIによるVSRXの管理

#### (2) VSRXにおける事前設定

##### ② 取得する情報のRPCを確認

(例) show version 情報の  
RPC形式を確認

192.168.91.132 - Tera Term VT

File Edit Setup Control Window KanjiCode Help

Possible completions:

<[Enter]>	Execute this command
groups	Tag inherited data with the source group name
interface-ranges	Tag inherited data with the source interface-range name
rpc	Show corresponding xml rpc
	Pipe through a command

```
root> show version | display xml rpc
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/15.1X49/junos">
  <rpc>
    <get-software-information>
    </get-software-information>
  </rpc>
  <cli>
    <banner></banner>
  </cli>
</rpc-reply>

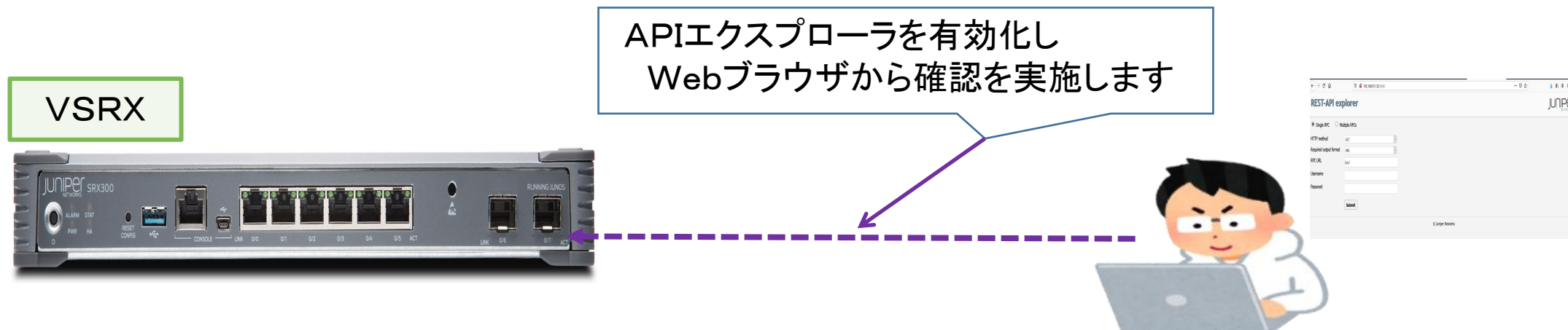
root>
```

## 3 REST APIによるVSRXの管理

### (3) APIエクスプローラによる確認

### 3 REST APIによるVSRXの管理

#### (3) APIエクスプローラによる確認



- ① APIエクスプローラを使用するための設定  
`root#set system services rest enable-explorer`
- ② WebブラウザからAPIエクスプローラへアクセス  
urlは `http://IPアドレス:ポート番号` (今回は8080)

### 3 REST APIによるVSRXの管理

#### (3) APIエクスプローラによる確認

##### ① APIエクスプローラを使用するための設定

```
root# set system services rest enable-explorer ?
Possible completions:
  <[Enter]>      Execute this command
+ apply-groups   Groups from which to inherit configuration data
+ apply-groups-except Don't inherit configuration data from these groups
> control        Control of the rest-api process
> http           Unencrypted HTTP connection settings
> https          Encrypted HTTPS connections
> traceoptions   Trace options for rest-api service
> |              Pipe through a command
```

```
[edit]
root# set system services rest enable-explorer
```

```
[edit]
root# commit
commit complete
```

```
[edit]
root#
```

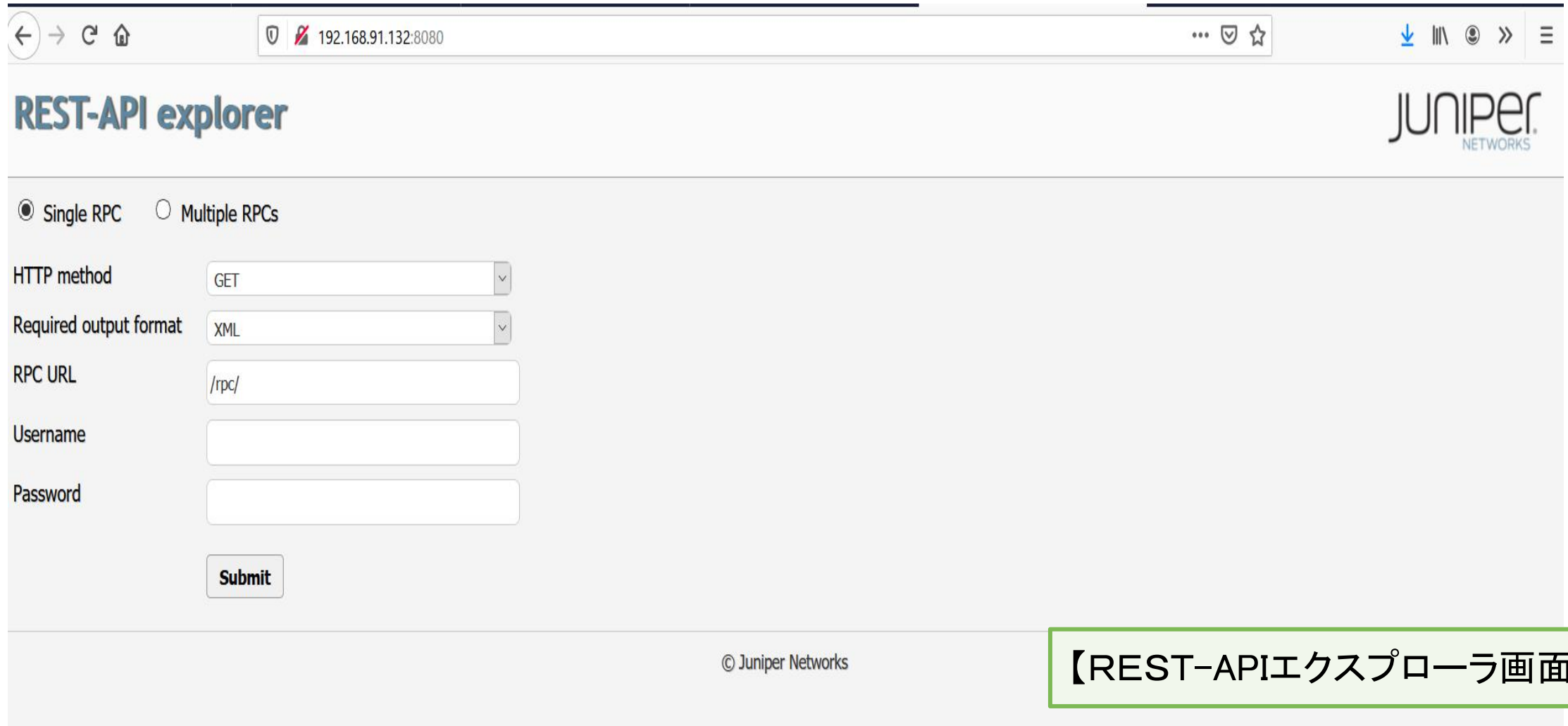
APIエクスプローラ  
を有効化する



### 3 REST APIによるVSRXの管理

(3) APIエクスプローラによる確認

② WebブラウザからAPIエクスプローラへアクセス



The screenshot shows a web browser window with the address bar displaying '192.168.91.132:8080'. The page title is 'REST-API explorer' and the Juniper Networks logo is in the top right corner. The interface includes a radio button selection for 'Single RPC' (selected) and 'Multiple RPCs'. Below this are dropdown menus for 'HTTP method' (set to 'GET') and 'Required output format' (set to 'XML'). There are text input fields for 'RPC URL' (containing '/rpc/'), 'Username', and 'Password'. A 'Submit' button is located at the bottom of the form. The footer of the page contains the text '© Juniper Networks'.

