

# 学認クラウドオンデマンド構築サービス (OCS)の概要

2022年12月23日

大江 和一

国立情報学研究所  
クラウド基盤研究開発センター

# はじめに

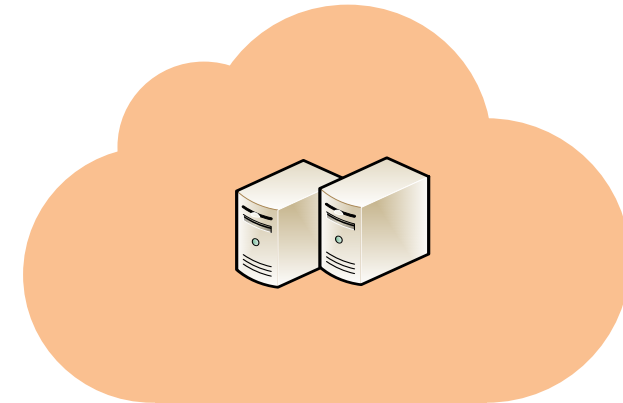
今回の学認クラウドオンデマンド構築サービスセミナー実施に当たり、実習及び実習後のお試し環境利用に必要な計算資源は、さくらのクラウド様よりご提供頂いております。

# OCSとは

# OCS提供の背景(1)



クラウドA



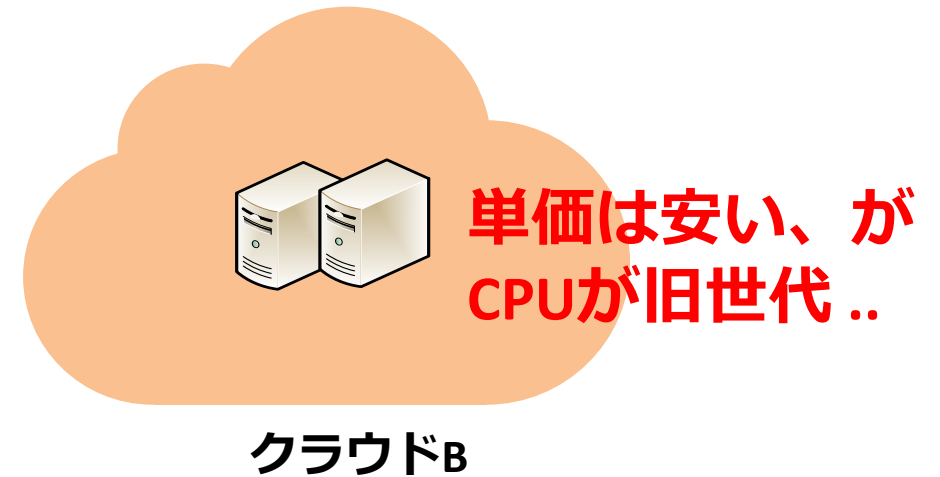
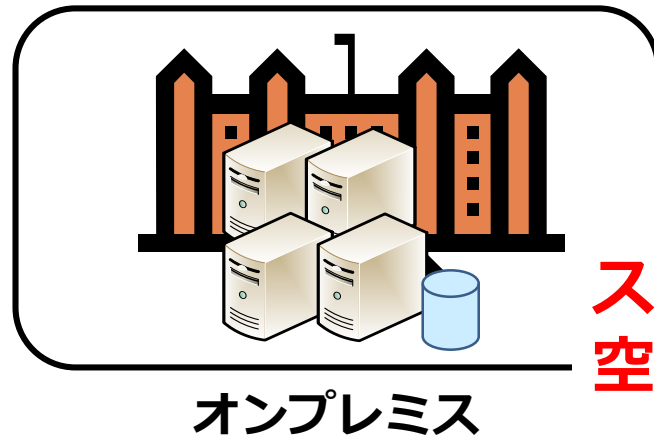
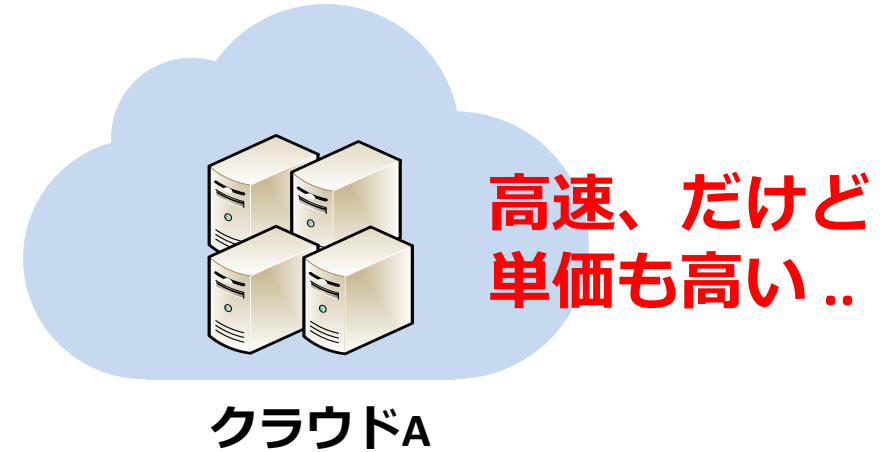
クラウドB



オンプレミス

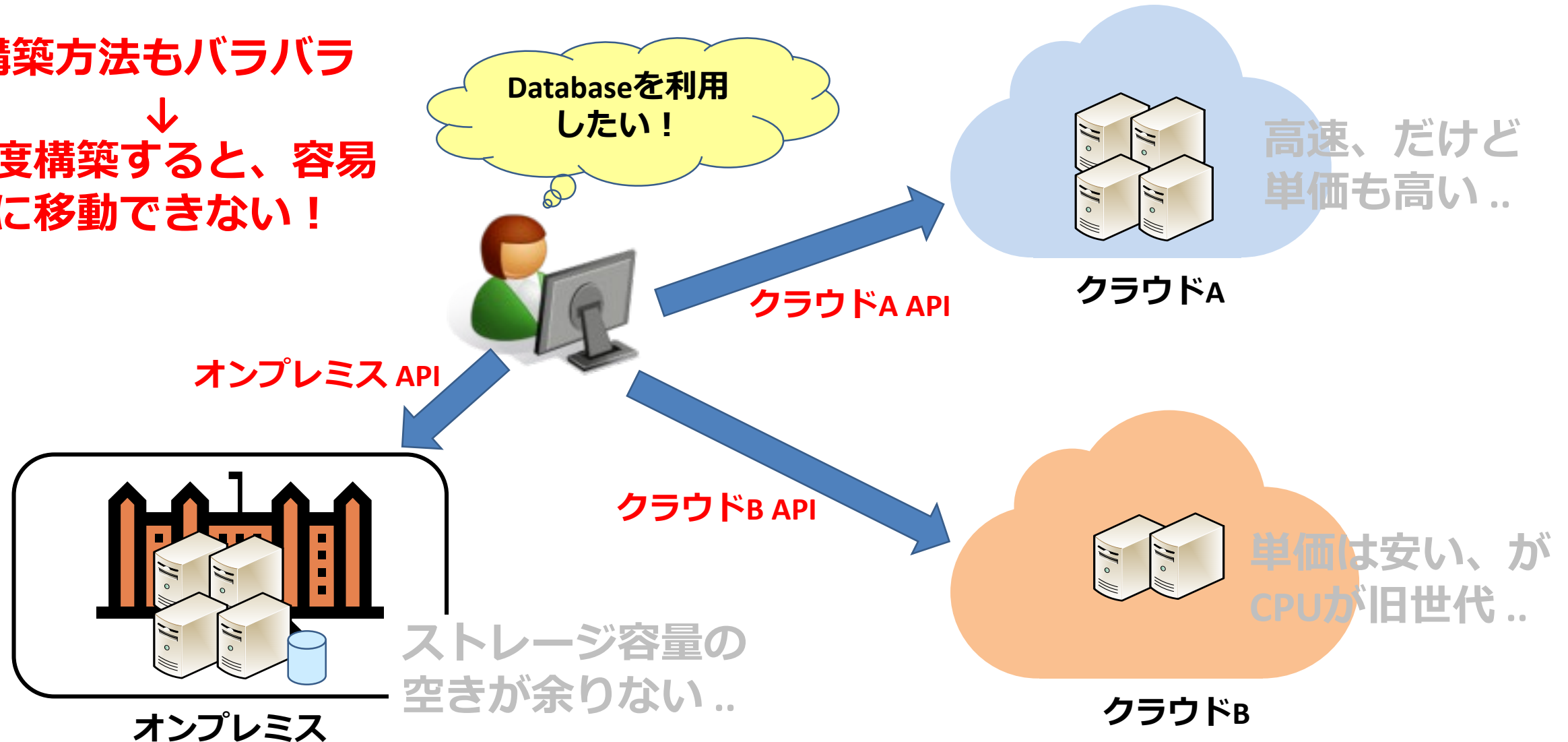
# OCS提供の背景(2)

