学認クラウドオンデマンド構築サービス（OCS）の概要

2022年9月7日

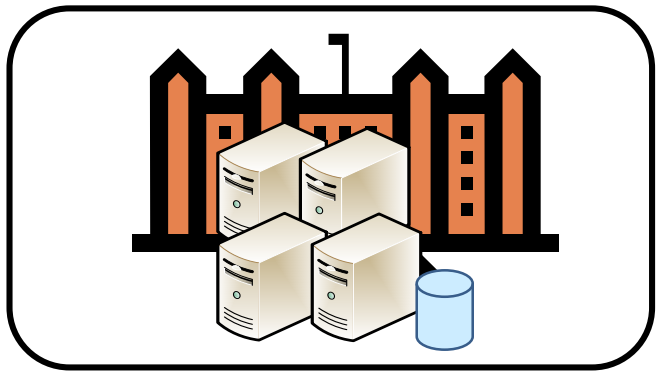
大江　和一

国立情報学研究所
クラウド基盤研究開発センター

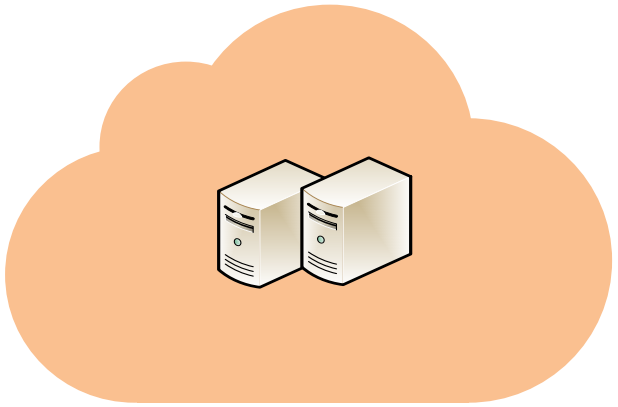
OCSとは

OCS提供の背景（1）

クラウドA



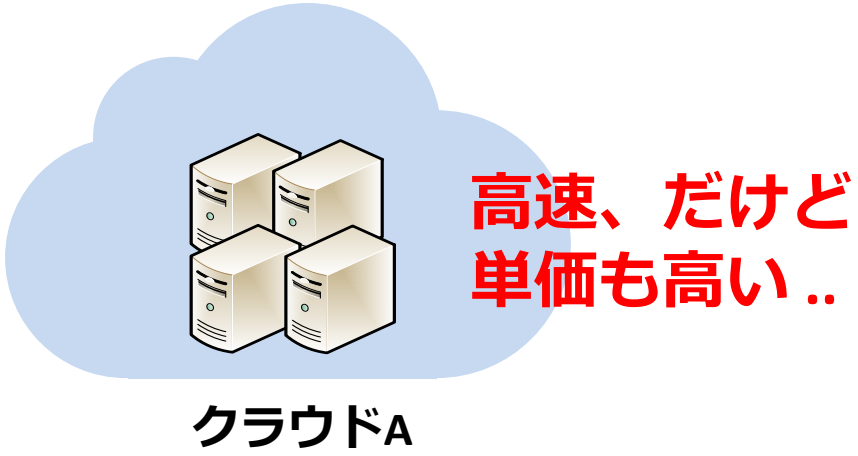
オンプレミス

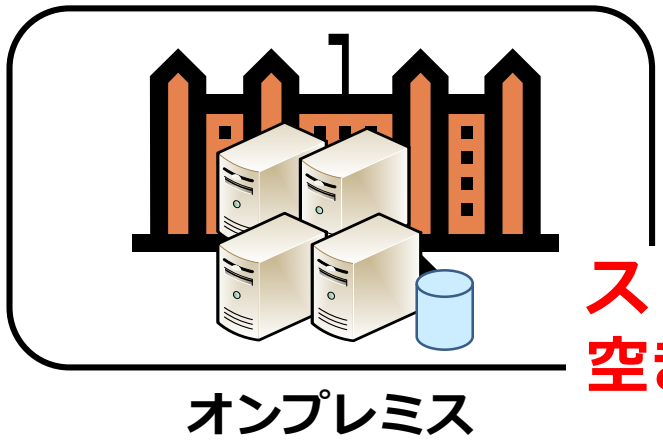