REST-API explorer

JUNIPER NETWORKS

☒ Single RPC ☐ Multiple RPCs

HTTP method GET

Required output format XML

RPC URL /rpc/

Username

Password

Submit

© Juniper Networks

【REST-APIエクスプローラ画面】

### 3 REST APIによるVSRXの管理

(3) APIエクスプローラによる確認

② WebブラウザからAPIエクスプローラへアクセス

○APIエクスプローラによる検索要領

The screenshot shows the REST-API explorer interface in Mozilla Firefox. The browser address bar shows the URL `130.230.0.1:8080`. The page title is "REST-API explorer".

**Request Configuration (Red Box):**

- HTTP method: GET
- Required output format: XML
- RPC URL: `/get-interface-information?interface-name=fxp0`
- Username: root
- Password: [masked]
- Submit button

**Response Details (Purple Box):**

**Response Headers:**

```
Content-Type: application/xml; charset=utf-8
Transfer-Encoding: chunked
Date: Sun, 19 Apr 2020 07:57:52 GMT
Server: lighttpd/1.4.32
```

**Response Body:**

```
<interface-information xmlns="http://xml.juniper.net/junos/15.1R3/interface-information">
  <physical-interface>
    <name>fxp0</name>
    <admin-status junos:format="Enabled">up</admin-status>
    <oper-status>up</oper-status>
    <local-index>65</local-index>
    <snmp-index>1</snmp-index>
    <if-type>Ethernet</if-type>
    <link-level-type>Ethernet</link-level-type>
    <mtu>1514</mtu>
    <speed>1000mbps</speed>
    <if-device-flags>
      <ifdf-present/>
      <ifdf-running/>
    </if-device-flags>
    <ifd-specific-config-flags>
      </ifd-specific-config-flags>
    <if-config-flags>
      <ifc-snmp-traps/>
    </if-config-flags>
  </physical-interface>
</interface-information>
```

**Request Headers:**

```
GET /rpc/get-interface-information?interface-name=fxp0
Authorization: Basic Og==
Accept: application/xml
Content-Type: application/xml
```

**cURL request:**

```
curl -X GET http://130.230.0.1:8080/rpc/get-interface-information?interface-name=fxp0 -u root: -H 'Accept: application/xml'
```

Annotations:

- ア** 問い合わせるのに必要な情報を入力 (Input the necessary information to inquire)
- イ** SRXから返信された内容 (Content returned from SRX)

A cartoon character is shown at the bottom, looking at a laptop.

## 3 REST APIによるVSRXの管理

### (4) CURLコマンドによるAPIアクセス

### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス



- 以下のcurlコマンドを実行し、情報を取得します。
- #curl△-u△”ユーザ/パスワード”http://IPアドレス:ポート番号/rpc/rpcコマンド
- ★デフォルトの出力結果はXML形式で返信されます。
- 【3つの例について紹介します】
- ・バージョン情報の取得
  - ・インタフェース情報の取得
  - ・セキュリティーポリシー情報の取得

## 3 REST APIによるVSRXの管理

### (3) CurlコマンドによるAPIアクセス

Ubuntu18

#### ○ Ubuntu18にcurlコマンドをインストールします

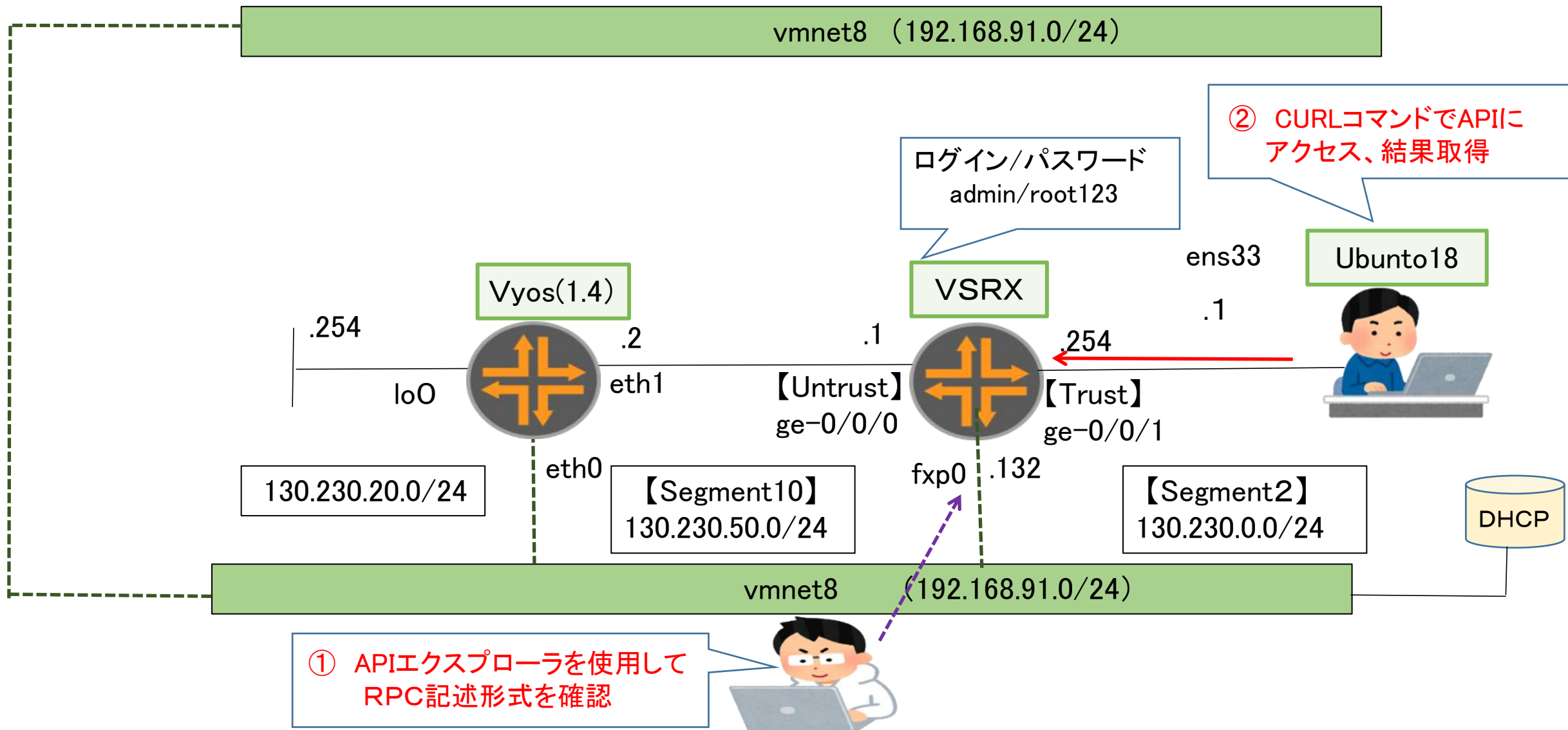
```
gorosuke@ubuntu:~$ sudo apt install curl
[sudo] gorosuke のパスワード:
パッケージリストを読み込んでいます... 完了
依存関係ツリーを作成しています
状態情報を読み取っています... 完了
以下の追加パッケージがインストールされます:
  libcurl4
以下のパッケージが新たにインストールされます:
  curl libcurl4
アップグレード: 0 個、新規インストール: 2 個、削除: 0 個、保留: 500 個。
378 kB のアーカイブを取得する必要があります。
この操作後に追加で 1,053 kB のディスク容量が消費されます。
続行しますか? [Y/n] y
取得:1 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 libcurl4 amd64 7.58.0-2ubuntu3.16 [220 kB]
取得:2 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 curl amd64 7.58.0-2ubuntu3.16 [159 kB]
378 kB を 3秒 で取得しました (123 kB/s)
以前に未選択のパッケージ libcurl4:amd64 を選択しています。
```