どの環境を選ぶべきか？



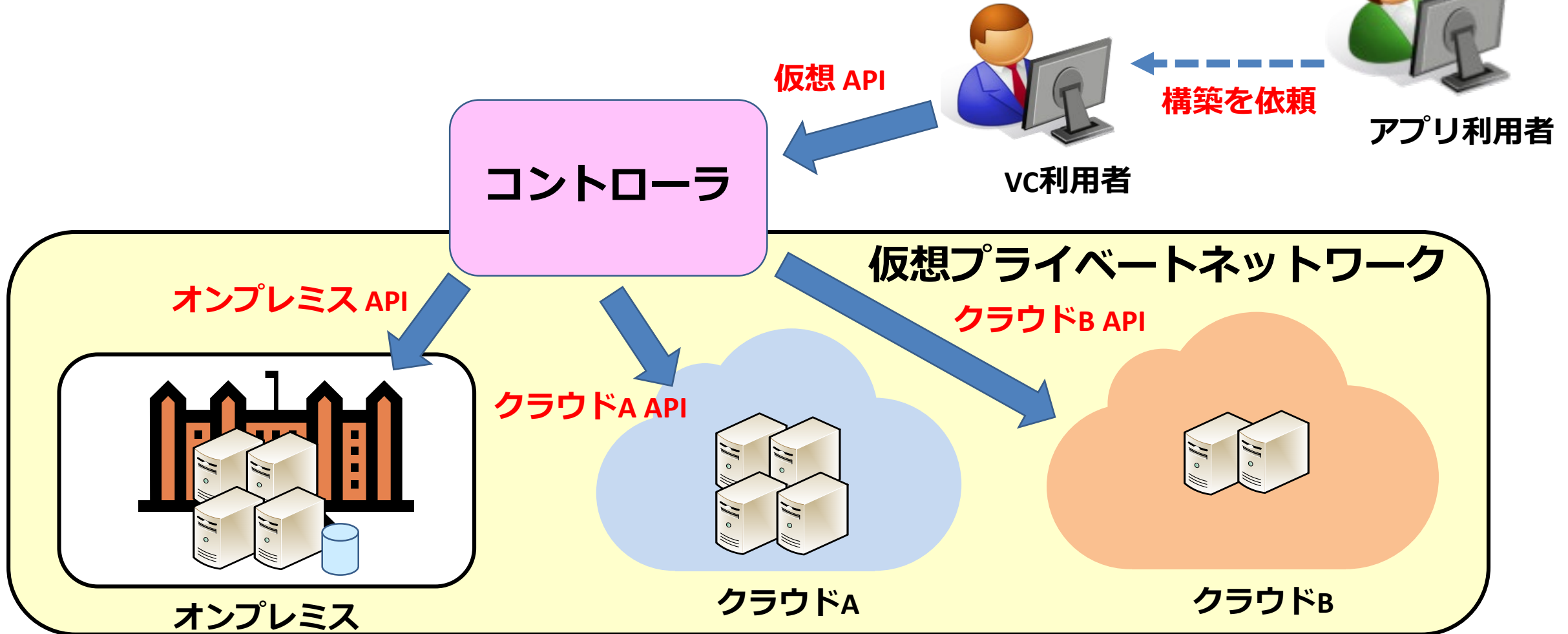
# OCS提供の背景(3)

構築方法もバラバラ  
↓  
一度構築すると、容易  
に移動できない！



# OCSの特徴(1)

仮想APIのみで全ての資源の操作が可能！



# OCSの特徴(1)

**オンプレミスに  
Database構築！**

**コントローラ**

仮想 API

Databaseを利用  
したい！

構築を依頼

アプリ利用者

vc利用者

オンプレミス API

仮想プライベートネットワーク

クラウドB API

クラウドA API

オンプレミス

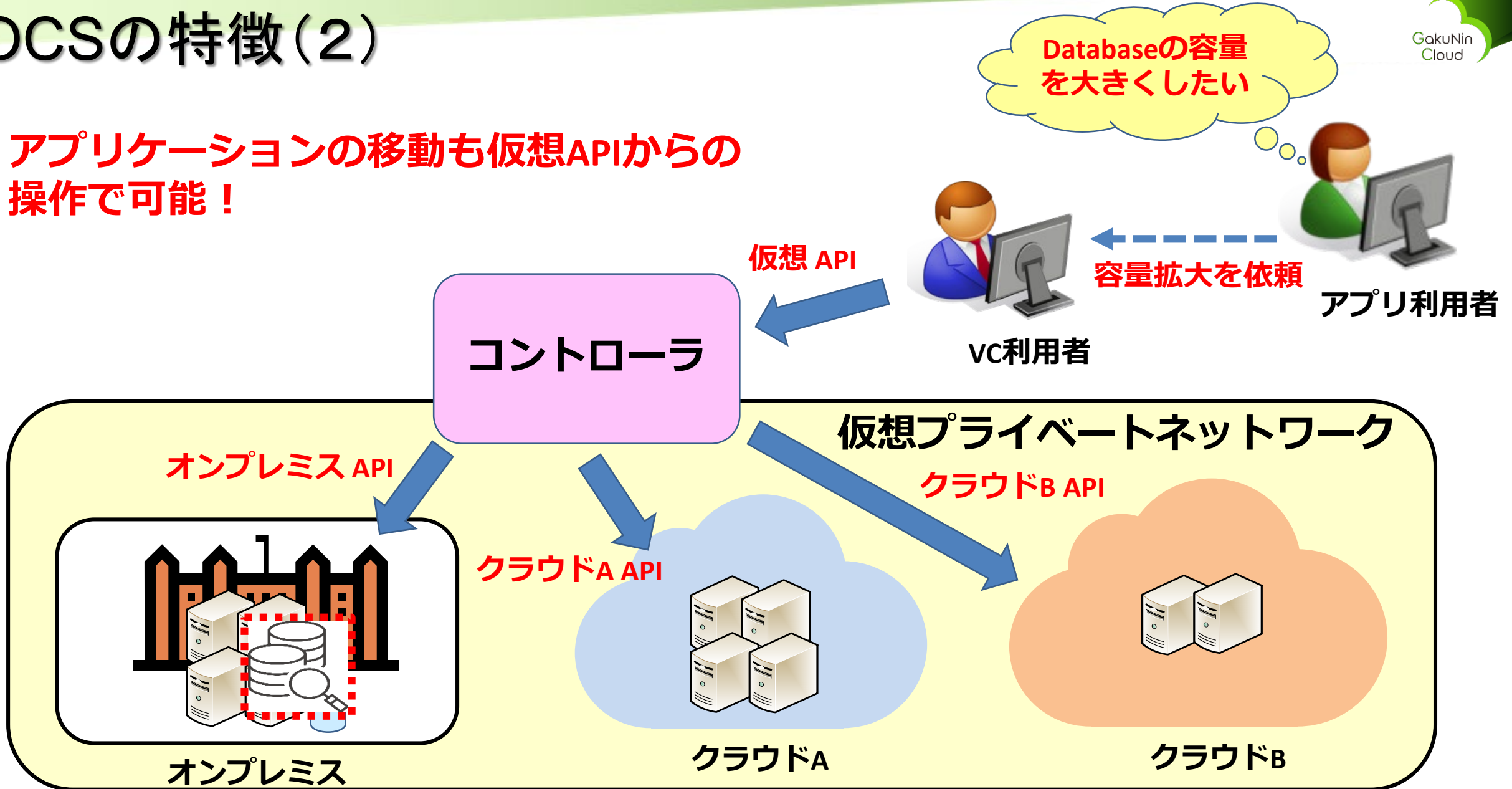
クラウドA

クラウドB



# OCSの特徴(2)

アプリケーションの移動も仮想APIからの  
操作で可能！



# OCSの特徴(2)

クラウドAに容量を拡大  
してDatabaseを移動！

コントローラ

仮想 API

Databaseの容量  
を大きくしたい

容量拡大を依頼

アプリ利用者

vc利用者

仮想プライベートネットワーク

クラウドB API

クラウドA API

ストレージが足らな  
くなったので移動

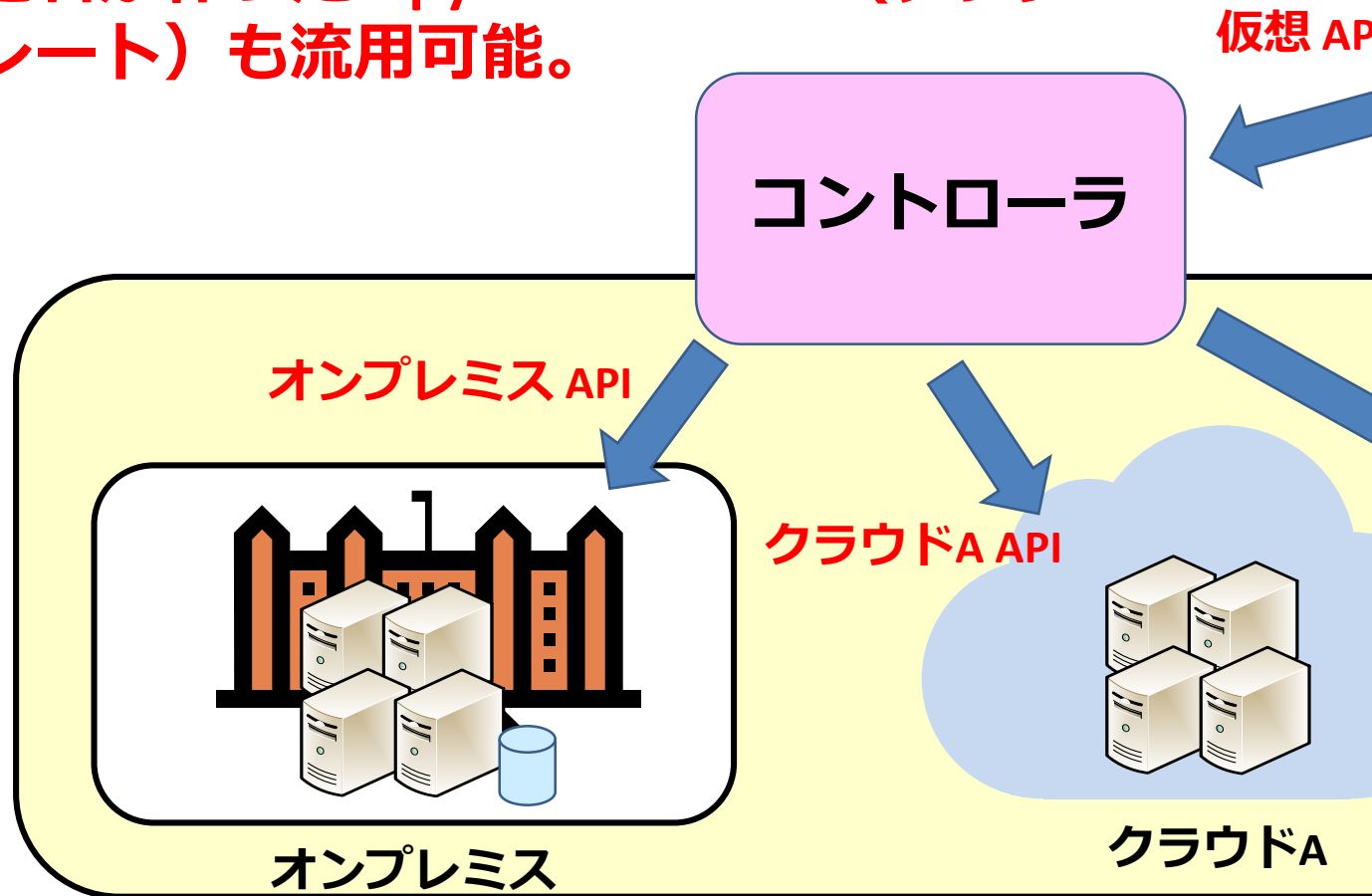
オンプレミス

クラウドA

クラウドB

# OCSの特徴(3)

仮想APIはJupyter Notebookを介してアクセスするため、構築作業の再現性が高い！  
他者が作ったJupyter Notebook（テンプレート）も流用可能。