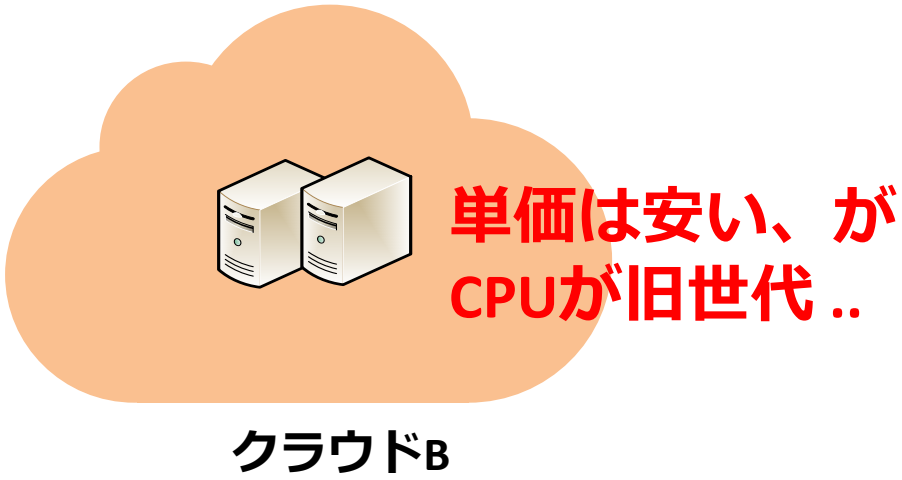
クラウドB

OCS提供の背景(2)

どの環境を選ぶべきか？







OCS提供の背景(3)

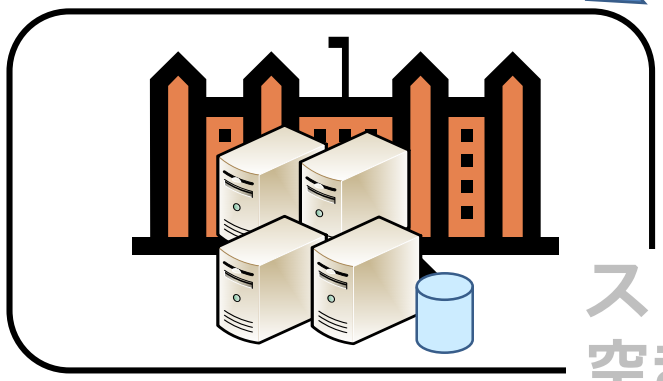
構築方法もバラバラ
↓
一度構築すると、容易に移動できない！

クラウドA API

高速、だけど単価も高い..

クラウドA

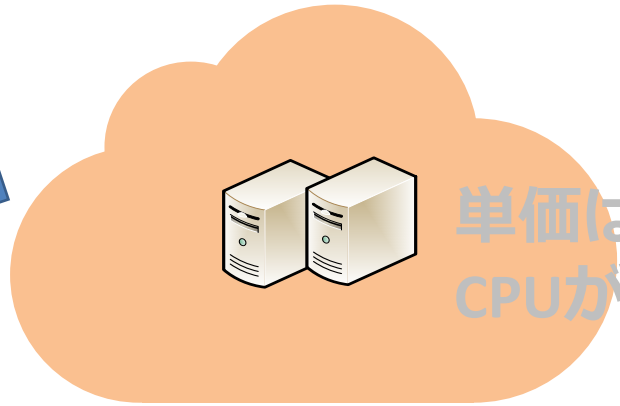
オンプレミス API



ストレージ容量の空きが余りない..