#### ○ インストール後の確認

```
gorosuke@ubuntu:~$ curl -V
curl 7.58.0 (x86_64-pc-linux-gnu) libcurl/7.58.0 OpenSSL/1.1.1 zlib/1.2.11 libidn2/2.0.4 libpsl/0.19.1 (+libidn2/2.0.4) nghttp2/1.30.0 librtmp/2.3
Release-Date: 2018-01-24
Protocols: dict file ftp ftps gopher http https imap imaps ldap ldaps pop3 pop3s rtmp rtsp smb smbs smtp smtps telnet tftp
Features: AsynchDNS IDN IPv6 Largefile GSS-API Kerberos SPNEGO NTLM NTLM_WB SSL libz TLS-SRP HTTP2 UnixSockets HTTPS-proxy PSL
```

### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス



## 3 REST APIによるVSRXの管理

### (4) CURLコマンドによるAPIアクセス

- ・ バージョン情報の確認



### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス

【XML形式】

- ・ バージョン情報の確認

```
[root@localhost goro]# curl -u "root:root123" http://130.230.0.1:8080/rpc/get-software-information
<software-information>
<host-name/>
<product-model>vsrx</product-model>
<product-name>vsrx</product-name>
<junos-version>15.1X49-D140.2</junos-version>
<package-information>
<name>junos</name>
<comment>JUNOS Software Release [15.1X49-D140.2]</comment>
</package-information>
</software-information>
[root@localhost goro]#
```

バージョン情報を  
Curlコマンドにより取得

結果の返信  
→ 通常はXML形式となる



### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス

【TEXT形式】

- ・ バージョン情報の確認

```
[goro@localhost ~]$ su
```

```
パスワード:
```

```
[root@localhost goro]# curl http://130.230.0.1:8080/rpc/get-software-information -u "root:root123" -H "Content-Type: plain/text" -H "Accept: text/plain" -d "<brief/>"
```

```
Model: vsrx
```

```
Junos: 15.1X49-D140.2
```

```
JUNOS Software Release [15.1X49-D140.2]
```

```
[root@localhost goro]#
```

バージョン情報を  
Curlコマンドにより取得  
(テキスト形式で要求)

結果の返信  
→ テキスト形式で返信

### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス

【JSON形式】

- ・ バージョン情報の確認

```
goro@localhost:/home/goro
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
[goro@localhost ~]$ su
パスワード:
[root@localhost goro]# curl http://130.230.0.1:8080/rpc/get-software-information -u
"root:root123" -H "Content-Type: plain/text" -H "Accept: application/json"
{
  "software-information" : [
    {
      "host-name" : [
        {
        }
      ],
      "product-model" : [
        {
          "data" : "vsrx"
        }
      ],
      "product-name" : [
        {
          "data" : "vsrx"
        }
      ]
    }
  ]
}
```

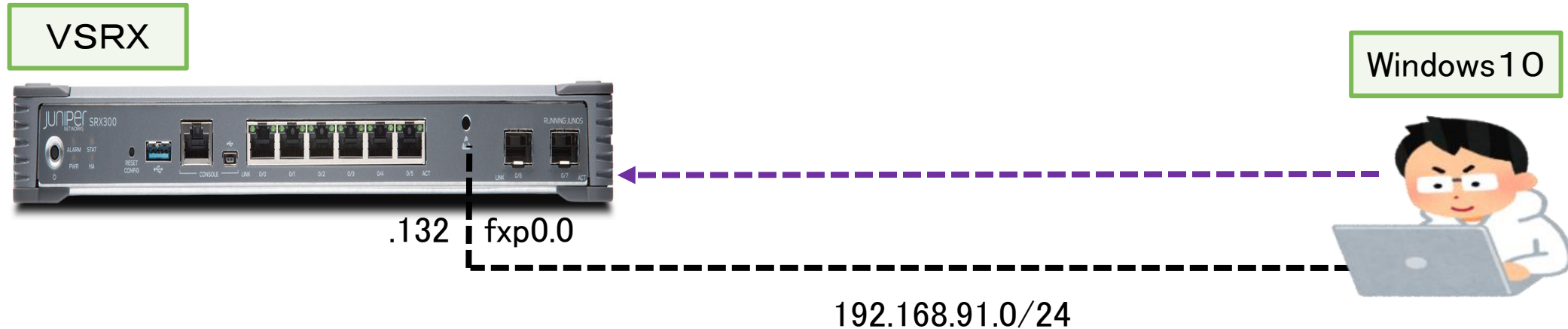
バージョン情報を  
Curlコマンドにより取得  
(テキスト形式で要求)

結果の返信  
→ JSON形式で返信

### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス

【参考】 Windows10であればCurlコマンドを使用することができます！



```
C:\Users\ユーザー>curl http://192.168.91.132:8080/rpc/get-system-uptime-information -u "admin:root123" -H
"Content-Type: application/xml" -H "Accept: text/plain"
Current time: 2022-03-28 10:49:38 UTC
Time Source: LOCAL CLOCK
System booted: 2022-03-28 10:03:20 UTC (00:46:18 ago)
Protocols started: 2022-03-28 10:03:20 UTC (00:46:18 ago)
Last configured: 2022-03-27 01:04:36 UTC (1d 09:45 ago) by admin
10:49AM up 46 mins, 2 users, load averages: 0.10, 0.11, 0.08
```

