

API/Web化によるネットワーク自動化

~プログラミング未経験のNWエンジニアの話~

株式会社IDCフロンティア
R&D室 井上一清

2015年7月16日

Facebook、Google、Twitterなど、最近はやりのWebアプリケーションではAPIが当たり前のように実装されており、API設計がビジネスの勝敗を左右すると言っても過言ではない状態になっています。また、Webアプリケーションだけでなく、インフラにおいてもAPIの重要性は増してきていますが、ネットワークのプロビジョニングではAPI対応は遅れているように思われます。

今回のセッションでは、そもそもネットワークにおけるAPI対応とは何なのか、APIを用いてどのように自動化を行うのか、どのような設計が必要で、必要なスキルは何なのか、課題は何か、といった部分を事例も交えて発表、議論させていただければと思います。

- 世の中でのAPIの使われ方
- ネットワークのAPI対応とは
- 事例 (Web/API入力でルータに設定を投入)
- ネットワークのAPI対応のためのソフトウェア設計
- 使用したツール (Sinatra, MySQL, ActiveRecord, NETCONF, JavaScript, jQueryなど)
- 必要なスキル、課題

Web APIで様々なサービスが連携されている

何ができるのか

複数サービスの連携。サービスを広く使わせる

ex) 自社システムとの連携、3rd Partyとの連携

IT系で伸びている会社は全てAPIを活用している



APIがビジネスの
勝敗を左右する！



```
</script>
<!-- BEGIN: WP Social Bookmarking Light -->
<script>(function(d, s, id) {
  var js, fjs = d.getElementsByTagName(s)[0];
  if (d.getElementById(id)) return;
  js = d.createElement(s); js.id = id;
  js.src = "//connect.facebook.net/ja_JP/sdk.js#xfbml=1&version=v2.0";
  fjs.parentNode.insertBefore(js, fjs);
}(document, 'script', 'facebook-jssdk'));</script>
```

IDCFサイトの「いいね」のJavascript

<https://developers.facebook.com/docs/graph-api/reference> より引用

WebサイトにFacebookのJavascriptを仕込んで、
class="facebook-icon" とかの属性を作るだけ

今や世界に何十億、何百億というサイトに「いいね」ボタンがある

Slack自体はただのチャットツール

APIを使った外部プログラムとの様々な
インテグレーション方法が用意されている。

- Roominoと連携したホテル予約、
- Twilio、Watsonと組み合わせた音声⇔テキストチャット
- Microsoft Translatorと連携したリアルタイム翻訳チャット
- Nagiosと連携させたSlackでの監視通知

会場限り

会場限り

SlackのAPIは非常にキレイで
使いやすらしい

会場限り

会場限り

会場限り

色々なアプリを組み合わせ
てユーザ独自の使い方を提供
※使われる側のアプリもAPI必須

APIを介して他システムと連携させることで
エコシステムを生み出している

アプリケーションの世界では
APIなしのサービスというのは考えられない
API/SDK設計を最初から考えて作っている

インフラの世界ではどうか？

VMの例

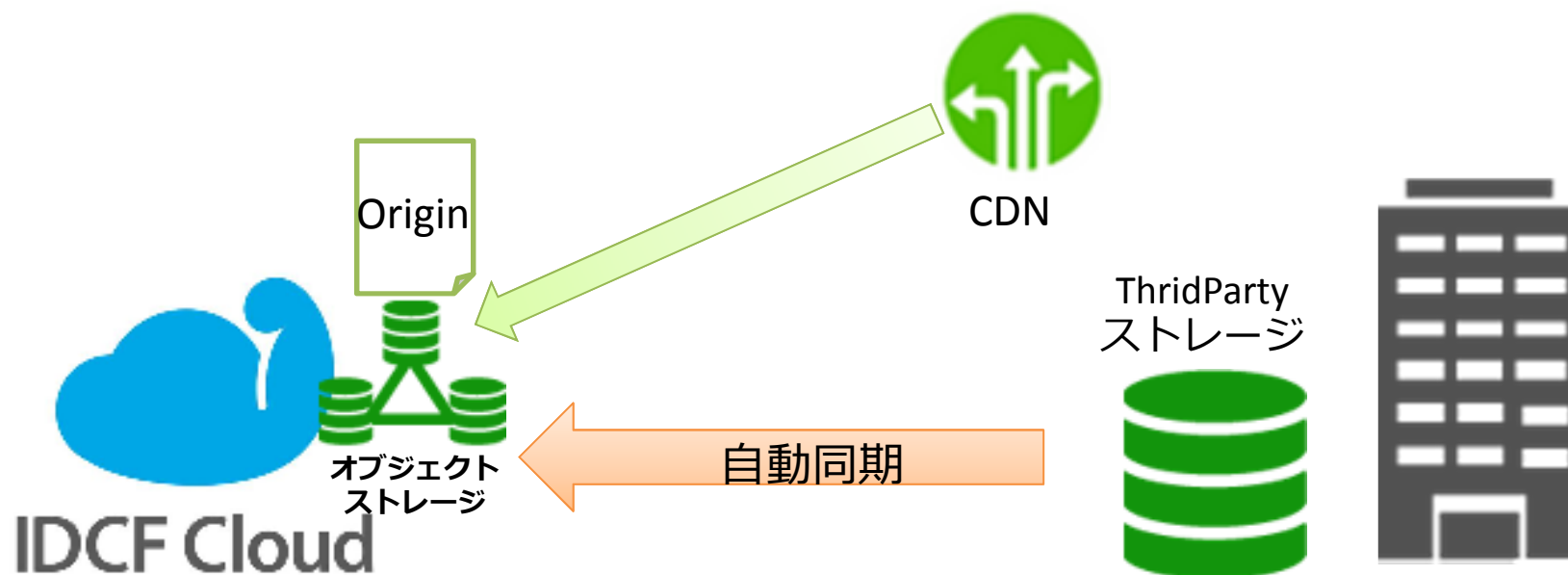
- 負荷に応じたサーバのAutoScale
- 定期バックアップ (snapshot)
- 構成管理、バージョン管理
- 本番環境と同等の環境を即座に構築



APIというよりスクリプト的なものもありますが。
まあスクリプトもAPIも同じ用なもの(´・ω・`)

ストレージの例

企業用ストレージとクラウドストレージの連携



ネットワークの連携って・・・??

そもそも要らないのか・・・??

いや、そんなことはない！
ネットワークも自動化とかしたい

• Address – アドレスに関するメソッド

- [disassociateIpAddress](#)
- [associateIpAddress](#)
- [listPublicIpAddresses](#)

• Firewall – ファイアウォールに関するメソッド

- [listPortForwardingRules](#)
- [updatePortForwardingRule](#)
- [createPortForwardingRule](#)
- [deletePortForwardingRule](#)
- [createFirewallRule](#)
- [deleteFirewallRule](#)
- [listFirewallRules](#)
- [createEgressFirewallRule](#)
- [deleteEgressFirewallRule](#)
- [listEgressFirewallRules](#)

FWとかLBとかはそこそこある。
(どちらかというとならサーバより)

• Load Balancer – ロードバランサーに関するメソッド

- [assignToLoadBalancerRule](#)
- [removeFromLoadBalancerRule](#)
- [createLoadBalancerRule](#)
- [createLBStickinessPolicy](#)
- [deleteLoadBalancerRule](#)
- [deleteLBStickinessPolicy](#)
- [listLoadBalancerRules](#)
- [listLBStickinessPolicies](#)
- [listLoadBalancerRuleInstances](#)
- [updateLoadBalancerRule](#)

• NAT – NATに関するメソッド

- [enableStaticNat](#)

• VPN – VPNに関するメソッド

- [createRemoteAccessVpn](#)
- [deleteRemoteAccessVpn](#)
- [listRemoteAccessVpns](#)
- [removeVpnUser](#)
- [addVpnUser](#)
- [listVpnUsers](#)

L2/L3周りはAPIがあんまりないヾ(°Д°)ノ

- 台数がサーバほど多くない
 - ネットワークは「組み合わせ」で使うものなので、汎用スクリプト化しづらい
 - OSが多岐にわたる
 - 標準的な外部APIがない
 - 手作業でも何とかなる。あるいは手作業の方が早い。
 - 何となく手作業の方が安心。
 - 何故かプログラミングのできるネットワークエンジニアが少ない
 - 今さら勉強するの面倒くさい
- etc、 etc、 etc ・ ・