## 1.1 初期化 Jupyter Notebookの記述例

```
[1]: parameters
1 vcc_access_token = "..."
2 testname = "TEST-2022-03-15"

[2]:
1 from common import logsetting
2 from vcpsdk.vcpsdk import VcpSDK
3
4 #
5 # VCP SDK の初期化
6 #
7
8 sdk = VcpSDK(vcc_access_token)
9
10 # VCP SDK バージョン確認
11 sdk.version()
12
13 # UnitGroup作成
14 my_ugroup_name = "03_sample" + testname
15
16 ugroup = sdk.get_ugroup(my_ugroup_name)
17 if ugroup is None:
18     ugroup = sdk.create_ugroup(my_ugroup_name)

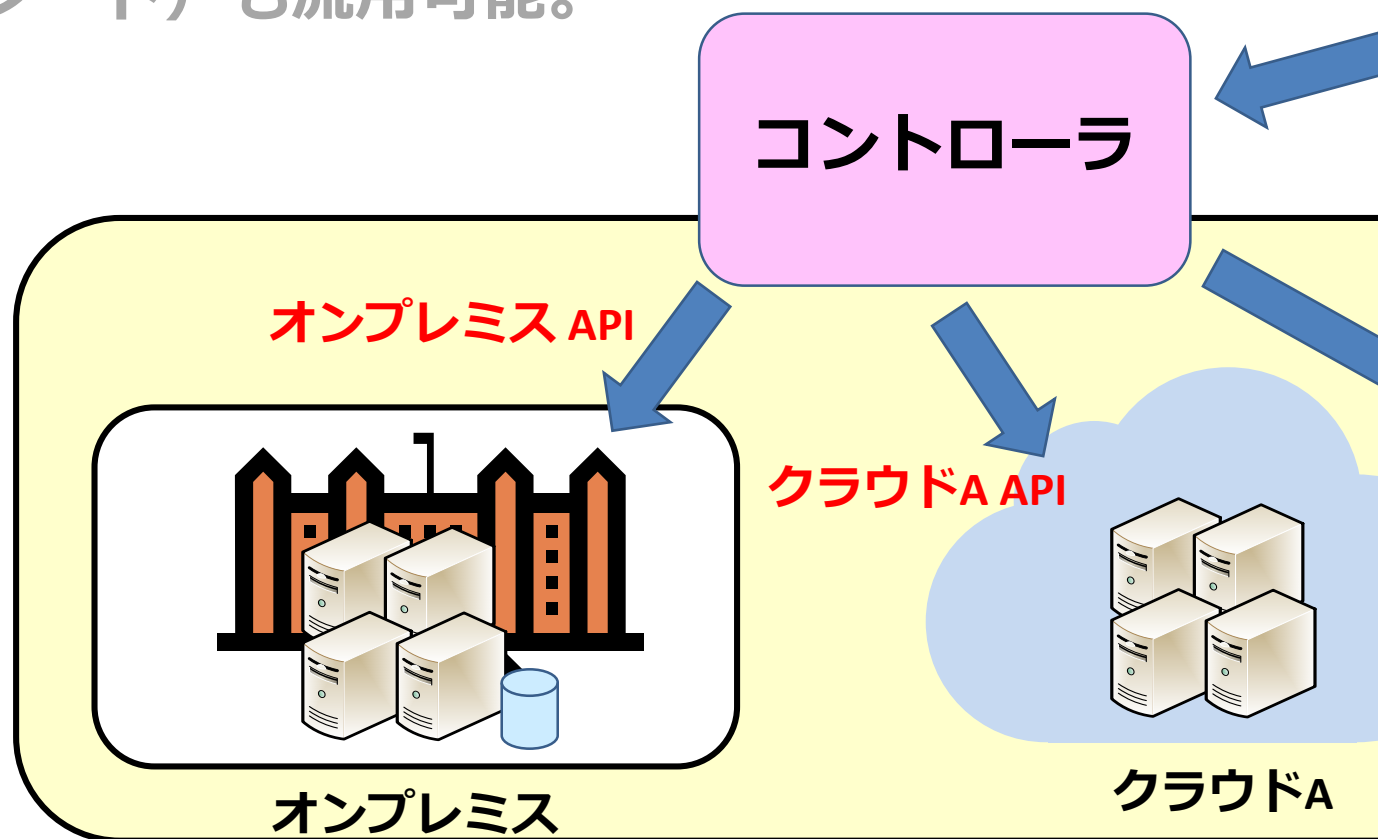
vcplib:
  filename: /home/jovyan/vcpsdk/vcplib/occtr.py
  version: 20.10.0+20201001

vcpsdk:
```

# OCSの特徴(3)

仮想APIはJupyter Notebookを介してアクセスするため、構築作業の再現性が高い！  
他者が作ったJupyter Notebook（テンプレート）も流用可能。

**VC利用者となる敷居は低いです！**



### 1.1 初期化 Jupyter Notebookの記述例

```

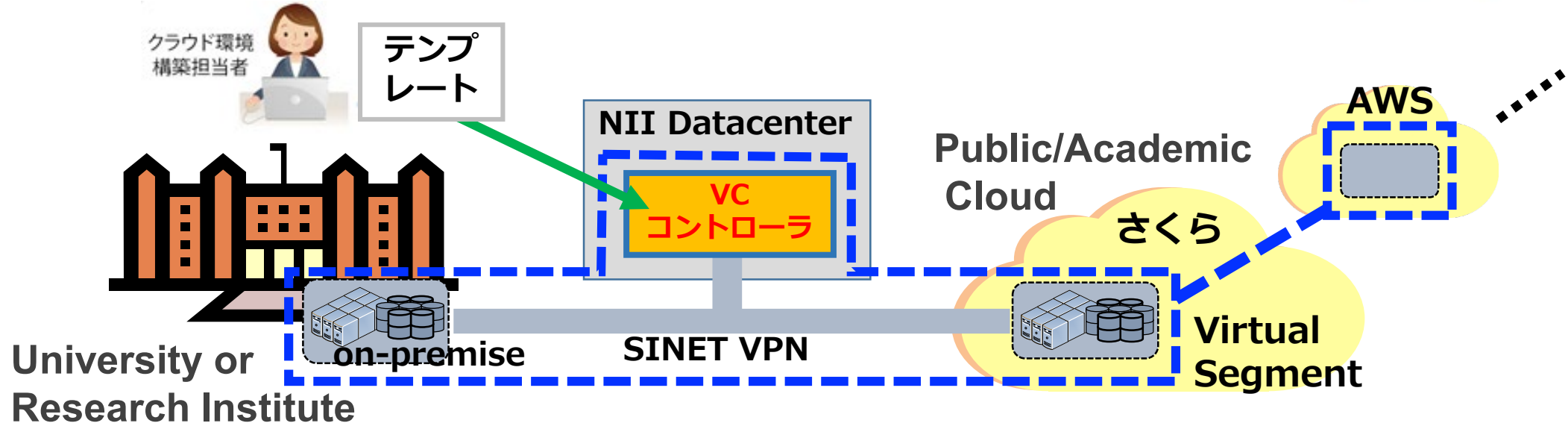
[1]: parameters
1 vcc_access_token = "..."
2 testname = "TEST-2022-03-15"

[2]:
1 from common import logsetting
2 from vcpsdk.vcpsdk import VcpSDK
3
4 #
5 # VCP SDK の初期化
6 #
7
8 sdk = VcpSDK(vcc_access_token)
9
10 # VCP SDK バージョン確認
11 sdk.version()
12
13 # UnitGroup作成
14 my_ugroup_name = "03_sample" + testname
15
16 ugroup = sdk.get_ugroup(my_ugroup_name)
17 if ugroup is None:
18     ugroup = sdk.create_ugroup(my_ugroup_name)

vcplib:
  filename: /home/jovyan/vcpsdk/vcplib/occtr.py
  version: 20.10.0+20201001

vcpsdk:
  
```

# OCSの特徴(まとめ)



- テンプレートを用いて、オンプレミスやクラウド(IaaS)上にアプリケーション実行環境を構築するサービス
  - 仮想プライベートネットワーク(VPN)内に利用する資源を囲い込み、仮想コントローラ(VCコントローラ)から操作することで、全ての資源を統一的に利用できる。
  - VCコントローラの操作は、可読性が高いテンプレート(JupyterNotebook)からの操作が可能。