## 3 REST APIによるVSRXの管理

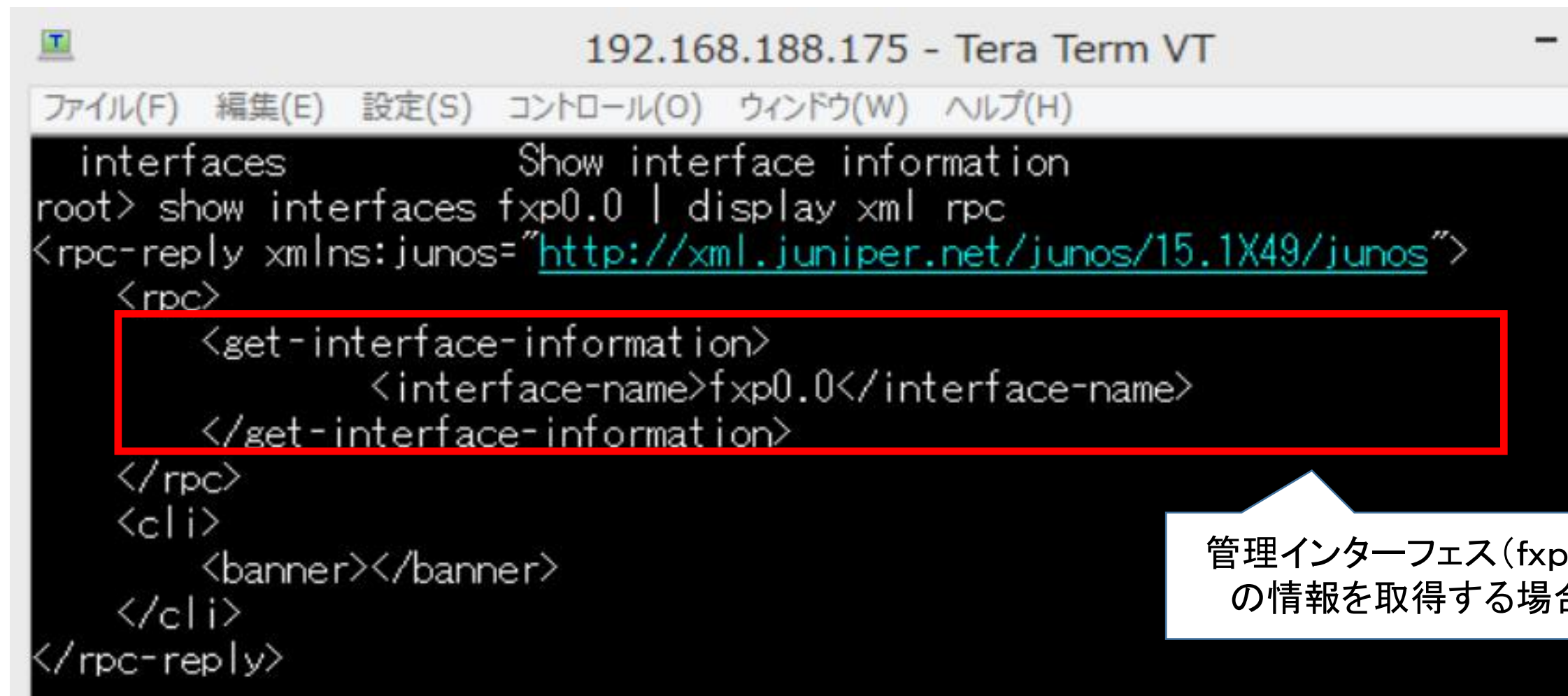
### (4) CurlコマンドによるAPIアクセス

- ・ インタフェース情報の取得

### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス

- ・ インタフェース情報の取得



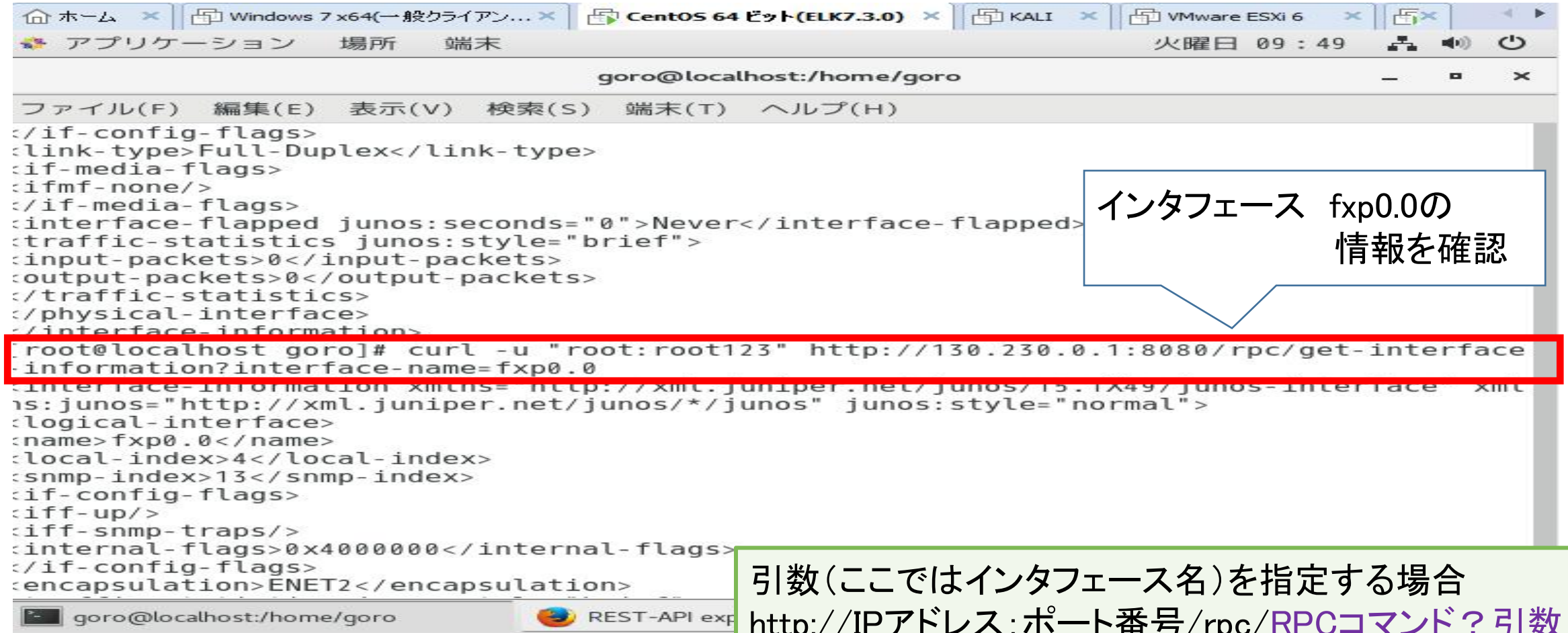
```
192.168.188.175 - Tera Term VT
ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H)
interfaces          Show interface information
root> show interfaces fxp0.0 | display xml rpc
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/15.1X49/junos">
  <rpc>
    <get-interface-information>
      <interface-name>fxp0.0</interface-name>
    </get-interface-information>
  </rpc>
  <cli>
    <banner></banner>
  </cli>
</rpc-reply>
```

管理インターフェース(fxp)  
の情報を取得する場合

### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス

- ・ インタフェース情報の取得



The screenshot shows a terminal window with a Junos configuration and a REST API curl command. The configuration is for interface fxp0.0. The curl command is used to retrieve the interface information via the REST API. A red box highlights the curl command, and a blue box points to it with the text 'インタフェース fxp0.0の情報を確認'. A green box at the bottom explains the URL format: '引数(ここではインタフェース名)を指定する場合 http://IPアドレス:ポート番号/rpc/RPCコマンド? 引数のように表現します'.

```
goro@localhost:/home/goro
ファイル(F) 編集(E) 表示(V) 検索(S) 端末(T) ヘルプ(H)
</if-config-flags>
<link-type>Full-Duplex</link-type>
<if-media-flags>
<ifmf-none/>
</if-media-flags>
<interface-flapped junos:seconds="0">Never</interface-flapped>
<traffic-statistics junos:style="brief">
<input-packets>0</input-packets>
<output-packets>0</output-packets>
</traffic-statistics>
</physical-interface>
</interface-information>
[root@localhost goro]# curl -u "root:root123" http://130.230.0.1:8080/rpc/get-interface-information?interface-name=fxp0.0
<interface-information xmlns="http://xml.juniper.net/junos/15.1X49/junos-interface" xmlns:junos="http://xml.juniper.net/junos/*/junos" junos:style="normal">
<logical-interface>
<name>fxp0.0</name>
<local-index>4</local-index>
<snmp-index>13</snmp-index>
<if-config-flags>
<iff-up/>
<iff-snmp-traps/>
<internal-flags>0x4000000</internal-flags>
</if-config-flags>
<encapsulation>ENET2</encapsulation>
</logical-interface>
</interface-information>
```

インタフェース fxp0.0の  
情報を確認

引数(ここではインタフェース名)を指定する場合  
http://IPアドレス:ポート番号/rpc/RPCコマンド? 引数  
のように表現します

### 3 REST APIによるVSRXの管理

#### (4) CurlコマンドによるAPIアクセス

- ・ セキュリティポリシー情報の取得



### 3 REST APIによるVSRXの管理

#### (3) CurlコマンドによるAPIアクセス

- ・ セキュリティポリシー情報の取得

```
[root@localhost goro]# curl http://130.230.0.1:8080/rpc/get-firewall-policies -u "root:root123" -H "Content-Type: plain/text" -H "Accept: text/plain"
```

```
Default policy: deny-all
```

```
From zone: untrust, To zone: trust
```

```
Policy: untrust_to_trust, State: enabled, Index: 4, Scope Policy: 0, Sequence number: 1
```

```
Source addresses: any
```

```
Destination addresses: any
```

```
Applications: any
```

```
Action: permit
```

```
From zone: trust, To zone: untrust
```

```
Policy: trust_to_untrust, State: enabled, Index: 5, Scope Policy: 0, Sequence number: 1
```

```
Source addresses: any
```

```
Destination addresses: any
```

```
Applications: any
```

```
Action: permit
```

```
[root@localhost goro]#
```

セキュリティポリシー  
(Firewallルール)情報を  
テキスト形式で表示



## 3 REST APIによるVSRXの管理

### (5) PythonスクリプトによるAPIアクセス

### 3 REST APIによるVSRXの管理

(5) PythonスクリプトによるAPIアクセス

○以下のパスにスクリプトを作成していきましょう！

/home/gorosuke/python3

```
gorosuke@ubuntu:/home$ cd /home/gorosuke/python3/  
gorosuke@ubuntu:~/python3$ ls -l  
合計 8  
-rw-r--r-- 1 gorosuke gorosuke 422  4月  2 20:38 interface-fxp0-api-text.py  
-rw-r--r-- 1 gorosuke gorosuke 435  4月  2 16:44 interface-fxp0-api.py  
gorosuke@ubuntu:~/python3$ pwd  
/home/gorosuke/python3
```