やったるう！

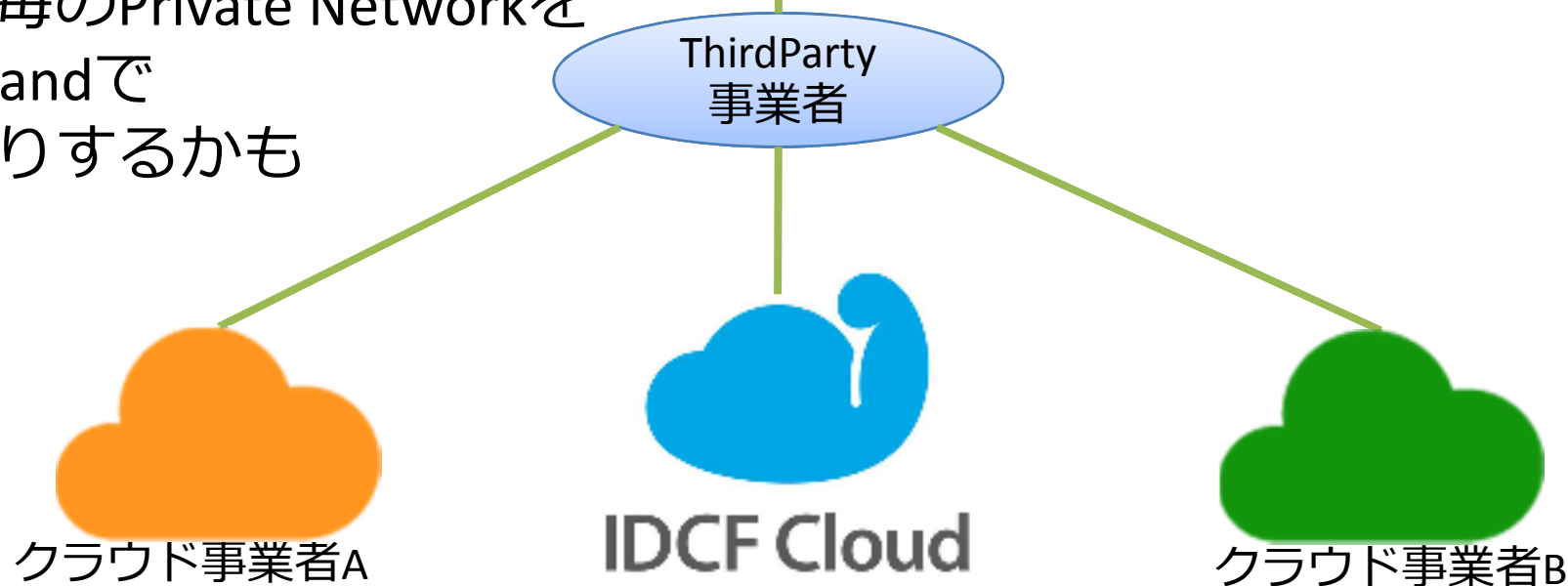
でもネットワークのAPI、
ネットワークの連携で
何ができるんだ？

> もしもネットワークが自動化できたら・・・？

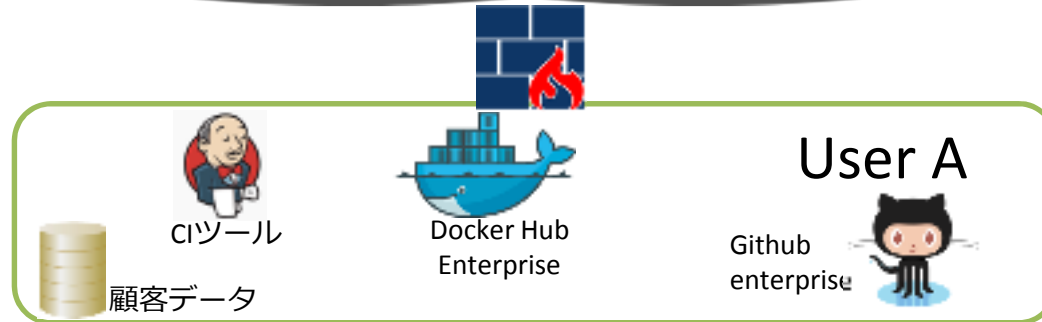
13



クラウドを跨いで、
ユーザ毎のPrivate Networkを
OnDemandで
作れたりするかも

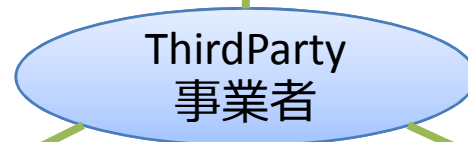


> 例えば・・開発環境の場合



クラウドを跨いで、