# OCSの特徴(テンプレート)

他者が作ったテンプレートの流用も可能



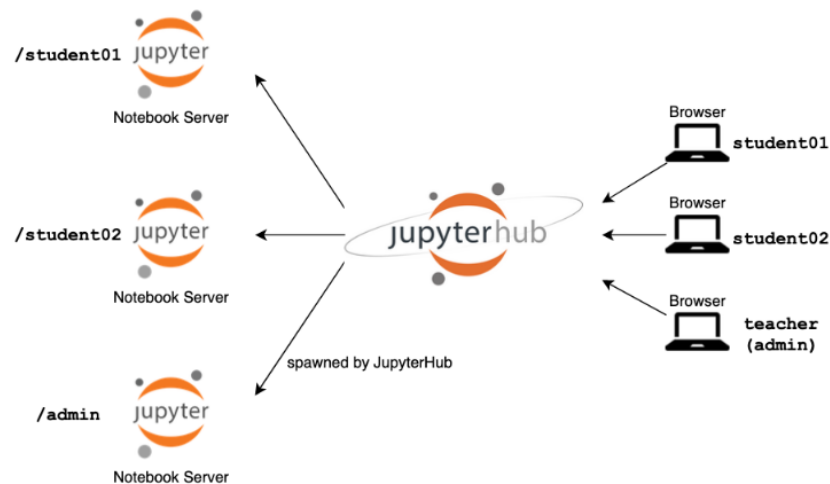
## The Littlest JupyterHub による軽量Python実習環境の構築

JupyterHub は、Webブラウザからアクセス可能なマルチユーザ対応の認証機能付きJupyter Notebookサーバです。

JupyterHubを利用して管理者が用意したNotebookをユーザがブラウザからすぐに実行可能な環境を提供できるため、Pythonによるプログラミング研修やワークショップを開催したり、講義演習環境として活用したりするのに適しています。

本ハンズオンでは、JupyterHubを小規模なグループで手軽に利用することを想定し、単一のサーバで実行するために開発された「The Littlest JupyterHub」(以下「TLJH」と略)をVCPを用いて構築します。

ハンズオンご参加の皆様には、このテンプレートでTLJHによるVCPアプリケーション環境を構築していただきます。



### 構築環境情報の入力

TLJH環境の構築情報を入力します。必要に応じ、下記の情報を修正してください。

★ハンズオンでは以下のパラメータを変更しないでください★

```
#####
### ハンズオンでは以下のパラメータを変更しないでください。 ###
#####

# UnitGroup
ugroup_name = "hands003"

# プロバイダ
vc_provider = "aws"
```

スクリプトを組み込むことができ、ここから実行できる。実行結果を残すことも出来る。

### VCノードのspecを指定

TLJH を利用するのに十分な性能

固定割当IPアドレスは、ハンズオンでは指定していません。

```
In [ ]: # UnitGroup の作成
unit_group = vcp.create_ugroup(ugroup_name)

# VCノード spec
spec = vcp.get_spec(vo_provider, vcnode_flavor)

# spec オプション (ディスクサイズ 単位:GB)
spec.volume_size = volume_size

# spec オプション (固定割当IPアドレス)
spec.ip_addresses = [fixed_ipaddress]

# ssh keyfiles
import os
ssh_public_key = os.path.expanduser("~/ssh/id_rsa.pub")
spec.set_ssh_pubkey(ssh_public_key)
```

### Unitの作成とVCノードの起動

Unitを作成します。Unitを作成すると同時に VCノード (ここでは Amazon EC2インスタンス) が起動します。処理が完了するまで1分半~2分程度がかかります。

```
In [ ]: # Unitの作成 (同時に VCノードが作成される)
unit = unit_group.create_unit('tljh-node', spec)
```

### 疎通確認

まず、ssh の `known_hosts` の設定を行います。

その後、VCノードに対して `uname -a` を実行し、`ubuntu x86_64 Linux` が起動していることを確認します。起動していない場合は、`spec.image` に誤りがあります。本テンプレート下部にある「環境の削除」を実行し、`spec.image` を修正、全てのセルを `unfreeze` してから、最初から再実行してください。

```
In [ ]: # unit_group.find_ip_addresses() は UnitGroup内の全VCノードのIPアドレスのリストを返します
ip_address = unit_group.find_ip_addresses(node_state='RUNNING')[0] # 今は1つのVCノードのみ起動しているので [0] で最初の要素を取り出す
print(ip_address)

# ssh 設定
!touch ~/.ssh/known_hosts
!ssh-keygen -R [ip_address] # ~/.ssh/known_hosts から古いホストキーを削除する
!ssh-keyscan -H [ip_address] >> ~/.ssh/known_hosts # ホストキーの登録

# システムの確認
!ssh [ip_address] uname -a
```

### TLJH (The Littlest JupyterHub) 環境の構築

VCノード上に、本ハンズオン用に用意したThe Littlest JupyterHubのコンテナイメージを使用して環境を構築します。

### TLJHコンテナイメージの取得

VCノード上にコンテナイメージを取得するために `docker pull` を実行します。

図表を組み合わせた説明を挿入できる

# OCSを支えるVCPの仕組み

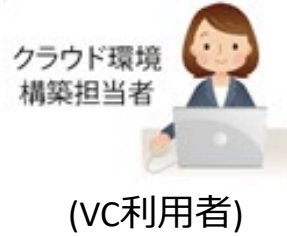
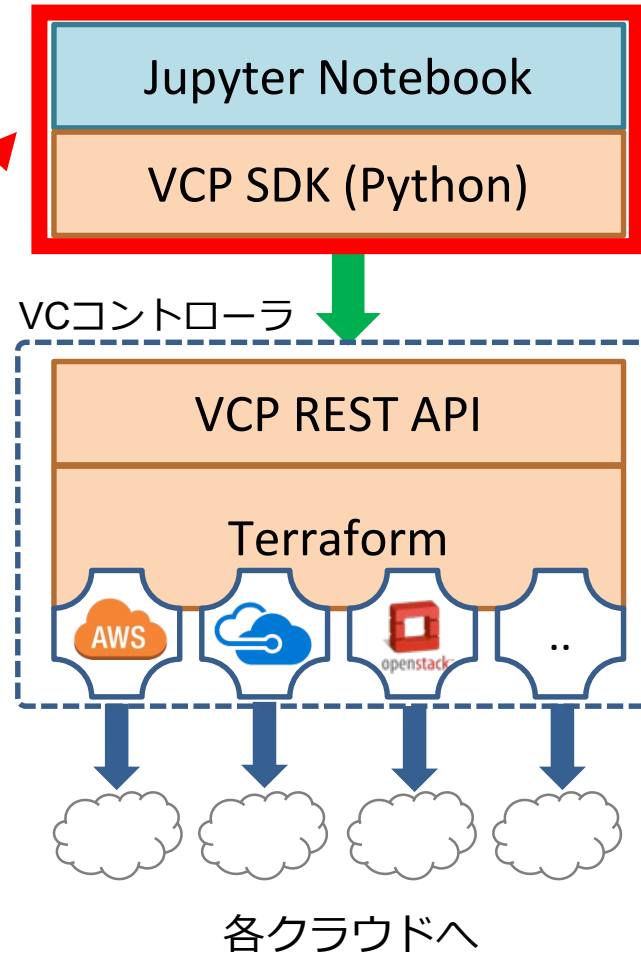
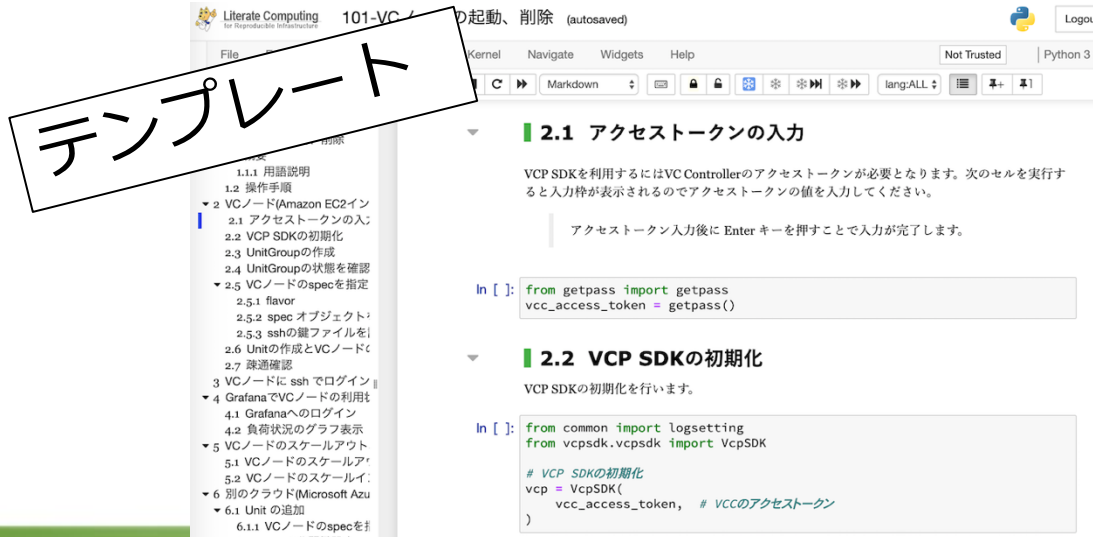