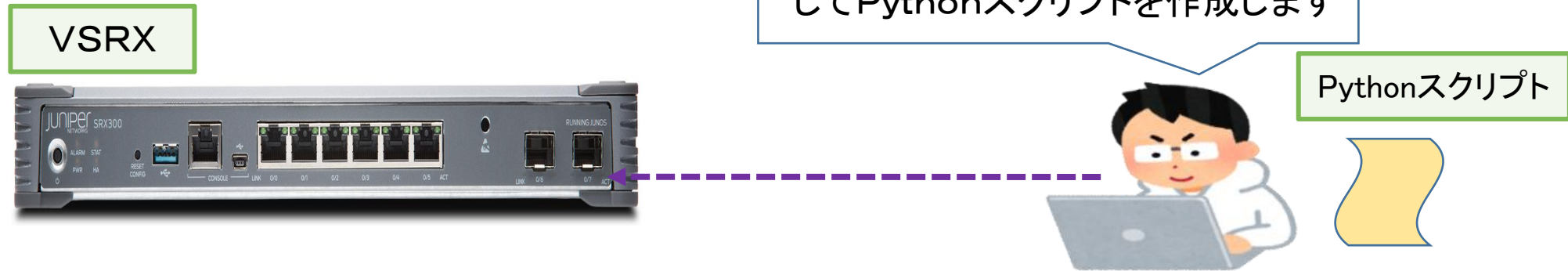
色々作成して  
動かしてみよう～

## 3 REST APIによるVSRXの管理

### (5) PythonスクリプトによるAPIアクセス

### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス



○ PythonスクリプトによりAPIへアクセスする条件は以下の通りです

- Pythonのバージョンは3. X系を使用しましょう
- requestsモジュールを使用するのでpipによりインストールしましょう

### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス

##### ○ Requestモジュールとは??

Requestsは、PythonのHTTP通信ライブラリです。

Requestsを使うとWebサイトの情報取得や画像の収集などを簡単に行えます。

Python には標準で urllib というライブラリが存在しますが、Requests はそれよりもシンプルに、人が直感的に分かりやすいプログラムを記述できます。

(Python3のpipインストール)

```
#python3 -m pip install pip
```

(pipによるrequestモジュールのインストール)

```
#pip install requests
```

### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス

【Ubuntu18】

##### ○ pipのインストール

```
$sudo△python3△-m△easy_install△pip
```

/pythonのパッケージ管理システムであるpipをインストールします

```
$pip3△install△requests
```

/requestsモジュールをpipでインストールします

```
$pip3 list
```

/requestsモジュールがインストールできたかどうかを確認します

### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス

【Ubuntu18】

```
$sudo python3 -m easy_install pip
```

/pythonのパッケージ管理システムであるpipをインストールします

```
gorosuke@ubuntu:~$ sudo python3 -m easy_install pip
```

```
[sudo] gorosuke のパスワード:
```

```
Searching for pip
```

```
Best match: pip 9.0.1
```

```
Adding pip 9.0.1 to easy-install.pth file
```

```
Installing pip script to /usr/local/bin
```

```
Installing pip3 script to /usr/local/bin
```

```
Installing pip3.6 script to /usr/local/bin
```

```
Using /usr/lib/python3/dist-packages
```

```
Processing dependencies for pip
```

```
Finished processing dependencies for pip
```



### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス

【Ubuntu18】

\$pip3 install requests

/requestsモジュールをpipによりインストールします

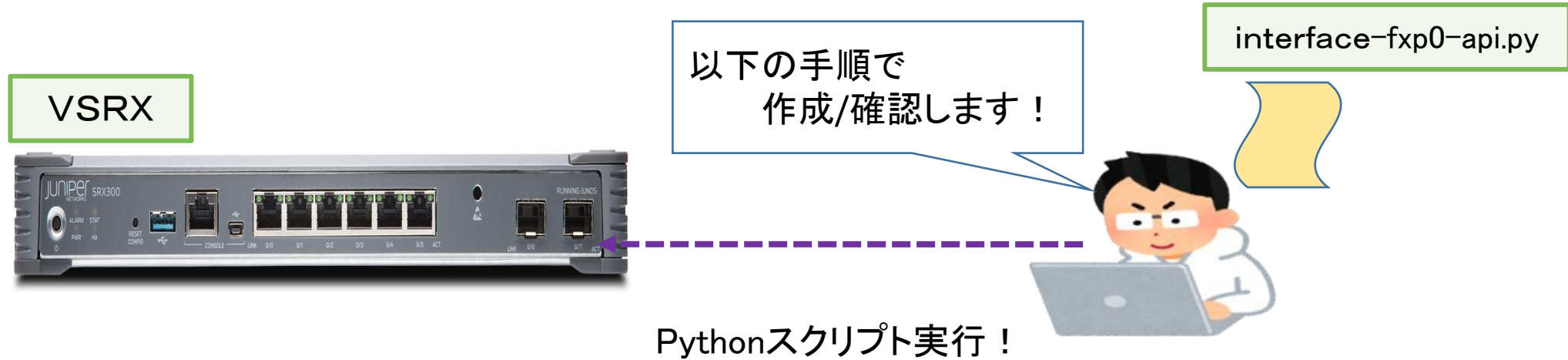
```
gorosuke@ubuntu:~$ pip3 install requests
Collecting requests
  Downloading https://files.pythonhosted.org/packages/2d/61/08076519c80041bc0ffa1a8af0cbd3bf3e2b62af10435d269a9d0f40564d/requests-2.27.1-py2.py3-none-any.whl (63kB)
    100% |████████████████████████████████████████| 71kB 85kB/s
Collecting charset-normalizer~=2.0.0; python_version >= "3" (from requests)
  Downloading https://files.pythonhosted.org/packages/06/b3/24afc8868eba069a7f03650ac750a778862dc34941a4bebeb58706715726/charset_normalizer-2.0.12-py3-none-any.whl
Collecting urllib3<1.27,>=1.21.1 (from requests)
  Downloading https://files.pythonhosted.org/packages/ec/03/062e6444ce4baf1eac17a6a0ebfe36bb1ad05e1df0e20b110de59c278498/urllib3-1.26.9-py2.py3-none-any.whl (138kB)
    100% |████████████████████████████████████████| 143kB 119kB/s
Collecting idna<4,>=2.5; python_version >= "3" (from requests)
  Downloading https://files.pythonhosted.org/packages/04/a2/d918dcd22354d8958fe113e1a3630137e0fc8b44859ade3063982eacd2a4/idna-3.3-py3-none-any.whl (61kB)
    100% |████████████████████████████████████████| 61kB 198kB/s
Collecting certifi>=2017.4.17 (from requests)
  Downloading https://files.pythonhosted.org/packages/37/45/946c02767aabb873146011e665728b680884cd8fe70dde973c640e45b775/certifi-2021.10.8-py2.py3-none-any.whl (149kB)
    100% |████████████████████████████████████████| 153kB 276kB/s
Installing collected packages: charset-normalizer, urllib3, idna, certifi, requests
Successfully installed certifi-2021.10.8 charset-normalizer-2.0.12 idna-3.3 requests-2.27.1 urllib3-1.26.9
```



### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス

- fxp0.0の状態をスクリプトで確認してみましょう！



- junos上でRPCでの出力形式を確認します
- APIエクスプローラを使用して細部を確認します
- Pythonスクリプト(interface-fxp0-api.py)を作成します。
- Pythonスクリプトを実行し、結果を確認します。

### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス

- junos上でRPCでの出力形式を確認します

```
admin@VSRX> show interfaces fxp0.0 | display xml rpc
<rpc-reply xmlns:junos="http://xml.juniper.net/junos/15.1X49/junos">
  <rpc>
    <get-interface-information>
      <interface-name>fxp0.0</interface-name>
    </get-interface-information>
  </rpc>
  <cli>
    <banner></banner>
  </cli>
</rpc-reply>

admin@VSRX> □
```

### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス

- Pythonスクリプト(interface-fxp0-api.py)を作成します。

スクリプト名 : interface-fxp0-api.py

```
import requests
import pprint
```

get\_jsonという関数を定義

```
def get_json(interface):
```

```
    source = 'http://130.230.0.254:8080/rpc'
    method = '/get-interface-information'
```

urlへの部品を定義

```
    param = '/interface-name=' + interface
```

アクセスするurlを定義

```
    options = ''
```

```
    url = source + method + param + options
```

json形式でアクセス

```
    headers = {'Content-type' : 'application/json' , 'Accept' : 'application/json' }
```

```
    r = requests.get(url,auth=('root', 'root123'),headers=headers)
```

上記urlに対して  
getによりアクセスする

```
    return r.json()
```

```
pprint.pprint(get_json('fxp0.0'))
```

get\_jsonの結果を出力する

### 3 REST APIによるVSRXの管理

#### (5) PythonスクリプトによるAPIアクセス

- Pythonスクリプト(interface-fxp0-api.py)を実行し、結果を確認します

```
gorosuke@ubuntu:~/python3$ python3 interface-fxp0-api.py
```

```
{'interface-information': [{'attributes': {'junos:style': 'normal',  
                                          'xmlns': 'http://xml.juniper.net/junos/15.1X49/junos-interface'},  
  'logical-interface': [{'address-family': [{'address-family-flags': [{'iff-is-primary': [{'data': None}],  
                                                                    'iff-sendbroadcast-pkt-to-re': [{'data': None}]}],  
    'address-family-name': [{'data': 'inet'}],  
    'interface-address': [{'ifa-broadcast': [{'data': '192.168.91.255'}],  
                          'ifa-destination': [{'data': '192.168.91/24'}],  
                          'ifa-flags': [{'ifaf-current-default': [{'data': None}],  
                                        'ifaf-current-preferred': [{'data': None}],  
                                        'ifaf-current-primary': [{'data': None}]}],  
                          'ifa-local': [{'data': '192.168.91.132'}]}],  
    'mtu': [{'data': '1500'}]},  
  'encapsulation': [{'data': 'ENET2'}],  
  'filter-information': [{}],  
  'if-config-flags': [{'iff-snmp-traps': [{'data': None}],  
                      'iff-up': [{'data': None}],  
                      'internal-flags': [{'data': '0x4000000'}]}],  
  'local-index': [{'data': '4'}],  
  'name': [{'data': 'fxp0.0'}],  
  'snmp-index': [{'data': '13'}],  
  'traffic-statistics': [{'attributes': {'junos:style': 'brief'},  
    'input-packets': [{'data': '481'}],  
    'output-packets': [{'data': '304'}]}]}
```

## 4 Netmikoモジュール によるNW管理

(1) Netmikoとは??

## 4 NetmikoモジュールによるNW管理

### ○ Netmikoとは？？

Netmikoとはネットワークデバイスとかにsshやtelnetで接続してなんやかやをやってくれるpythonのライブラリの一つ

<https://github.com/ktbyers/netmiko>

- SSH接続が基本。一部機種で Telnet接続、シリアルポート接続にも対応。
- Cisco, Juniper, Palo Alto, Dell, Linuxなどマルチベンダに対応。
- pipでインストール可能: `pip install netmiko`
- 2020年1月24日時点の最新バージョンはVersion 3.0.0

## 4 NetmikoモジュールによるNW管理

## ○ Netmikoモジュールのインストール及び確認

## 【Ubuntu 18】

## /アップデート及びモジュールのインストール

```
$ sudo apt-get update
```

```
$ sudo apt-get install python3-netmiko
```



インストール後  
dpkgコマンドにより確認！

```
gorosuke@ubuntu:~$ sudo dpkg-query -f='${Package} ${Version} ${Architecture} ${Description}\n' | grep netmiko
ii python-netmiko 1.4.3-1
    all multi-vendor library for SSH connections to network devices - Python 2.X
ii python3-netmiko 1.4.3-1
    all multi-vendor library for SSH connections to network devices - Python 3.X
gorosuke@ubuntu:~$
```



## 4 Netmikoモジュール によるNW管理

### (2) VSRXへの事前設定



## 4 NetmikoモジュールによるNW管理

### (2) VSRXの事前設定

- SSHアクセスできるようにしましょう！

VSRX



SSHでアクセスできるように  
事前設定します



- Netmikoは基本SSH接続が必要であるため設定しておきます  
`root#set system services ssh`

- 設定後、SRXに対してSSH接続可能かどうかを確認します。  
Windowsであればターミナルソフト(Tereterm)  
LinuxであればSSHコマンドにより確認します。