環境毎のPrivate Networkを
OnDemandで
作れたりするかも

コードや顧客データは自社で
機密な情報は自社で



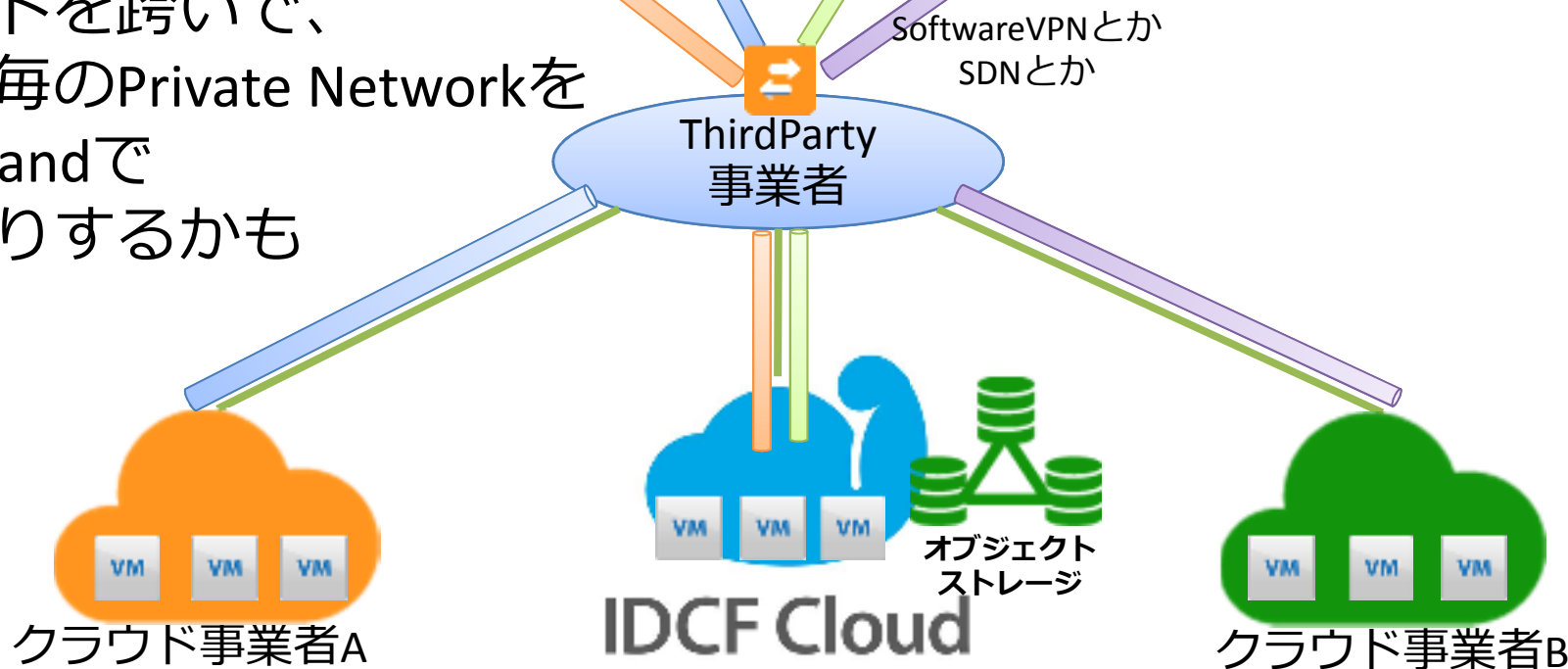
環境毎に仮想ネットワークを作成



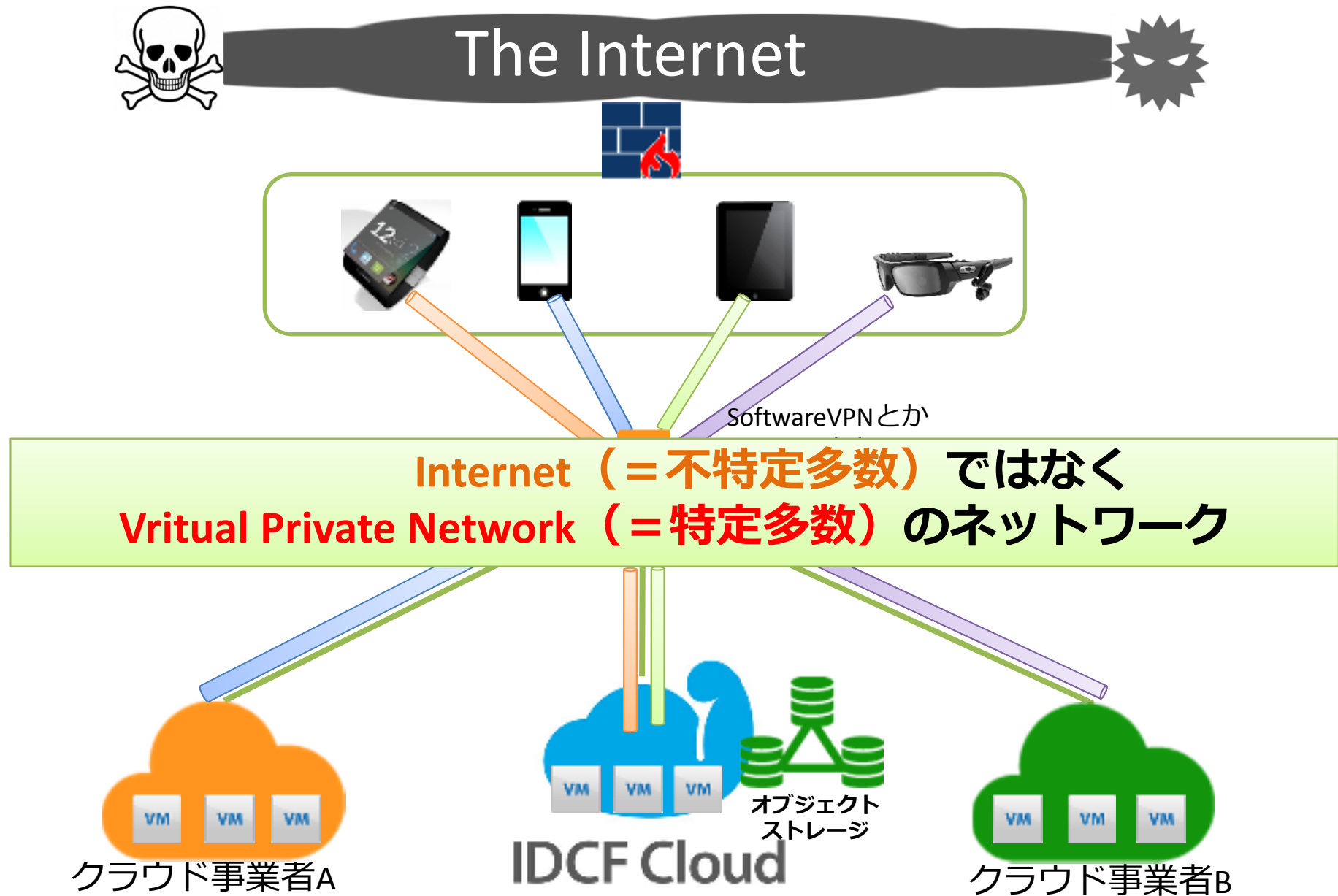
> スマホ、IoT向けの専用仮想ネットワーク



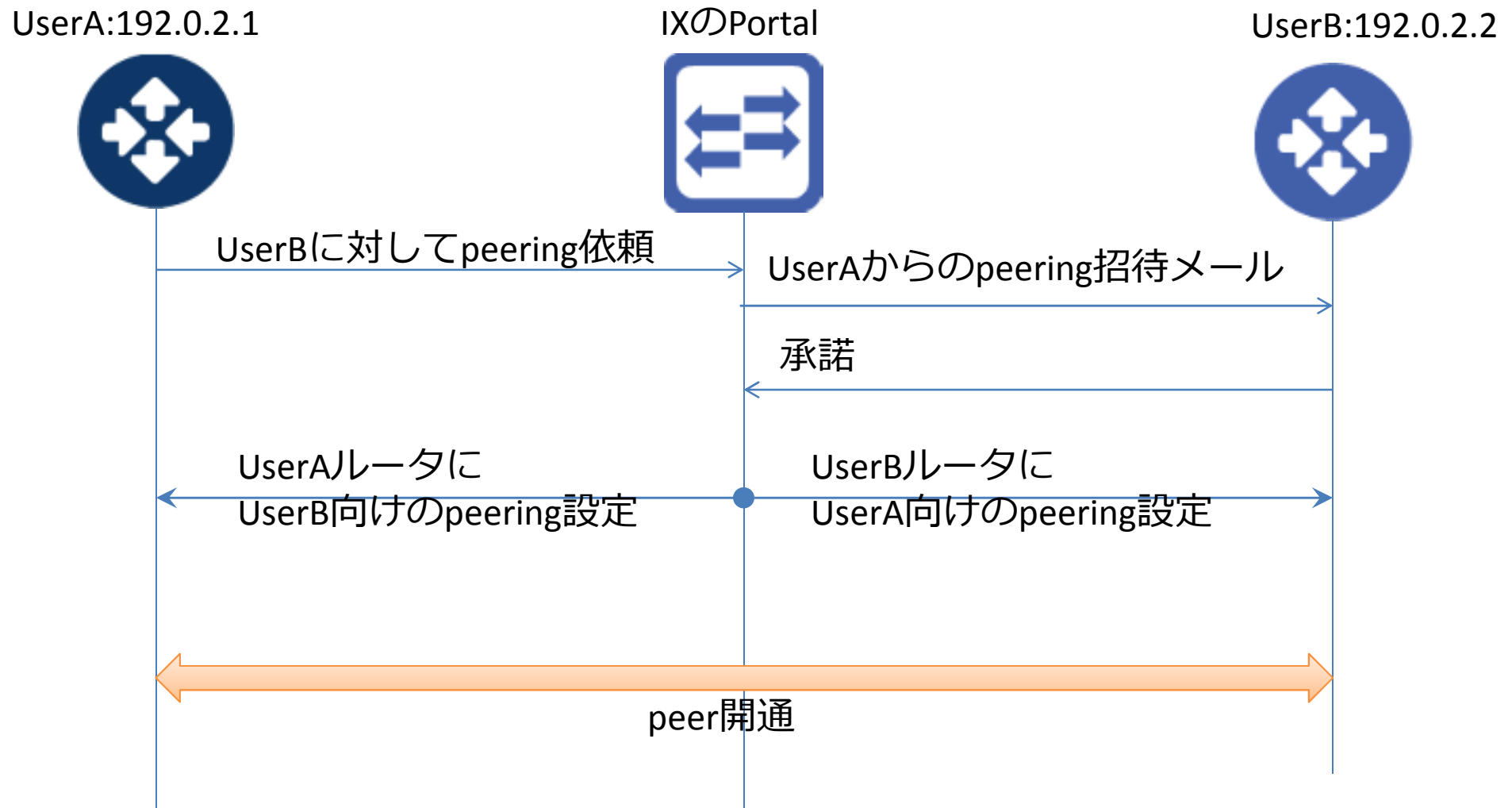
クラウドを跨いで、
アプリ毎のPrivate Networkを
OnDemandで
作れたりするかも