オンプレミス

クラウドB API

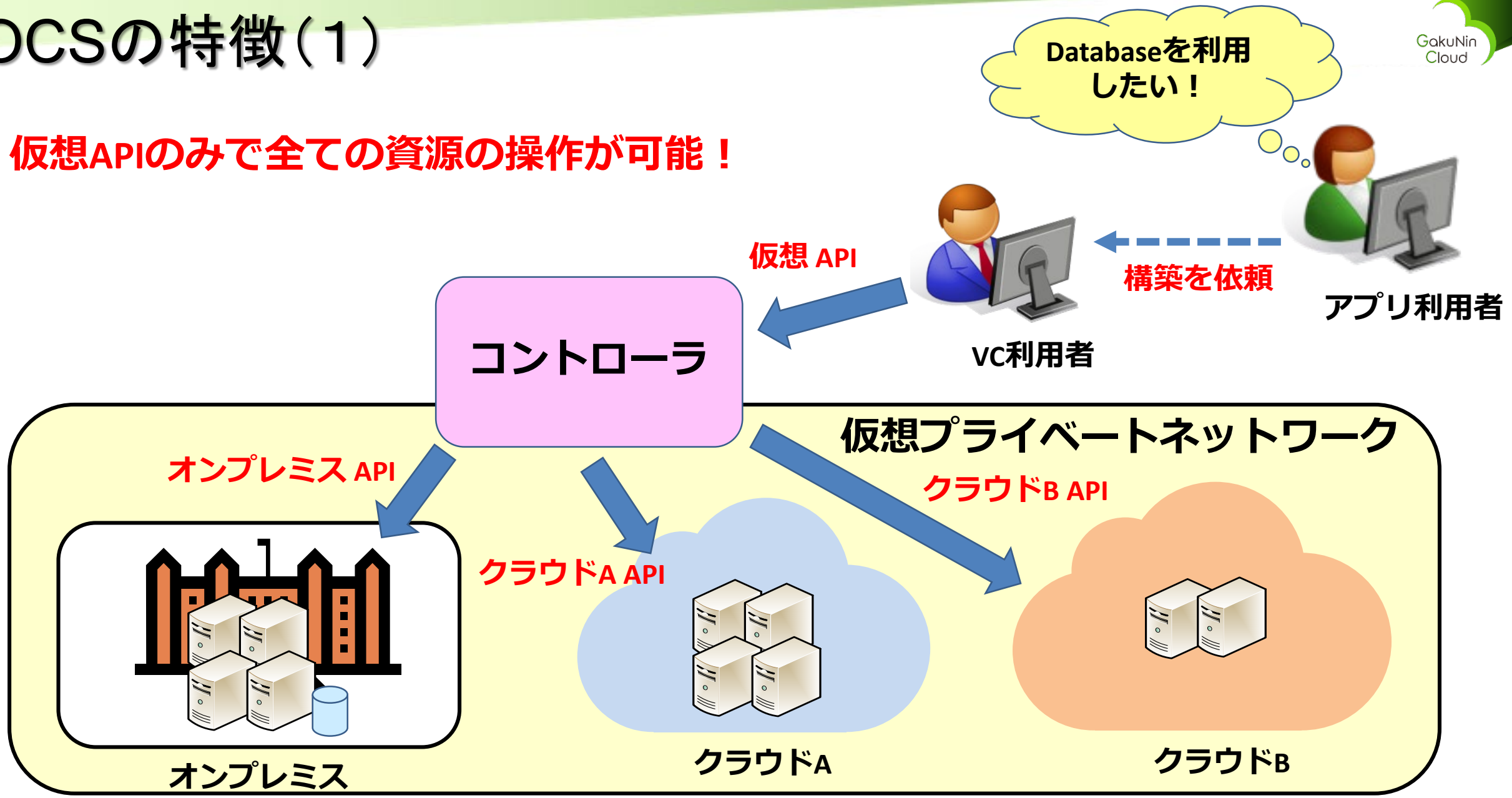


単価は安い、がCPUが旧世代..