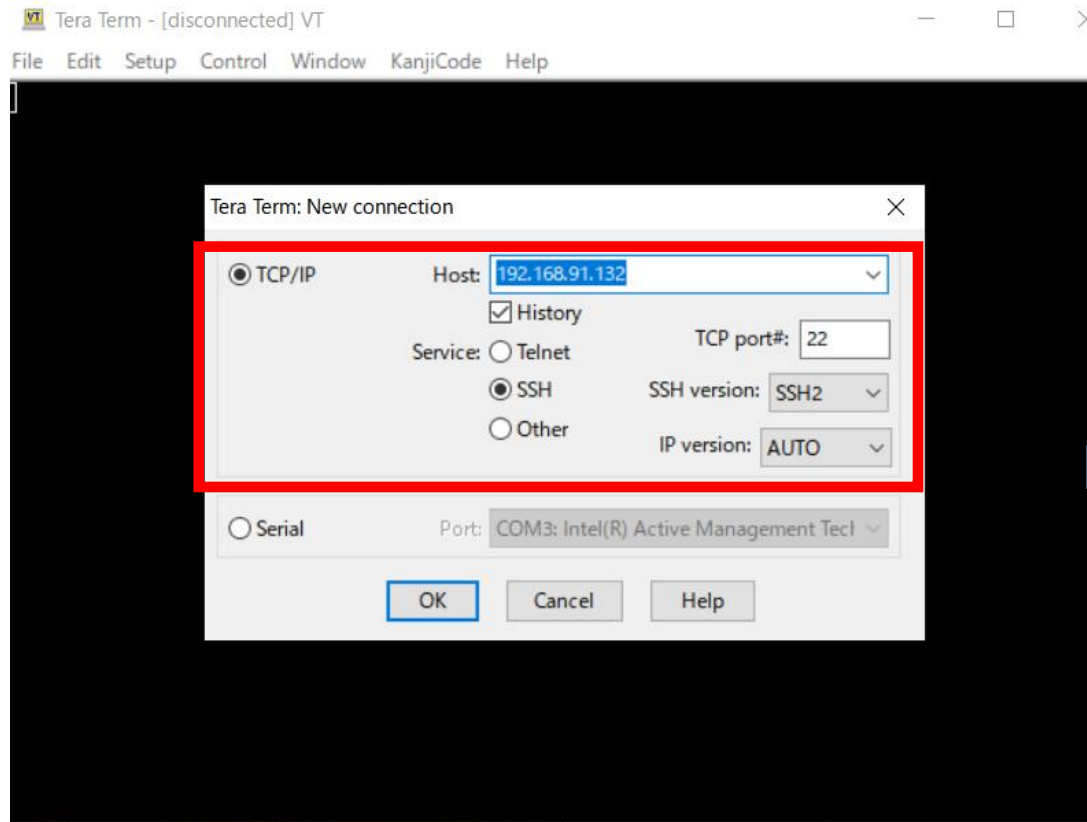
## 4 NetmikoモジュールによるNW管理

### (2) VSRXの事前設定

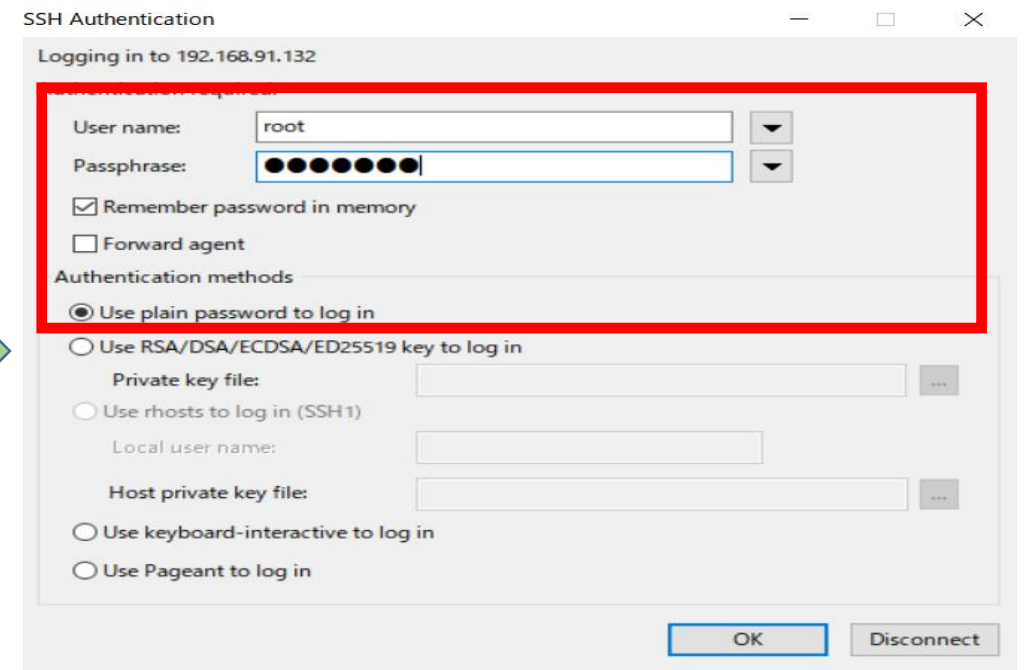
○ SSHアクセスできるようにしましょう！

TeraTermを用いたアクセス例

【IPアドレスおよびSSHを指定】



【ユーザ/パスワードを指定】



## 4 Netmikoモジュール によるNW管理

### (3) スクリプト作成及び確認

## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

- show version情報を取得するスクリプトを作成及び確認しましょう！

VSRX



以下の手順で  
作成/確認します！

show-version-netmiko.py

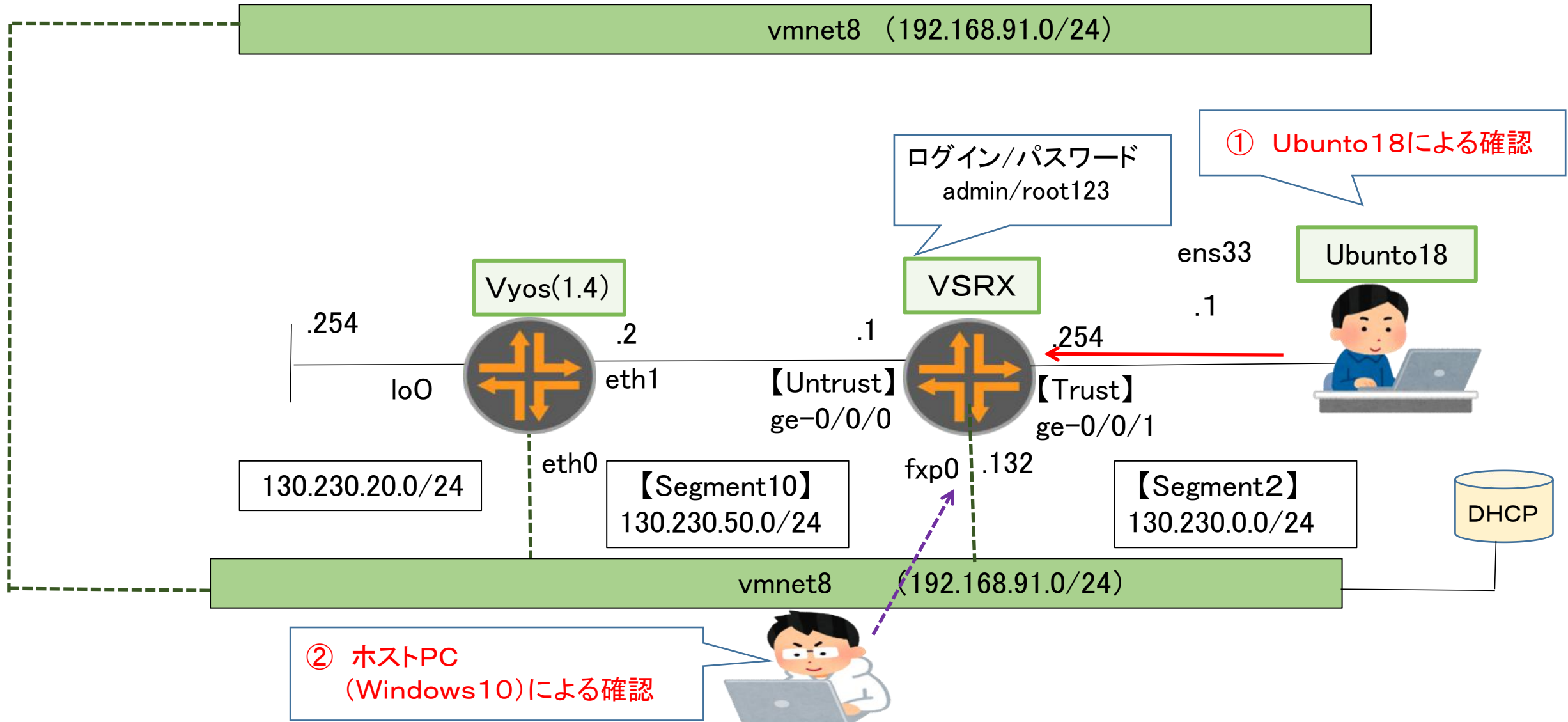


Pythonスクリプト実行！

- Netmikoを使用してスクリプトを作成します
- スクリプトを起動します
- 結果を確認します

## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認



## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

show-version-netmiko.py

```
GNU nano 2.9.3 show-version-netmiko.py

#!/usr/bin/env python3
from netmiko import ConnectHandler

remote_device = {
    'device_type' : 'juniper',
    'ip' : '130.230.0.254' ,
    'username' : 'admin',
    'password' : 'root123',
}

net_connect = ConnectHandler(**remote_device)
output = net_connect.send_command('show version')
print(output)
```

対象のデバイスを定義

対象デバイスと接続  
コマンドを発行  
結果を表示

## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

#### Onet\_connectで利用できる関数の例

`net_connect.send_command()`

–この関数は、コマンドをネットワークチャネルに送信し、パターンに基づいて出力を返します。

`net_connect.send_command_timing()`

–ネットワークチャネルに送信されたコマンドからのタイミングに基づいて出力を返します。

`net_connect.send_config_set()`

–構成設定をリモートデバイスに適用します。

`net_connect.send_config_from_file()`

–外部ファイルから構成設定を適用します

`net_connect.save_config()`

–実行コンフィギュレーションをスタートアップコンフィギュレーションとしてエクスポートおよび保存します。

`net_connect.enable()`

–デバイスにクエリを実行してイネーブルモードをアクティブにします。

`net_connect.find_prompt()`

–現在のルータプロンプトを返します

`net_connect.commit()`

–JuniperやIOS-XRなどのデバイスでコミットコマンドを実行します

`net_connect.disconnect()`

–セッションを終了します

## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

show-version-netmiko.py

```
gorosuke@ubuntu:~/python3$ python3 show-version-netmiko.py  
  
Hostname: VSRX  
Model: vsrx  
Junos: 15.1X49-D140.2  
JUNOS Software Release [15.1X49-D140.2]  
  
gorosuke@ubuntu:~/python3$
```