> スマホ、IoT向けの専用仮想ネットワーク



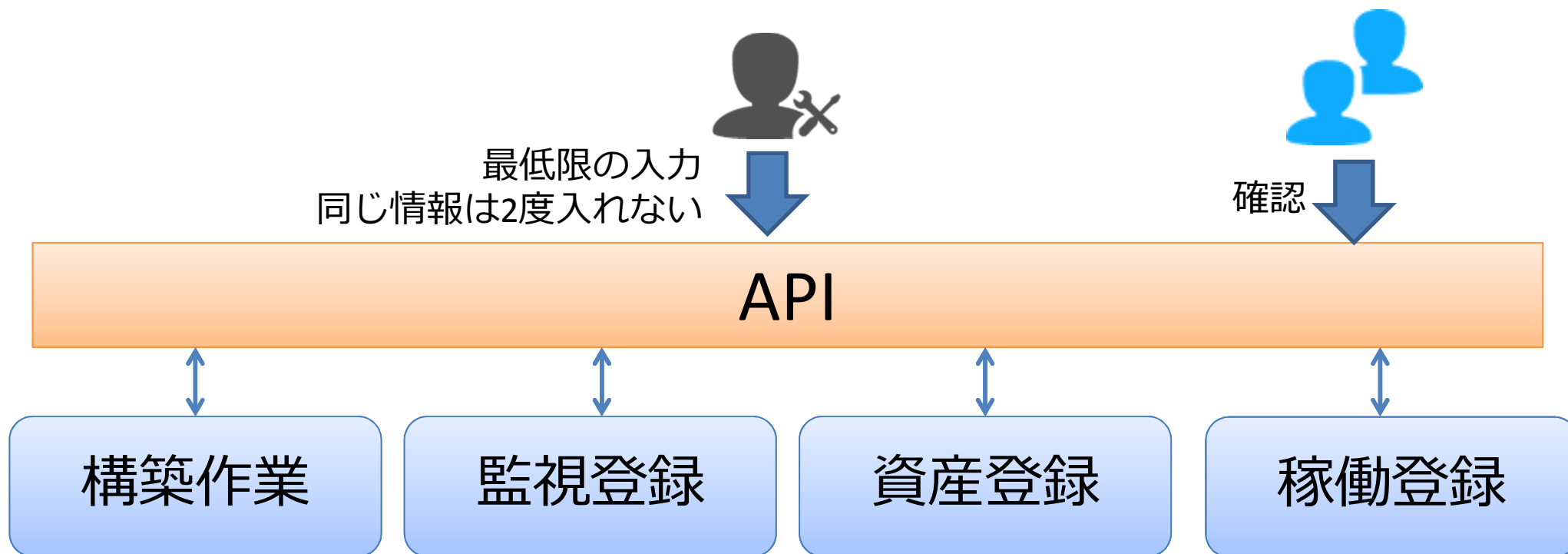
> こんなのもアリ？ Peeringの自動化



Peering設定もIX事業者にお任せすることもできる??
※セキュリティの話や大人の話は目をつぶります。

ネットワークでも自動化の世界、
色々なモノが連携する世界
を創ってあげたら面白そう！！

ルーチン的な業務、人力が面倒な業務も自動化(効率化)できるのでは



必ずしもAPI使えば良いというわけではないが、
自動化したいものは沢山ある

そもそもネットワークのAPIって何？

- ネットワーク機器に対してのAPI
 - ✓ CLI/GUIでも設定できるが、APIを用いてProgrammableに設定し易くするもの
- 事業者等のネットワークサービスに対してのAPI
 - ✓ IaaS等のクラウドネットワークをエンドユーザがAPIを用いて設定すること
 - ✓ ex) AWSのCreateVpc、AcceptVpcPeeringConnection とか

前者はメーカーが作るもの、後者は事業者が作るもの

そもそも自動化って何？ Software使うこと??

- SDN
 - 物理ネットワークの上に仮想ネットワークを作ること
- NFV
 - x86サーバ上でNW機能を実装すること
- ホワイต์ボックス
 - HWとSWを分離したスイッチ
 - Ansible/Chefとかの構成管理ツールを使って管理
- 数百台のスイッチをまとめて自動構築（ZTP）
- 設定作業の自動化（AutoProvisioning）

※SDNやNFVもAutoProvisioningをし易くするものではあるが、ここでは別ものとして定義

今回はこれがメイン

自動化、AutoProvisioningとは



作業をプログラム化すること



手順を漏れなく、ロジック化すること
曖昧さは受け入れられない。厳格さが必要
(とはいえ、個別対応ができる柔軟さも重要)