# 管理ソフトウェアの概要(1)

## ■ Virtual Cloud Provider (VCP)

- 本機能の中心ソフトウェア
- プロバイダI/Fを抽象化したREST API
- VCPの利用を容易にするPythonライブラリ VCP SDK

## ■ Jupyter Notebook(+NII拡張)からVCP SDKを利用して操作





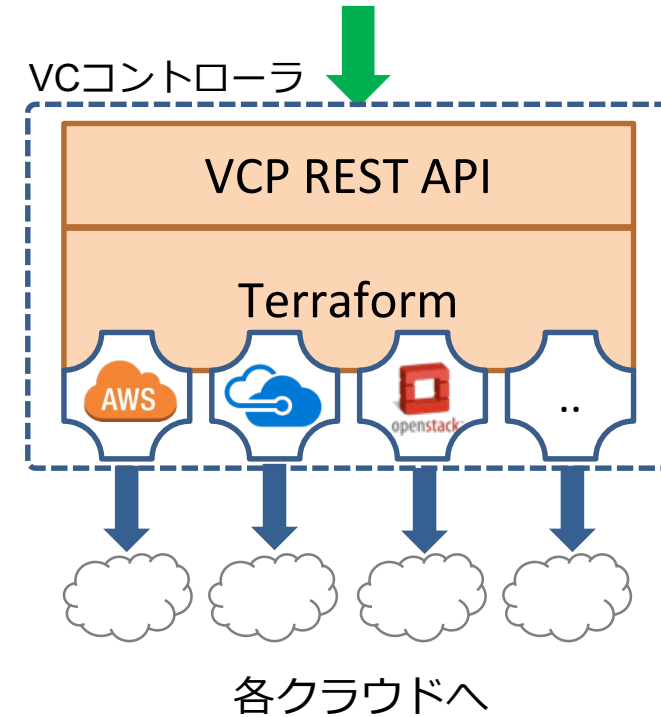
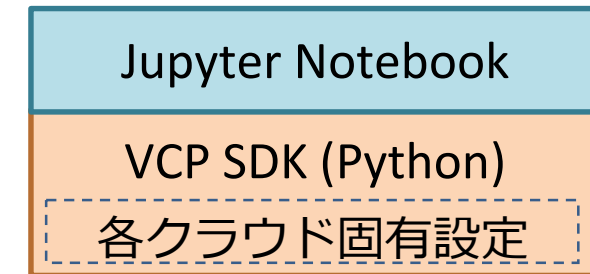
# 管理ソフトウェアの概要(2)

## ■ VCP SDK

- 各クラウドの固有設定をSDK内に隠蔽することで、Jupyter Notebookを変更することなくクラウド間での使い回しを実現



クラウド環境  
構築担当者



```
1.2 新規 server を作成

[5]:
1 #
2 # 作成するserverのspec情報を作成
3 #
4 spec = sdk.get_spec("aws", "medium")
5
6 #
7 # 変更できること
8 #
9 # spec.num_nodes = 1
10 spec.num_nodes = 2
11 #spec.instance_type = 't2.medium'
12 # spec.params_v = ['/opt:/opt']
13 # spec.volume_size = 40
14 # spec.volume_type = "standard" # standard/io1/gp2/sc1/st1
15 # spec.ip_addresses = ['起動するnodeの静的なIPアドレス']
16 # spec.image = 'vcp/base:1.5' # base container
17 # 追加で使用するVolume
18 # spec.disks = ['vol-08cbb04b35c8c9545']
19
20 # cloud上のタグ設定
21 spec.set_tag('key1', 'value1')
22 spec.set_tag('key2', 'value2')
23
```