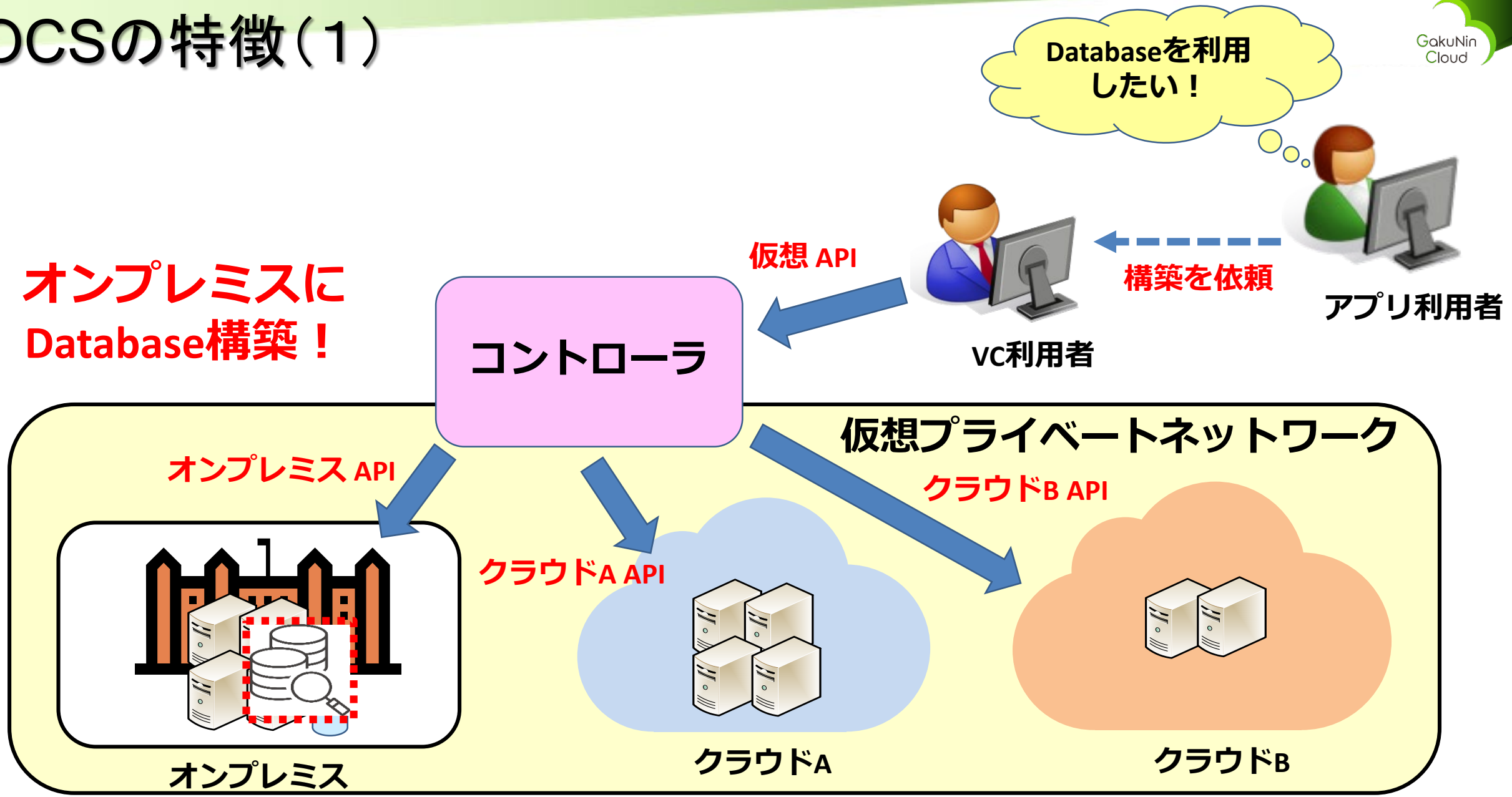
クラウドB

OCSの特徴（1）

仮想APIのみで全ての資源の操作が可能！