1. 作業手順の標準化

- 構成の標準化
- ポリシーの統一化
- Configの機械的な生成

標準化なくして自動化なし(°Д°)ゞ
イレギュラーなもの曖昧なものは
Code化しづらい

2. ツールによる設定の簡素化

- library等の各種ツール
- メーカー、機種、Version等の相違を吸収

Bash、NetConf、APIなどでの設定
libraryがあると嬉しい

3. API化

- パラメータをhttpで入力し設定を投入
- JSON等で変数渡せるようなAPIも必要

各動作とAPIを関連づける
Rubyでもpythonでもphpでも何でも良い

4. 他システムとの連携

- アカウント情報との連携
- 認証、課金
- ユーザ向けAPIの統合

Job化
Workerでのスケジューリング

> 作業のプログラム化

クライアント




HTTP
Rest API

命令はAPI
情報はDB



人がやっていた部分を
ロジック化して、プログラム
に落とし込んでいく。

Sinatra
 **Ruby**

MySQL
DB

Job登録
未実行
Job確認

Jobs Setting devices

ここら辺のプロビツール
は結構あると思う

Workers

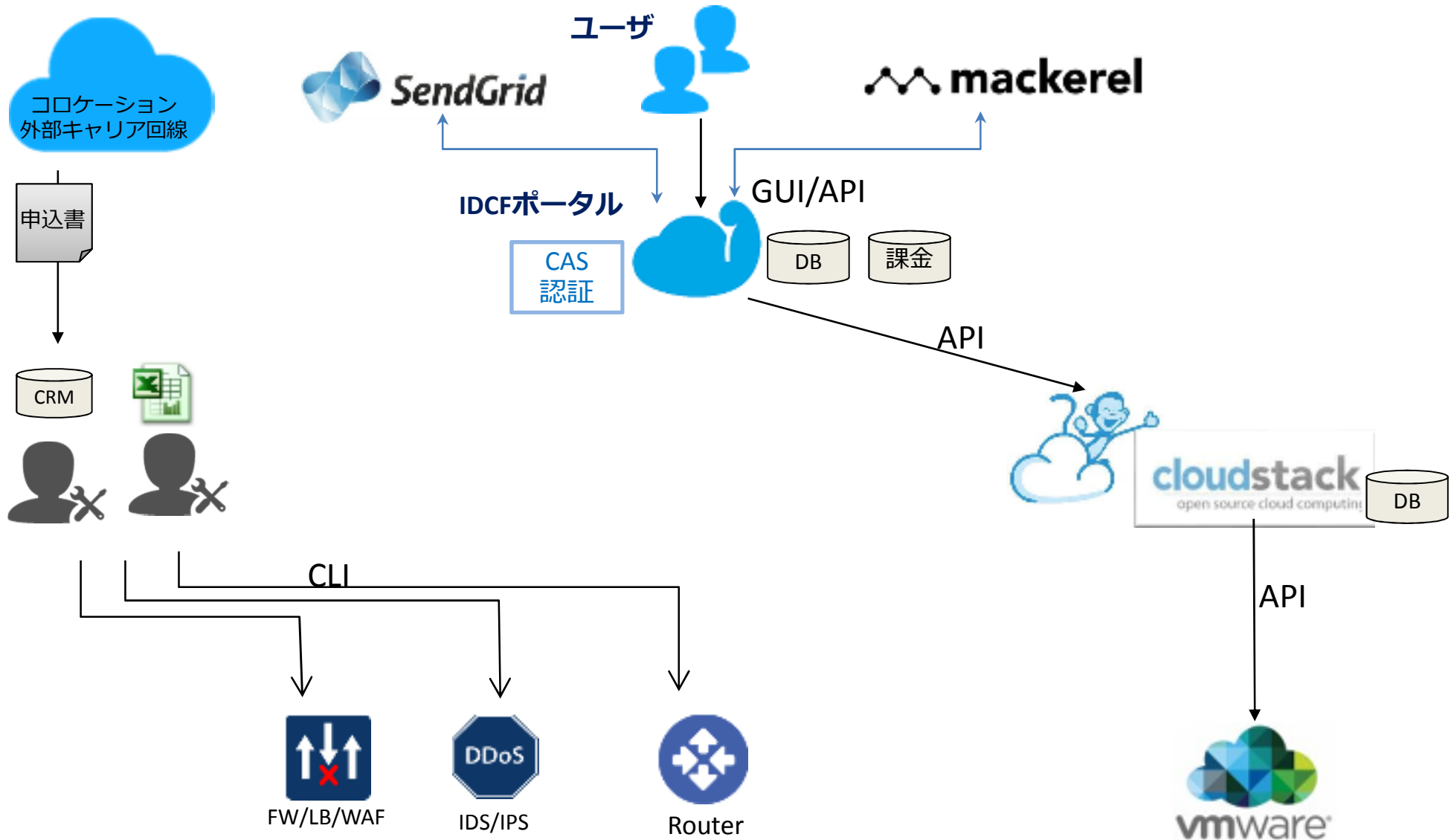


Worker Netconf Worker Netconf Worker Netconf

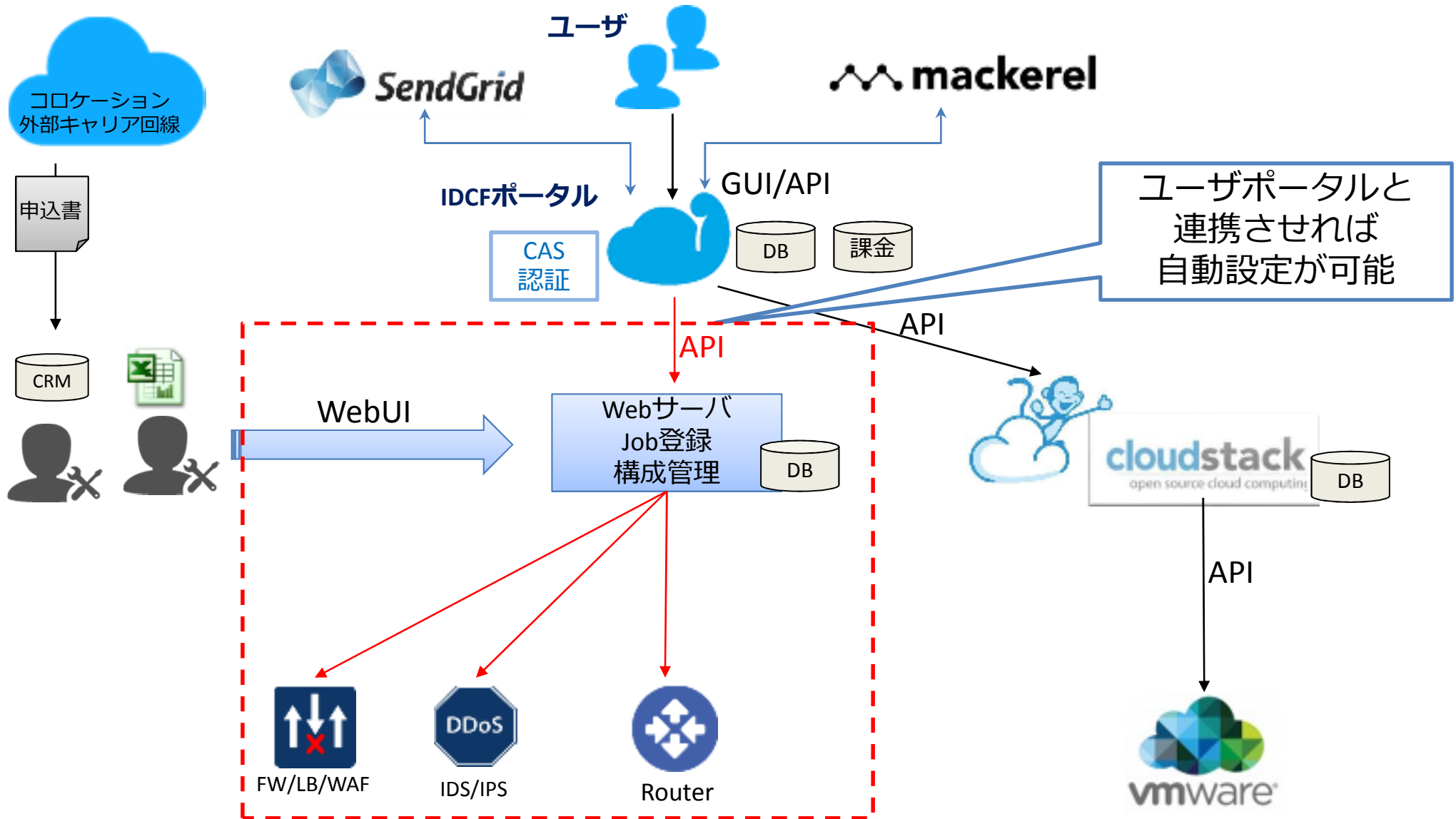
設定反映

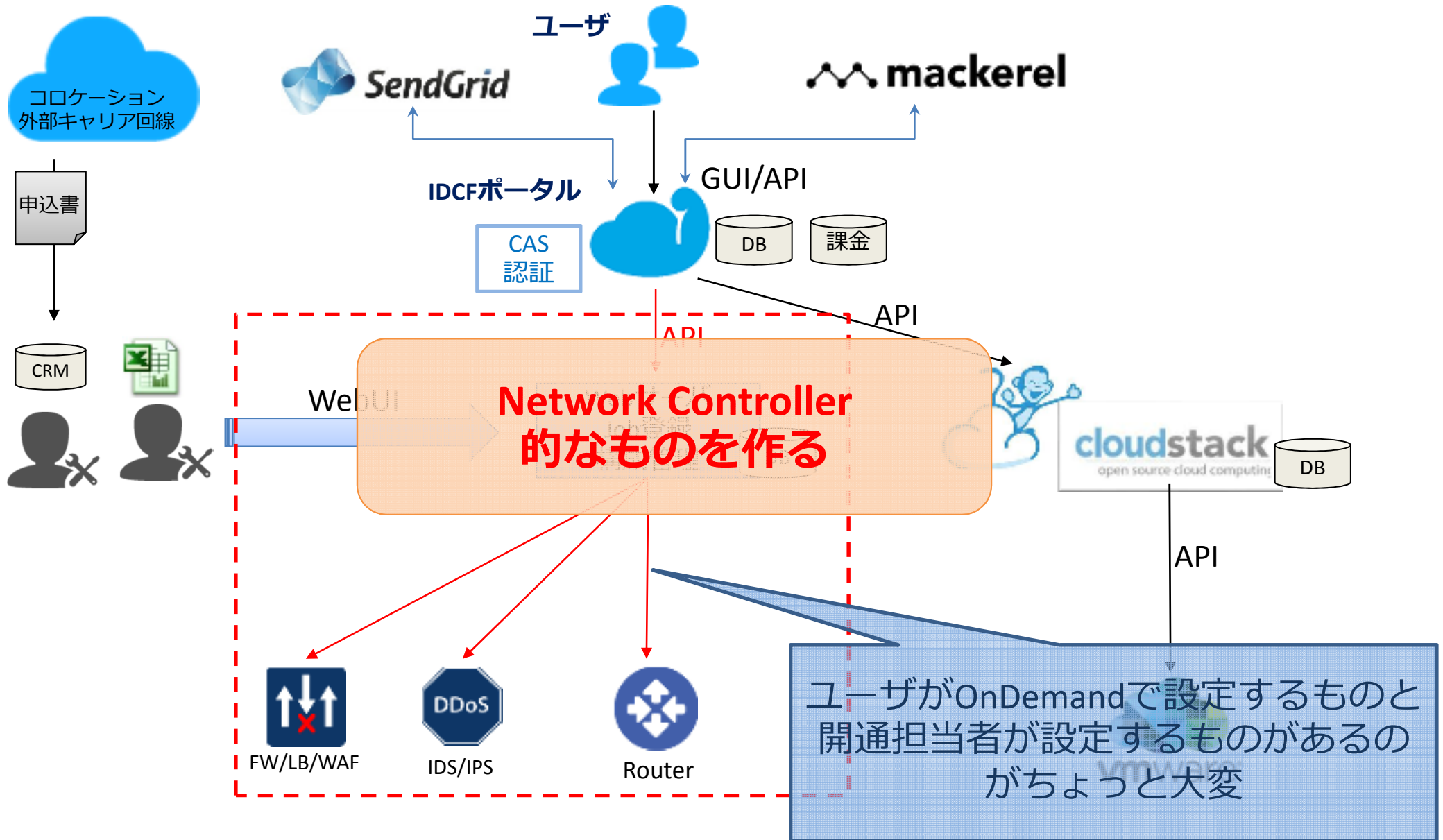
ルーター

> 連携のイメージ



> 連携のイメージ





ということで作ってみました。

言語	Webサーバ Webフレームワーク	DB O/Rマッピング	構成管理 プロビツール	Job管理
<ul style="list-style-type: none">• Ruby• Python• Go• PHP• Bash•••	<ul style="list-style-type: none">• Apache• Nginx• Webrick• Rack• WSGI• PSGI• Sinatra• Flask• Django• JavaScript• jQuery• Ajax	<ul style="list-style-type: none">• MySQL• ActiveRecord• MongoDB• Drizzle• MariaDB• Percona• Redis•	<ul style="list-style-type: none">• Ansible• Puppet• Chef• Fabric• Netconf• Bash• API•••	<ul style="list-style-type: none">• RabbitMQ• Celery• rundeck• cron• (Bash)• (MySQL)• Web UI