**VCP SDKの中で各クラウドのmediumを定義**

# 計算インスタンス (VCノード)

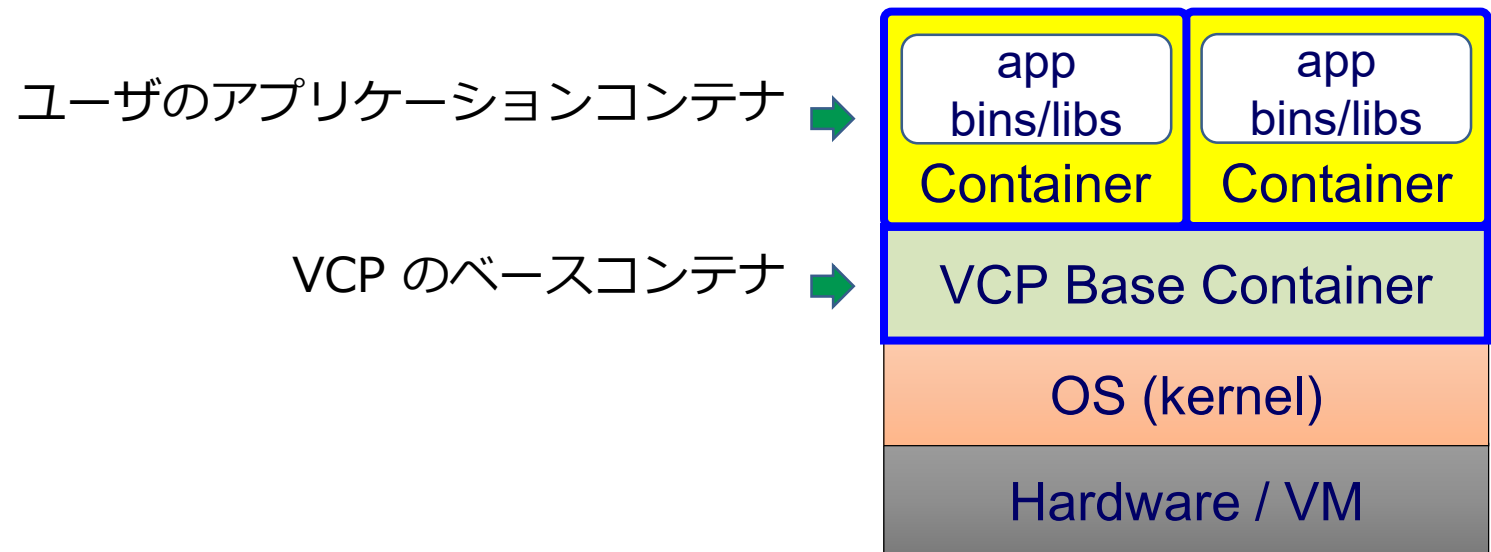
## ■ Docker in Docker 構成

### ■ ベースコンテナ

- 死活監視やメトリクス収集などシステムの基本機能

### ■ アプリケーションコンテナ

- アプリケーションと関連ソフトウェアをベースコンテナ上に起動
- Dockerのエコシステムが利用可能



# モニタリング機能

- ベースコンテナ、アプリコンテナのモニタリング情報を提供
- アプリケーションの収容設計を支援

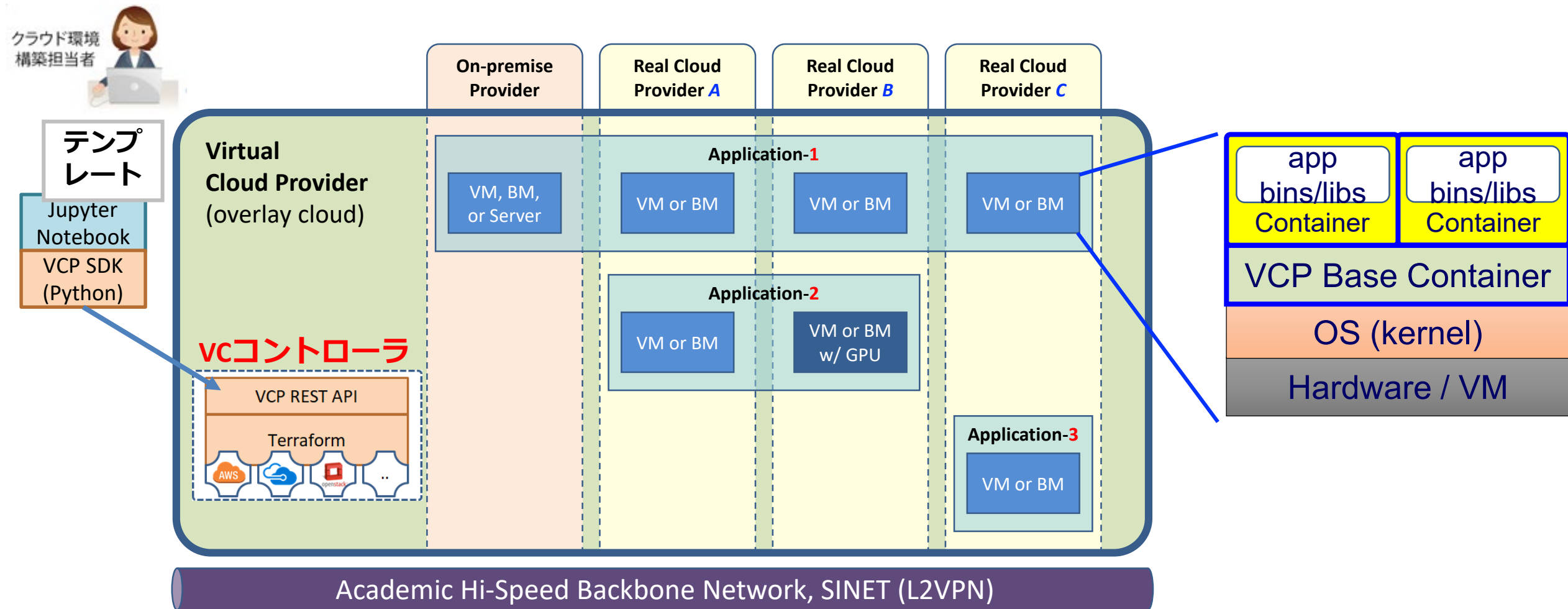


VCノード (ベースコンテナ) 毎の情報

アプリコンテナ毎の情報

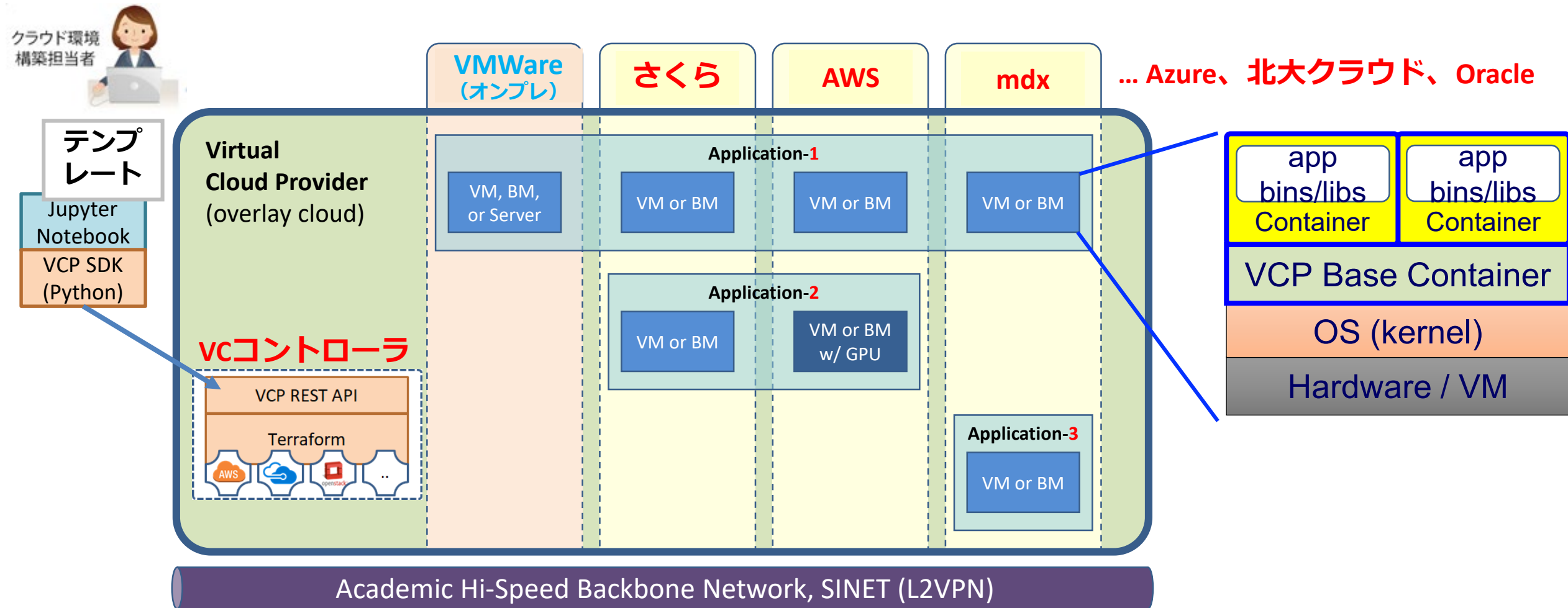
# OCSを利用したアプリケーション配備例

- オンプレ・複数の実クラウドを跨ってのアプリケーション配備が可能！



# OCSを利用したアプリケーション配備例

- オンプレ・複数の実クラウドを跨ってのアプリケーション配備が可能！



# サービス版とポータブル版

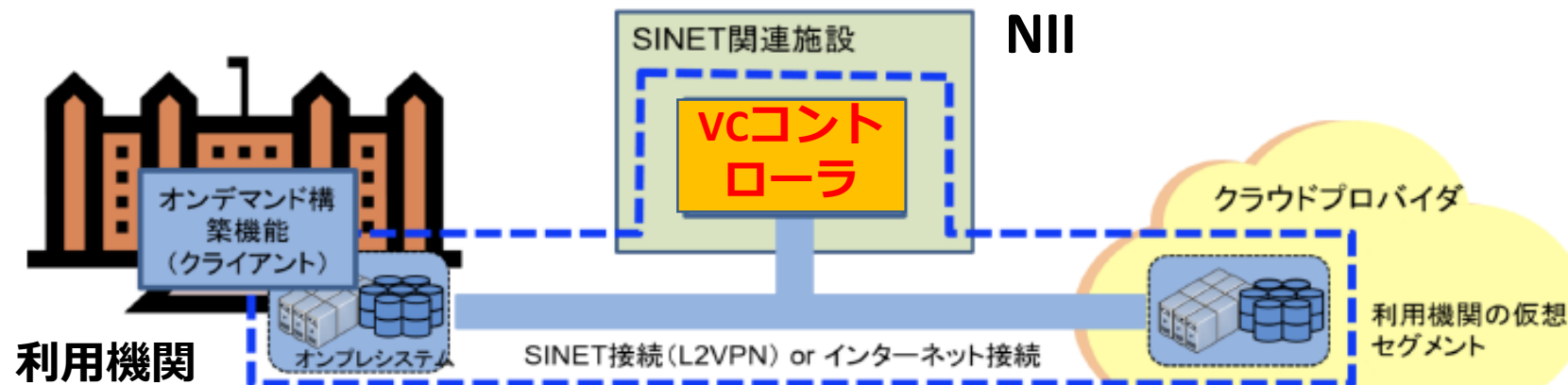
長所：

NII側でVCP運用・保守  
仮想ルータが利用可能

短所：

NIIへのVCP構築申請  
が必要

## サービス版



長所：

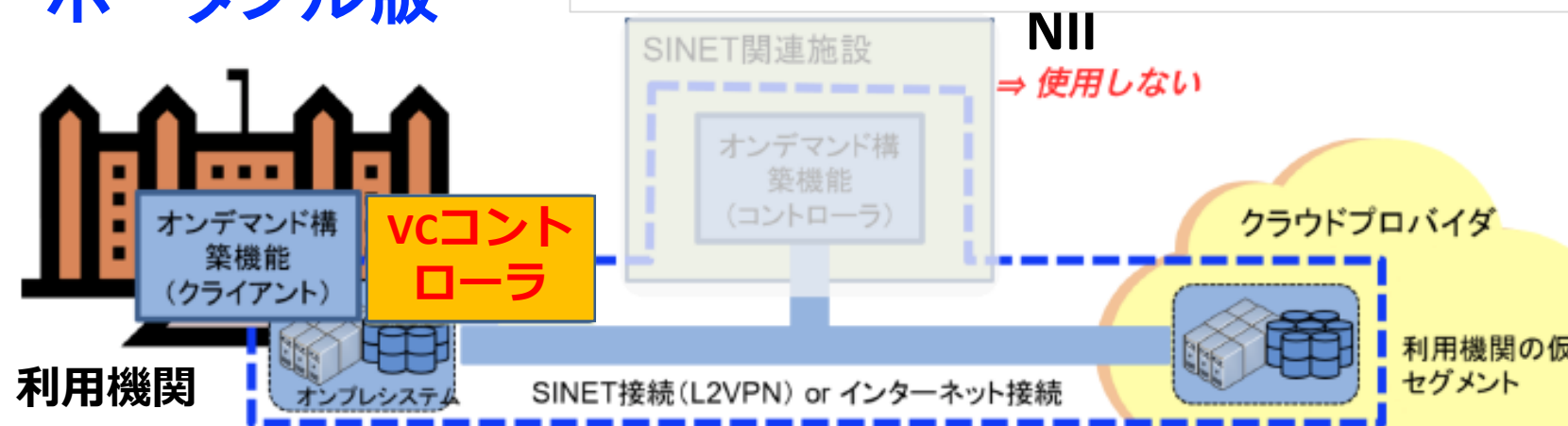