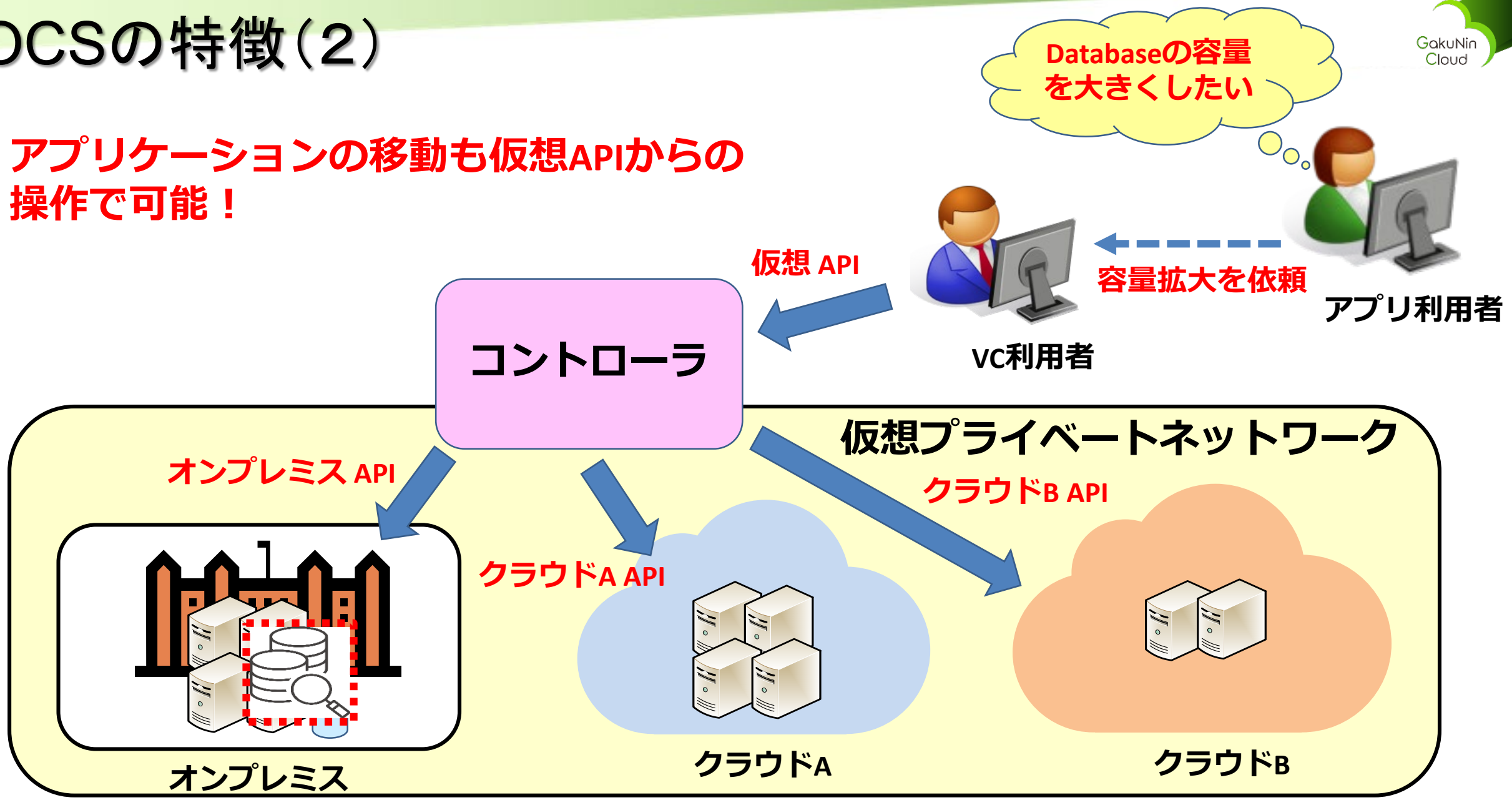


OCSの特徴（1）