スクリプトを実行  
→ 実行結果を確認できます！

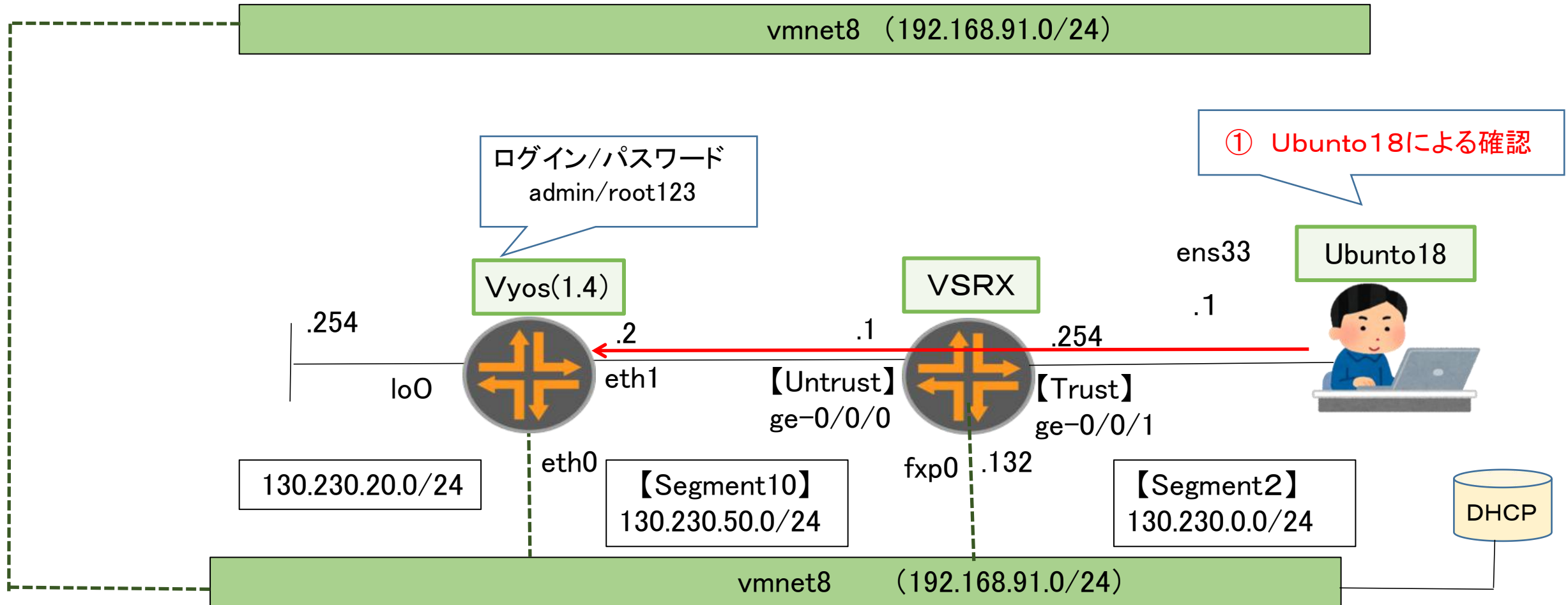




## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

Vyosにも適用出来ます！



## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

Vyosにも適用出来ます！

```
#!/usr/bin/env python3
from netmiko import ConnectHandler

remote_device = {
    'device_type' : 'vyos',
    'ip' : '130.230.50.2',
    'username' : 'vyos',
    'password' : 'vyos',
}

net_connect = ConnectHandler(**remote_device)
output = net_connect.send_command('show version')
print(output)
```

## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

Vyosにも適用出来ます！

```
gorosuke@ubuntu:~/python3$ python3 show-version-netmiko-vyos.py
```

```
Version:          VyOS 1.4-rolling-202203200929  
Release train:    sagitta
```

```
Built by:         autobuild@vyos.net  
Built on:         Sun 20 Mar 2022 09:29 UTC  
Build UUID:       5057e381-8f86-49c2-891f-fcf7b8f10209  
Build commit ID: 20fd8588efdc77
```

```
Architecture:     x86_64  
Boot via:         installed image  
System type:      VMware guest
```

```
Hardware vendor:   VMware, Inc.  
Hardware model:    VMware Virtual Platform  
Hardware S/N:      VMware-56 4d e7 ce 65 d4 7a 44-36 65 04 5b e7 7a 75 7a  
Hardware UUID:     cee74d56-d465-447a-3665-045be77a757a
```

```
Copyright:         VyOS maintainers and contributors
```

```
gorosuke@ubuntu:~/python3$
```

## 4 NetmikoモジュールによるNW管理

### (3) スクリプト作成及び確認

○ コード(test.py)

Windowsにおける例

test.py - C:\Users\ユーザー\test.py (3.7.1)

File Edit Format Run Options Window Help

```
from netmiko import ConnectHandler
```

```
juniper_junos = {  
    'device_type': 'juniper',  
    'ip': '192.168.91.130',  
    'username': 'admin',  
    'password': 'jisoku24',  
}
```

```
# global_delay_factor IS THE KEY :p
```

```
net_connect = ConnectHandler(**juniper_junos, global_delay_factor=2)
```

```
output = net_connect.send_command('show interfaces terse')
```

```
print(output)
```

## 4 NetmikoによるVSRXの管理

### (2) VSRXへの適用

#### ○ コード(test.py)実行例

【スクリプト実行例】

Windowsにおける例

```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:57:15) [MSC v.1915 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\ユーザー\test.py =====
Interface Admin Link Proto Local Remote
ge-0/0/0 up up
ge-0/0/0.0 up up inet 130.230.50.2/24
sr-0/0/0 up up
ip-0/0/0 up up
lsq-0/0/0 up up
lt-0/0/0 up up
mt-0/0/0 up up
sp-0/0/0 up up
sp-0/0/0.0 up up inet
sp-0/0/0.16383 up up inet6
ge-0/0/1 up up inet
ge-0/0/1.0 up up inet 192.168.20.254/24
dsc up up
em0 up up
em0.0 up up inet 128.0.0.1/2
em1 up up inet 192.168.1.2/24
em1.32768 up up
em2 up up
fxp0 up up
fxp0.0 up up inet 192.168.91.130/24
gre up up
ipip up up
irb up up
lo0 up up
lo0.16384 up up inet 127.0.0.1 --> 0/0
lo0.16385 up up inet 10.0.0.1 --> 0/0
10.0.0.16 --> 0/0
128.0.0.1 --> 0/0
128.0.0.4 --> 0/0
128.0.1.16 --> 0/0
lo0.32768 up up
lsi up up
```

Ln: 50 Col: 4

5

參考資料

## 5 参考資料

○github上でJunos REST APIへのスクリプト例が公開されています！

[https://github.com/ksator/junos\\_automation\\_with\\_rest\\_calls](https://github.com/ksator/junos_automation_with_rest_calls)

The screenshot displays the GitHub interface for the repository `ksator/junos_automation_with_rest_calls`. At the top, a yellow banner states: "This repository has been archived by the owner. It is now read-only." Below this, the repository name is shown with a "Public archive" label. Navigation tabs include Code, Issues (1), Pull requests, Actions, Projects, Wiki, Security, and Insights. The file explorer shows the following structure:

File/Folder	Description	Commit Date
explorer	Add files via upload	4 years ago
google_map	Update google_map_api.py	4 years ago
group_vars/vMX	m	4 years ago
junos	m	4 years ago
LICENSE	add MIT licence using Gitpython	4 years ago
README.md	Fixed README.md typo	4 years ago
ansible.cfg	m	4 years ago
hosts	m	4 years ago

The right sidebar contains the "About" section with the title "How to make rest calls to Junos" and tags: `python`, `ansible`, `rest-api`, `junos`, and `junos-automation`. It also lists repository statistics: 10 stars, 4 watching, and 5 forks. The "Releases" section indicates "No releases published".