VCP構築申請が不要と  
なり、すぐに利用可

短所：

利用機関側でVCP構築・  
運用・保守

## ポータブル版

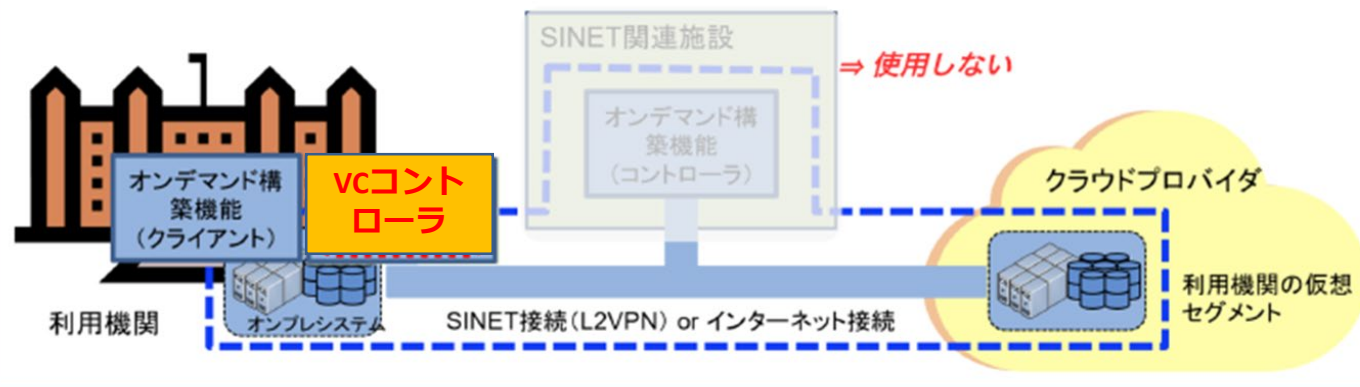
本日のハンズオンはさくら上でポータブル版を使用



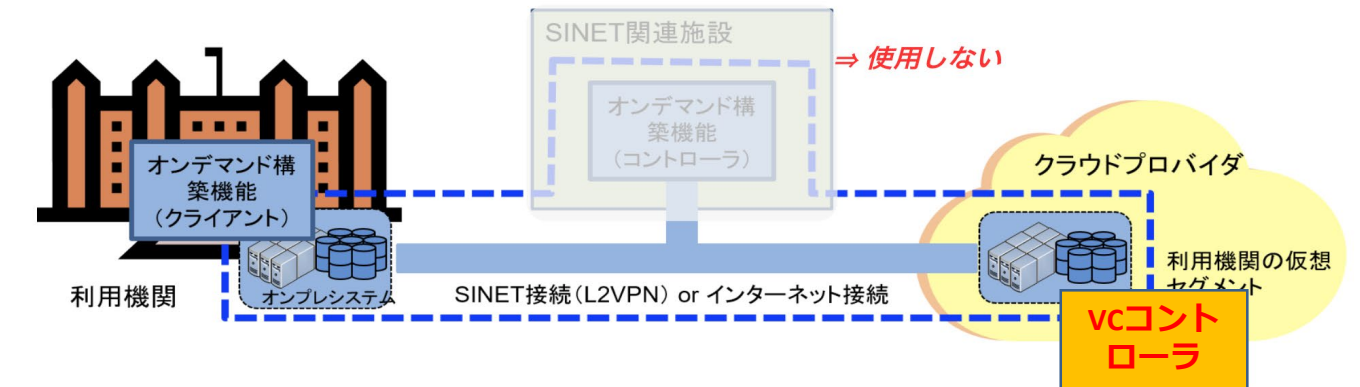


# ポータブル版の構成方法

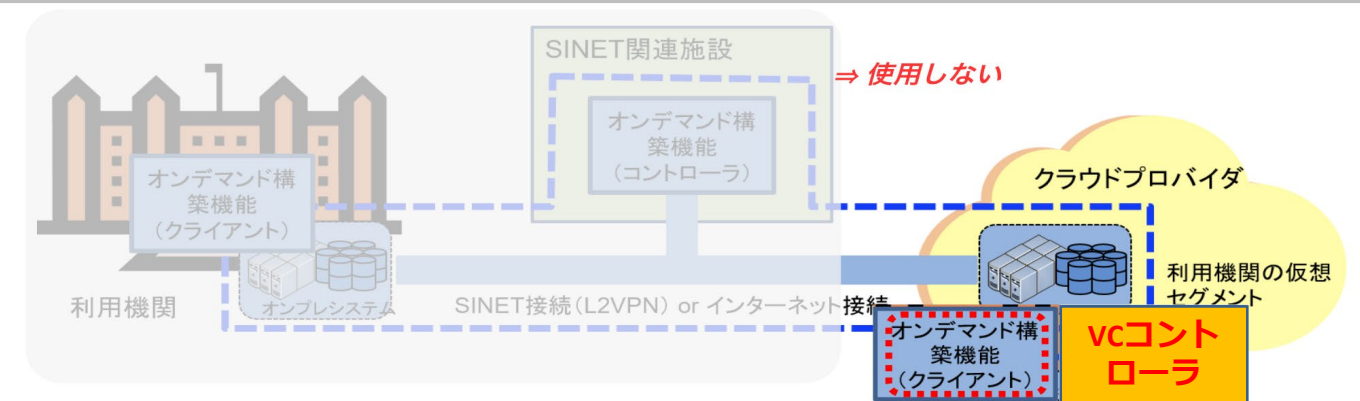
vcコントローラ：利用機関  
JupyterNotebook：利用機関  
(クライアント)



vcコントローラ：クラウド  
JupyterNotebook：利用機関  
(クライアント)

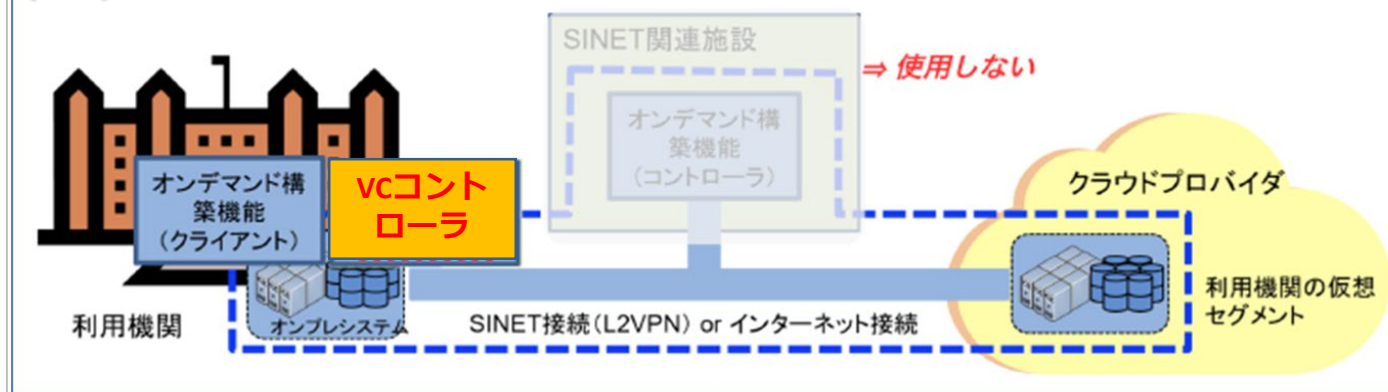


vcコントローラ：クラウド  
JupyterNotebook：クラウド  
(クライアント)

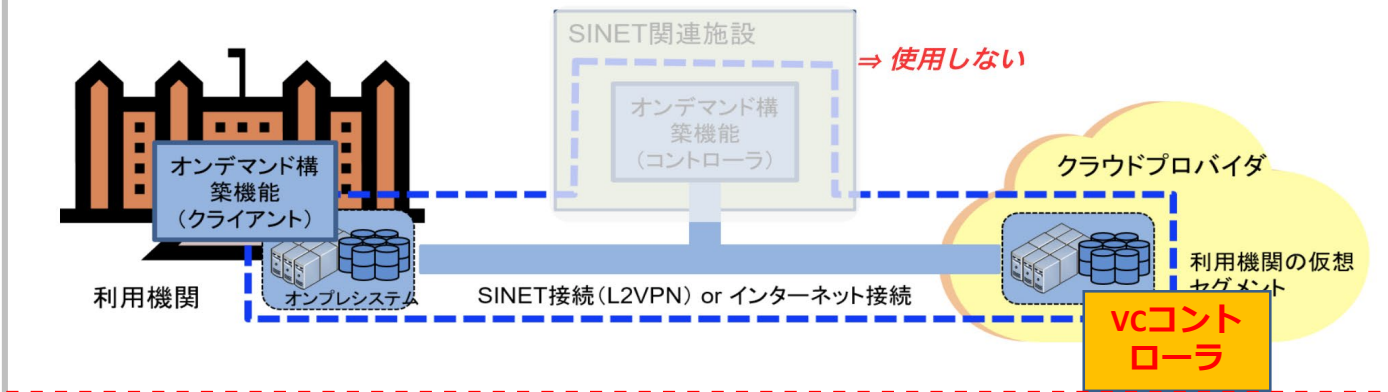


# ポータブル版の構成方法

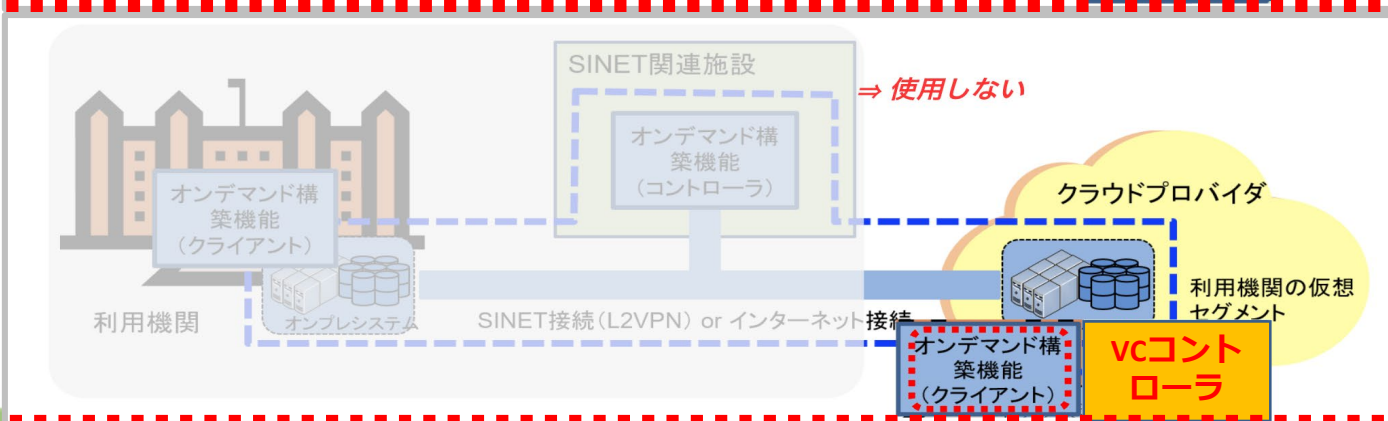
vcコントローラ：利用機関  
JupyterNotebook：利用機関  
(クライアント)



vcコントローラ：クラウド  
JupyterNotebook：利用機関  
(クライアント)



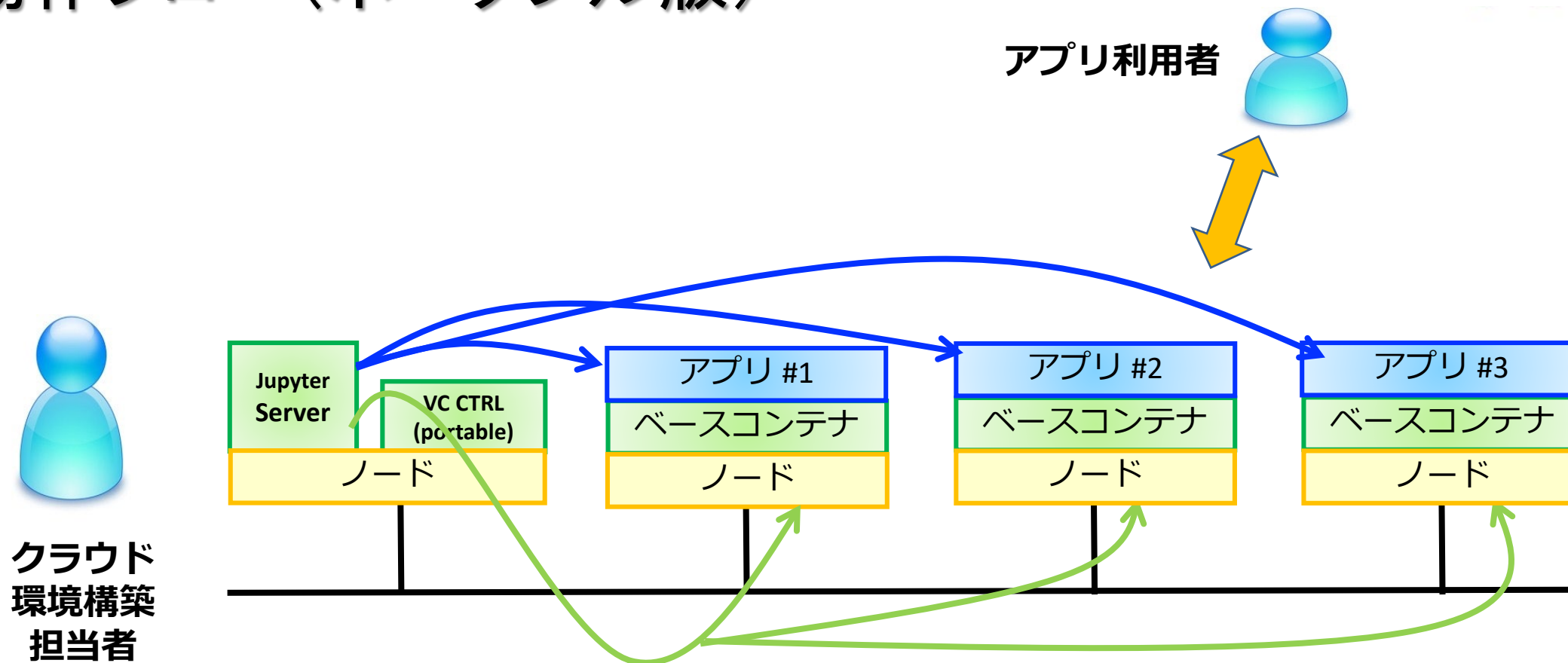
vcコントローラ：クラウド  
JupyterNotebook：クラウド  
(クライアント)