※推奨というわけではないです。極論ツールは何でも良いと思います。

各機能毎にAPIとClass/Methodを用意

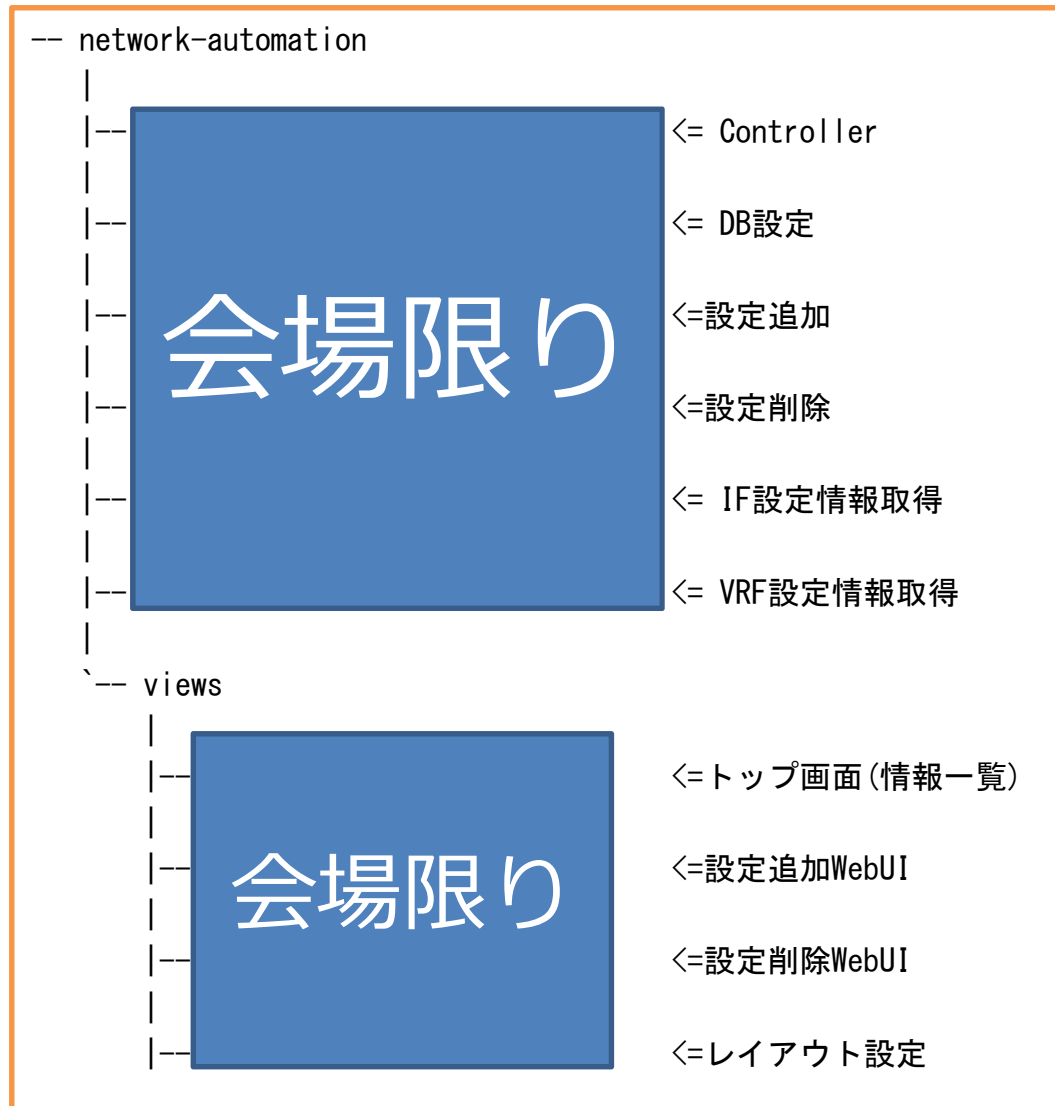
	Method	API Path	Argument	返回值
新規レコード追加	会場限り			
VPN設定				
設定対象VRFの情報取得				
削除対象IFの情報取得				
レコード削除				
VPN設定削除				
最新レコード10件分をJSON形式で取得				

複数の単語をつなげるのもいくつか表記方法がある。

	スパイナルケース spinal-case	スネークケース snake-case	キャメルケース camel-case	ドット dot
つなぎかた	- (ハイフン)	_ (アンダースコア)	大文字	ドット
例	/v1/idcf-inoue	/v1/idcf_inoue	/v1/idcfInoue	/v1/idcf.inoue
主な採用企業	Google, LinkedIn Facebook	Twitter Instagram slack	YouTube CloudStack OpenStack	Facebook ?
その他	SEO的には良いらしい	最近のはやり？	JavaScript、jQuery、 Rubyはこれが多い	あんまり見ない

APIは見易さも重要。表記方法もできれば統一してほしいが非常に難しいでしょうが。。

各ファイルの構造と役割



Class/Method単位で
ファイル化した

やり方は人それぞれ

一覧情報取得 (Topページ)

The screenshot shows the 'プライベートコネクト' (Private Connect) page. It features a search bar, a dropdown for '全てのアカウント', and buttons for '20件', '100件', and 'CSV'. Below is a table with columns: 番号, ドメイン, アカウント, ユーザ名, Vlan, VRF, 帯域, ソーン, セグメント, 取得日, 更新日.

番号	ドメイン	アカウント	ユーザ名	Vlan	VRF	帯域	ソーン	セグメント	取得日	更新日
NPV00001	70000000001	inoue0402	issei0402	10	100	1000	Tesla	192.168.1.0/24	2015-04-08 11:40:07 UTC	2015-04-08 11:40:07 UTC
NPV00002	70000000002	inoue0402	issei0402	20	100	1000	Henry	192.168.2.0/24	2015-04-08 11:40:15 UTC	2015-04-08 11:40:15 UTC
NPV00008	70000000008	inoue0408	issei0408	800	108	1000	Henry	192.168.8.0/24	2015-04-22 09:59:45 UTC	2015-04-22 10:01:16 UTC

実はここが一番大変。。

設定追加

Add PrivateConnect

The screenshot shows the '新規レコード追加' (Add New Record) form. It has input fields for '番号', 'ドメイン', 'アカウント', 'ユーザ名', and 'Vlan'. Below the form is a table titled '未実施プライベートコネクト一覧' (List of Unimplemented Private Connects) with columns: 番号, ドメイン, アカウント, ユーザ名, Vlan, VRF, 帯域, ソーン.

番号	ドメイン	アカウント	ユーザ名	Vlan	VRF	帯域	ソーン
NPV00422	70000000422	inoue0422	issei0422	400	22	1000	Tesla

設定削除

The screenshot shows the 'プライベートコネクト削除' (Delete Private Connect) page. It features a search bar, a dropdown for '全てのアカウント', and buttons for '20件', '100件', and 'CSV'. Below is a table with columns: ドメイン, アカウント, ユーザ名, Vlan, VRF, 帯域, ソーン, セグメント, 取得日, 更新日, 未実施.