**本日の実習はこの構成！**

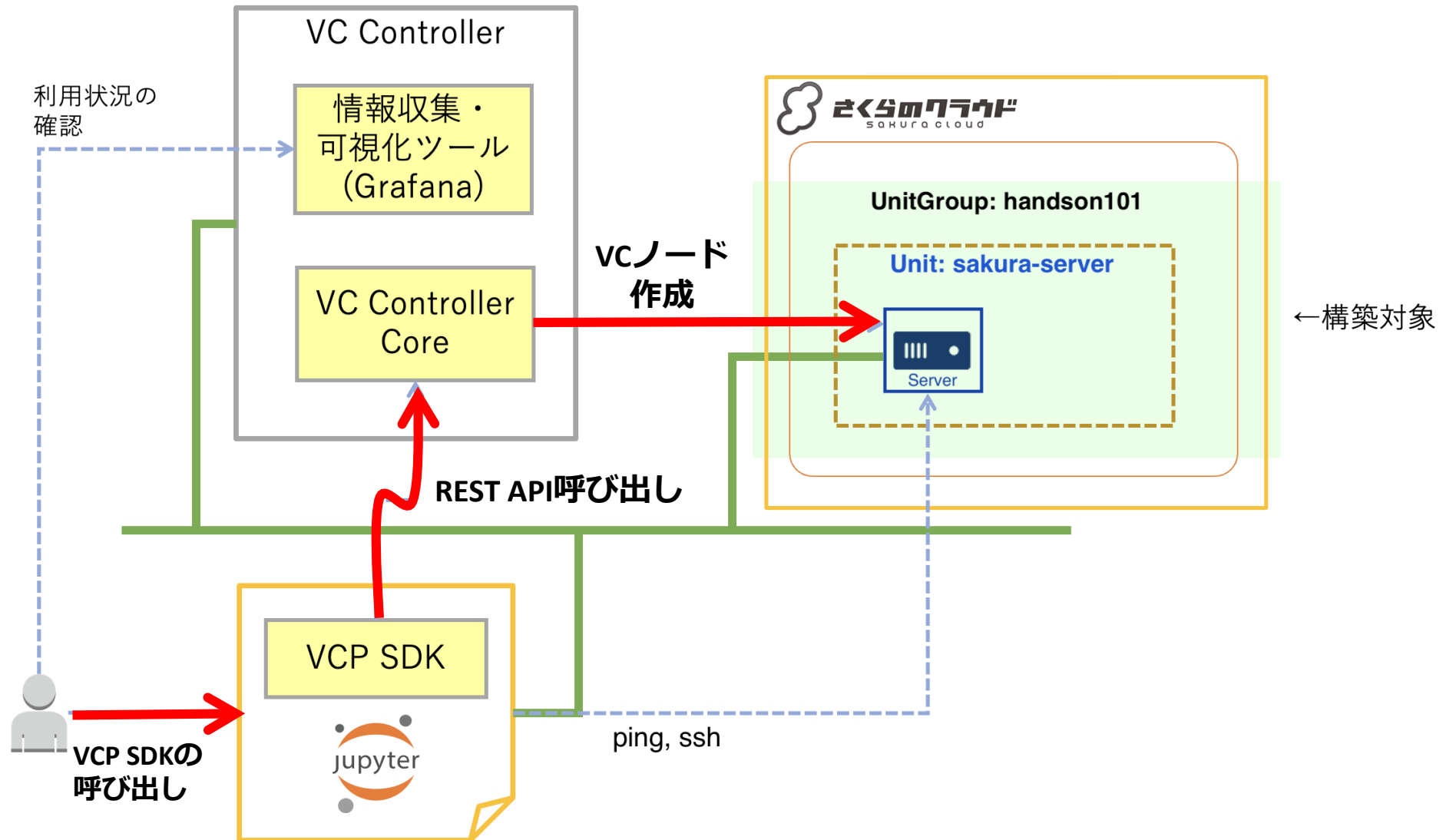


# 動作フロー(ポータブル版)

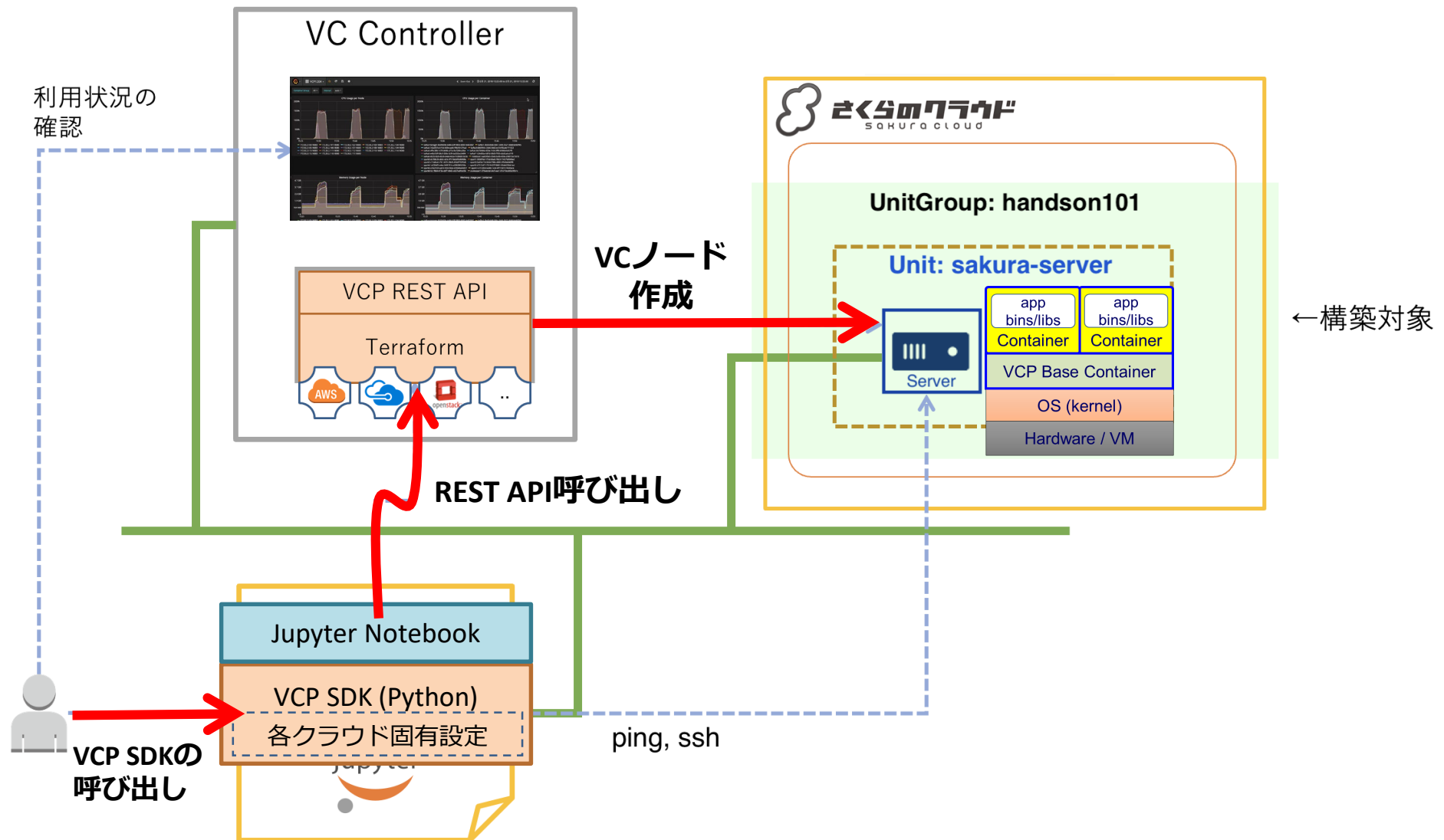


- ① OCSイメージ (Jupyter + VC CTRL) を起動
- ② VC CTRL経由でノードとベースコンテナを起動
- ③ ベースコンテナ上でアプリケーション環境構築

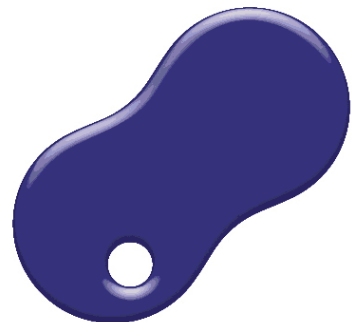
# ハンズオン教材とのマッピング



# ハンズオン教材とのマッピング



各種お問い合わせは、  
NIIクラウド支援室 [cld-office-  
support@nii.ac.jp](mailto:cld-office-support@nii.ac.jp)  
までお願いいたします！



大学共同利用機関法人 情報・システム研究機構

**国立情報学研究所**

**National Institute of Informatics**