ドメイン	アカウント	ユーザ名	Vlan	VRF	帯域	ソーン	セグメント	取得日	更新日	未実施
[X] NPV00001	70000000001	inoue0402	issei0402	10	100	1000	Tesla	192.168.1.0/24	2015-04-08 11:40:07 UTC	2015-04-08 11:40:07 UTC
[X] NPV00002	70000000002	inoue0402	issei0402	20	100	1000	Henry	192.168.2.0/24	2015-04-08 11:40:15 UTC	2015-04-08 11:40:15 UTC

JSON形式での情報取得

会場限り

POSTでのレコード追加

REQUEST

HTTP 150.205.102.121:4567/privateconnect POST Send

HEADERS form BODY text

```
1 {
2   "version": "07/000008",
3   "serial": "000000000",
4   "account": "issue0408",
5   "username": "issue0408",
6   "width": "100",
7   "url": "108",
8   "brandid": "1000",
9   "zone": "letra",
10  "system": "192.168.8.0/24",
11  "job_name": "IT"
12 }
13
```

length: 234 Bytes

ルータへの設定投入

REQUEST

HTTP 150.205.102.121:4567/set_privateconnect POST Send

1. 情報抽出

全文検索で対象情報の
容易な抽出が可能

2. 設定追加

Search.. 全てのアカウント ▾ 20件 100件 CSV

3. 設定削除

回線番号	ドメイン	アカウント	ユーザ名	Vlan	VRF	帯域	ゾーン	セグメント	取得日	更新日
NPVN00111	70000000111	inoue01	issei01	111	110	1000	Tesla	192.168.111.254/24	2015-06-15 10:42:35 UTC	2015-06-17 01:58:20 UTC
NPVN00608	70000000608	Inoue	Issei	50	60	1000	Pascal	192.168.1.0/24	2015-06-12 05:20:00 UTC	2015-06-15 06:53:06 UTC
NPVN00612	70000000612	inoue0612	issei0612	612	612	1000	Tesla	192.168.61.254/24	2015-06-12 05:15:41 UTC	2015-06-12 05:16:47 UTC
NPVN00615	70000000615	inoue0615	issei0615	15	6	1000	Pascal	192.168.15.254/24	2015-06-15 06:51:00 UTC	2015-06-15 06:51:18 UTC
									2015-06-	2015-06-

1. 情報抽出

Add PrivateConnect

新規レコード追加	番号	ドメイン	アカウント	ユーザ名	Vlan	VRF	ゾーン	VIP/セグメント
	例)NPVN00001	例)70000000001	例)idcfaccount	例)idcfara	例)101	例)201	例)Testa	例)192.168.1.254/24

2. 設定追加

未実施プライベートコネクト一覧

設定追加	番号	ドメイン	アカウント	ユーザ名	Vlan	VRF	ゾーン	VIP/セグメント	入力日時	Job実行(完了)
	NPVN00619	70000000619	inoue0619	inoue0619	619	619	Henry	192.168.61.254/24	2015-05-19 10:30:44 UTC	0 [X]

3. 設定削除

プライベートコネクト設定追加

下記の設定が入っています。

空白の場合には新規でのプライベートコネクト設定となります。

```
<name>
xe-2/0/0.19
</name><description>
NPVN00619:70000000619:inoue0619
</description><input-packets>
0
</input-packets><output-packets>
0
</output-packets><ifa-local>
192.168.19.253
</ifa-local>
```

キャンセル

設定投入

既存回線情報を見せて、
誤入力を抑制する

1. 情報抽出 プライベートコネクト削除

Search... 全てのアカウント ▾ 20件 100件 ☐ CSV

2. 設定追加

回線番号	ドメイン	アカウント	ユーザ名	Vlan	VRF	帯域	ゾーン	セグメント	取得日	更新
------	------	-------	------	------	-----	----	-----	-------	-----	----

3. 設定削除

[x] NPVN00111	70000000111	inoue01	issei01	111	110	プライベートコネクト設定削除				2015-06-	2015-
						削除対象のインタフェースの利用状況です					
[x] NPVN00608	70000000608	Inoue	Issei	50	60	<pre><name> xe-2/0/1.50 </name><description> NPVN00608-70000000608-Inoue </description><if- local> 192.168.1.253 </if- local><input-bytes> 0 </input-bytes><input-bytes> 0 </input-bytes><input-bytes> 0 </input-bytes><output-bytes> 0 </output-bytes><output-bytes> 0 </output-bytes><output-bytes> 0 </output-bytes><input-pps> 0 </input-pps><output-pps> 0 </output-pps><input-pps> 0 </input-pps><output-pps> 0 </output-pps></pre>					
[x] NPVN00612	70000000612	inoue0612	issei0612	612	612						

削除対象IFの利用状況を表示することで、誤入力を抑制する

キャンセル

設定削除



プログラミング未経験者でも
何とか作れたぞ(´▽`=)

- 入力IFの統一化、標準化
 - Bash expectを駆使、ある意味汎用的??
 - Netconf RPCベース、XMLでの表記
 - SOAP API 廃れつつある 複雑な入出力には向く?
 - Rest API 最近のデファクト URLがリソースに対応
- libraryの充実化、sample codeの充実化
 - 製品側でNetconfサポート、とかあっても意味ないです。
 - 使われるための仕組があってもなんぼです。
- ネットワークエンジニアでのSoftwareのエコシステム
 - メーカー側がPlugin、libraryとか作ってもユーザ側の方でも回していかないと普及しない。

Juniper

<https://github.com/Juniper/net-netconf>
<https://github.com/Juniper/netconf-perl>
<https://github.com/Juniper/netconf-java>
<https://github.com/Juniper/netconf-php>
<https://github.com/leopoul/ncclient>

Cisco

<https://github.com/jtimberman/ruby-cisco>
<https://github.com/nickpegg/ciscolib>

Brocade

<https://github.com/brocade/ncclient>
<https://github.com/brocade/brocade> (OpenStack Plugin)
<https://github.com/BRCDComm/BVC> (VyattaController)
https://github.com/zapman449/brocade_switchshow_aliases (Fiber switches)

Alaxala

<https://github.com/sumikawa/netconf>

Cumulus

<https://github.com/CumulusNetworks/cumulus-linux-ansible-modules>
<https://github.com/CumulusNetworks/cumulus-linux-chef-modules>
<https://github.com/CumulusNetworks/net-next>
<https://github.com/CumulusNetworks/quagga>
<https://github.com/CumulusNetworks/cumulus-cl-interfaces-puppet>
<https://github.com/OpenRTMFP/Cumulus> (MonaServer使ったSW)
<https://github.com/cotdsa/cumulus>
<http://cumulusnetworks.com/blog/cumulus-linux-2/>

プロジェクト名(アカウント名)は
早めに登録した方が良いです...φ(・ω・`)

標準API (全スイッチを共通で設定できるAPI) があるのが理想ですが、色々と難しいと思いますので、せめてメーカーからの正式なlibraryが欲しい



OpenStack Plugin (SDN) は各社競って開発していますが、
Underlay機器側も頑張ってください！！

OpenStack Plugin




開発エンジニアの人がその気になれば
多分すぐに作れると思います。

Plugins

The set of plugins included in the main Neutron distribution and supported by the Neutron community include:

- [Open vSwitch](#) Plugin
- [Cisco UCS/Nexus](#) Plugin
- [Cisco Nexus1000v](#) Plugin
- [Linux Bridge](#) Plugin
- [Modular Layer 2](#) Plugin
- [Nicira Network Virtualization Platform \(NVP\)](#) Plugin
- [Ryu OpenFlow Controller](#) Plugin
- [NEC OpenFlow](#) Plugin
- [Big Switch Controller](#) Plugin
- [Cloudbase Hyper-V](#) Plugin
- [MidoNet](#) Plugin
- [Brocade Neutron Plugin](#)  Brocade Neutron Plugin
- [PLUMgrid](#)  Plugin
- [Mellanox Neutron Plugin](#)  Mellanox Neutron Plugin
- [Embrane Neutron Plugin](#) 
- [IBM SDN-VE](#)  Plugin
- [CPLANE NETWORKS](#)  CPLANE NETWORKS
- [Nuage Networks](#) Plugin
- [OpenContrail](#)  OpenContrail Plugin

Additional plugins are available from other sources:

- [Extreme Networks](#) Plugin
- [Ruijie Networks](#)  Plugin
- [Juniper Networks](#)  Neutron Plugin
- [Calico](#)  Neutron Plugin  (docs)

If you have your own plugin, feel free to add it to this list.

性能・機能は横並び
「使われ易さ」が重要

TOPICSと苦勞話

Netconfもツライところがあります。

素のXML形式の記述はちょっと面倒。Junosは”show configuration | display xml”が便利！

Rubyだとハイフンは正規表現になるのでsendメソッド使う必要がある

RPC難しい。プログラムがlockから抜けないうちがあったり。。

やっぱりRestAPIが一番良いです（特に素人向けには）

XML形式

```
Netconf::SSH.new(target: "[router]", username: "inoue", password: ENV['PASSWORD']) do |c|
  puts device.rpc.lock(:and|date)
  puts device.rpc.edit_config { |x|
    x.configuration {
      x.interfaces {
        x.interface {
          x.name interface
          x.unit {
            x.name vlan
            x.description "!!vpn_num:!!{domain}:!!{account}"
            x.send("vlan-id", vlan)
            x.family {
              x.inet {
                x.filter {
                  x.input {
                    x.send("filter-name", "1G")
                  }
                }
              }
            }
            x.address {
              x.name router_a
              x.send("vrrp-group") {
                x.name 163
                x.send("virtual-address", gateway)
                x.priority "150"
              }
            }
          }
        }
      }
    }
  }
  puts device.rpc.validate(:and|date)
  puts device.rpc.commit
  puts device.rpc.unlock :and|date
end
```

Junos Config

```
interfaces {
  "interface" {
    unit "vlan" {
      description NPVN00615:70000000615:inoue0615;
      vlan-id 15;
      family inet {
        filter {
          input 1G;
        }
        address 192.168.15.253/24 {
          vrrp-group 163 {
            virtual-address 192.168.15.254;
            priority 150;
          }
        }
      }
    }
  }
}
```

> Ruby for JunosでのNetconfの情報確認

show関連の情報もxml形式なので、そのままだと見きれない。。

```
<logical-interface>
<name>
xe-2/0/0.111
</name>
<local-index>
881
</local-index>
<snmp-index>
743
</snmp-index>
<generation>
874
</generation>
<description>
NPVND0111:70000000111:inoue01
</description>
<if-config-flags>
<if-device-down/>
<if-snmp-traps/>
<internal-flags>
0x4000
</internal-flags>
</if-config-flags>
<link-address junos:format="VLAN-Tag [ 0x8100.111 ] ">
[ 0x8100.111 ]
</link-address>
<encapsulation>
ENET2
</encapsulation>
<traffic-statistics junos:style="verbose" junos:indent="2">
<input-bytes>
0
</input-bytes>
<output-bytes>
0
</output-bytes>
<input-packets>
0
</input-packets>
<output-packets>
0
</output-packets>
</traffic-statistics>
<local-traffic-statistics>
<input-bytes>
0
</input-bytes>
<output-bytes>
0
</output-bytes>
<input-packets>
0
</input-packets>
```

```
</input-packets>
<output-packets>
0
</output-packets>
</local-traffic-statistics>
<transit-traffic-statistics>
<input-bytes>
0
</input-bytes>
<input-bps>
0
</input-bps>
<output-bytes>
0
</output-bytes>
<output-bps>
0
</output-bps>
<input-packets>
0
</input-packets>
<input-pps>
0
</input-pps>
<output-packets>
0
</output-packets>
<output-pps>
0
</output-pps>
</transit-traffic-statistics>
<filter-information>
</filter-information>
<address-family>
<address-family-name>
inet
</address-family-name>
<mtu>
1500
</mtu>
<generation>
1232
</generation>
<route-table>
20
</route-table>
<address-family-flags>
<iff-is-primary/>
<iff-sendbcst-pkt-to-re/>
<internal-flags>
0x0
</internal-flags>
</address-family-flags>
<filter-information>
<filter-input>
```

```
</input-packets>
<output-packets>
0
</output-packets>
</local-traffic-statistics>
<transit-traffic-statistics>
<input-bytes>
0
</input-bytes>
<input-bps>
0
</input-bps>
<output-bytes>
0
</output-bytes>
<output-bps>
0
</output-bps>
<input-packets>
0
</input-packets>
<input-pps>
0
</input-pps>
<output-packets>
0
</output-packets>
<output-pps>
0
</output-pps>
</transit-traffic-statistics>
<filter-information>
</filter-information>
<address-family>
<address-family-name>
inet
</address-family-name>
<mtu>
1500
</mtu>
<generation>
1232
</generation>
<route-table>
20
</route-table>
<address-family-flags>
<iff-is-primary/>
<iff-sendbcst-pkt-to-re/>
<internal-flags>
0x0
</internal-flags>
</address-family-flags>
<filter-information>
<filter-input>
```

show interfaces xe-2/0/0.111 detailの結果 3スクロール。。

Node追加

```
$ curl -sk -H "Authorization: Basic xxxxxxxxxxxxxxxx" https://x.x.x.x/mgmt/tm/ltn/node -H 'Content-Type: application/json' -X POST -d '{"address": "192.168.0.1","description": "testdescription","name": "testname"}'
```

```
{"kind": "tm:ltn:node:nodestate", "name": "testname", "fullPath": "testname", "generation": 36, "selfLink": "https://localhost/mgmt/tm/ltn/node/testname?ver=11.5.1", "address": "192.168.0.1", "connectionLimit": 0, "description": "testdescription", "dynamicRatio": 1, "logging": "disabled", "monitor": "default", "rateLimit": "disabled", "ratio": 1, "session": "monitor-enabled", "state": "checking"}
```

Poolへのmember追加

```
$ curl -sk -H "Authorization: Basic xxxxxxxxxxxxxxxx" https://x.x.x.x/mgmt/tm/ltn/pool/~Common~pool_loganalyzer/members -H 'Content-Type: application/json' -X POST -d '{"name": "testname:514"}'
```

```
{"kind": "tm:ltn:pool:members:membersstate", "name": "testname:514", "fullPath": "testname:514", "generation": 38, "selfLink": "https://localhost/mgmt/tm/ltn/pool/~Common~pool_loganalyzer/members/testname:514?ver=11.5.1"}
```

設定のSync

```
$ curl -sk -H "Authorization: Basic xxxxxxxxxxxxxxxx" https://x.x.x.x/mgmt/tm/ltn/pool/~Common~pool_loganalyzer/members -H 'Content-Type: application/json' -X POST -d '{"name": "testname:514"}'
```

```
{"kind": "tm:ltn:pool:members:membersstate", "name": "testname:514", "fullPath": "testname:514", "generation": 38, "selfLink": "https://localhost/mgmt/tm/ltn/pool/~Common~pool_loganalyzer/members/testname:514?ver=11.5.1"}
```

Ruby NetAddr Package

ブロードキャストアドレスやゲートウェイアドレス（末尾アドレス）をゼロから記述することは意外と難しい。

例えば、192.168.1.0/24という入力値から、192.168.1.254というゲートウェイアドレスをIPAddrクラスだけで記述しようと意外と面倒。

NetAddrというlibraryがあって、これが便利だった。

<https://rubygems.org/gems/netaddr/versions/1.5.0>

<http://www.rubydoc.info/gems/netaddr/1.5.0/NetAddr>

右の他にもARPA形式出力やrange指定、IPv6対応など、色々便利なメソッドがある。

```
netaddr1 = NetAddr::CIDR.create('192.168.1.0/24')

# broadcastアドレスを算出
bc_address = netaddr1.last

# 数値化して1引く
gateway_int = NetAddr::CIDR.create(bc_address).to_i - 1
# その数値をアドレス表記に戻す
gateway = NetAddr.i_to_ip(gateway_int)

# /24部分だけを抽出
netmask = netaddr1.netmask

p bc_address
# => "192.168.1.255"

p gateway
# => "192.168.1.254"

p gateway + netmask
# => 192.168.1.254/24
```

- ドットインストール

<http://dotinstall.com/>

- 書籍

ruby、API関連の本

- 近くのソフトウェアエンジニア

- 気合！！

- IPアドレス表記じゃないものが入力されたら？
- ロジック的におかしいものが入力されたら？
- ユーザにどういう形でエラーを返す？
- 処理が途中で終わってしまった場合にはどうやってrollbackする？
- 操作履歴のログはどこでどうやってとる？
- 対象の機器がメンテ中だったり障害の場合はどうする？
- システムの冗長化/DRはどうする？データ保全是？
- 機器入れ替えるときにコードはどうやってメンテする？
- 世代管理、テスト、CI(Continuous Integration)
- 誰が引き継ぐ？

ソフトウェア開発は終わりが無い・
想定外のエラーをどこまで想定するか

色々大変ですが、

「ネットワークが連携」する世界ってどうですか？
作業が自動化(セルフ化)できたらどうですか？
ネットワークがアプリケーションのように扱えたら
どうですか？

きっとサービスの幅が広がっていく
きっとセキュリティ高まる
きっと安定したネットワークが提供できる

色々な英知が掛け合わさって
新しいモノが生まれてくる
それがインターネット

一人・一社でネットワークは創れない
つないでナンボです

Network × SoftwareでNext Internetが
生まれるかも！！

株式会社IDCフロンティア
技術開発本部 R&D室
井上一清



情報発信苦手ですが、これから頑張っていきたいと思います。

<http://qiita.com/inoueissei>

<https://github.com/inoueissei>

<https://www.facebook.com/inoue.issei>

<https://twitter.com/inoueissei>

ご清聴ありがとうございました！！



未来をささえる、Your Innovative Partner