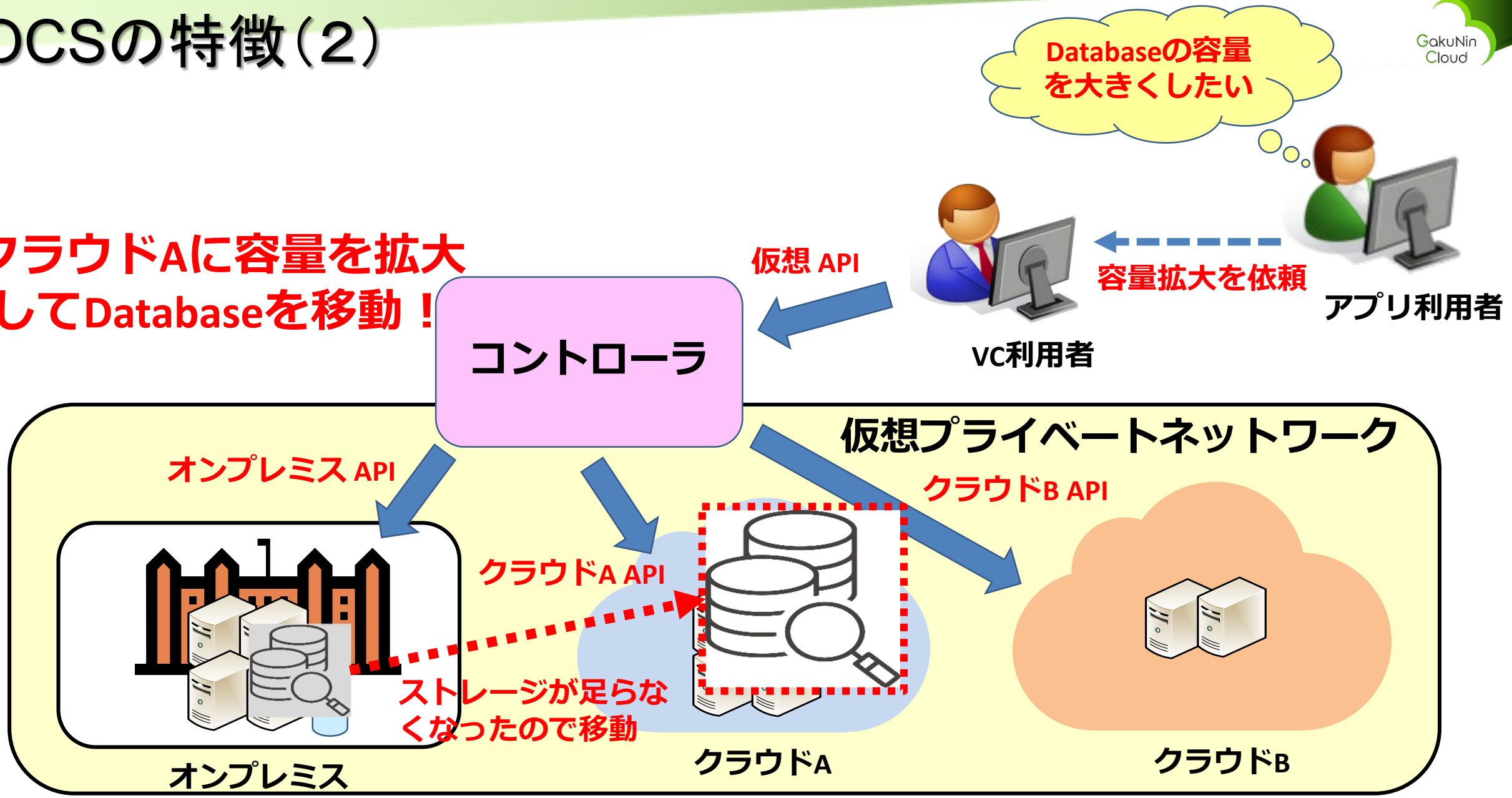


OCSの特徴(2)

アプリケーションの移動も仮想APIからの操作で可能！

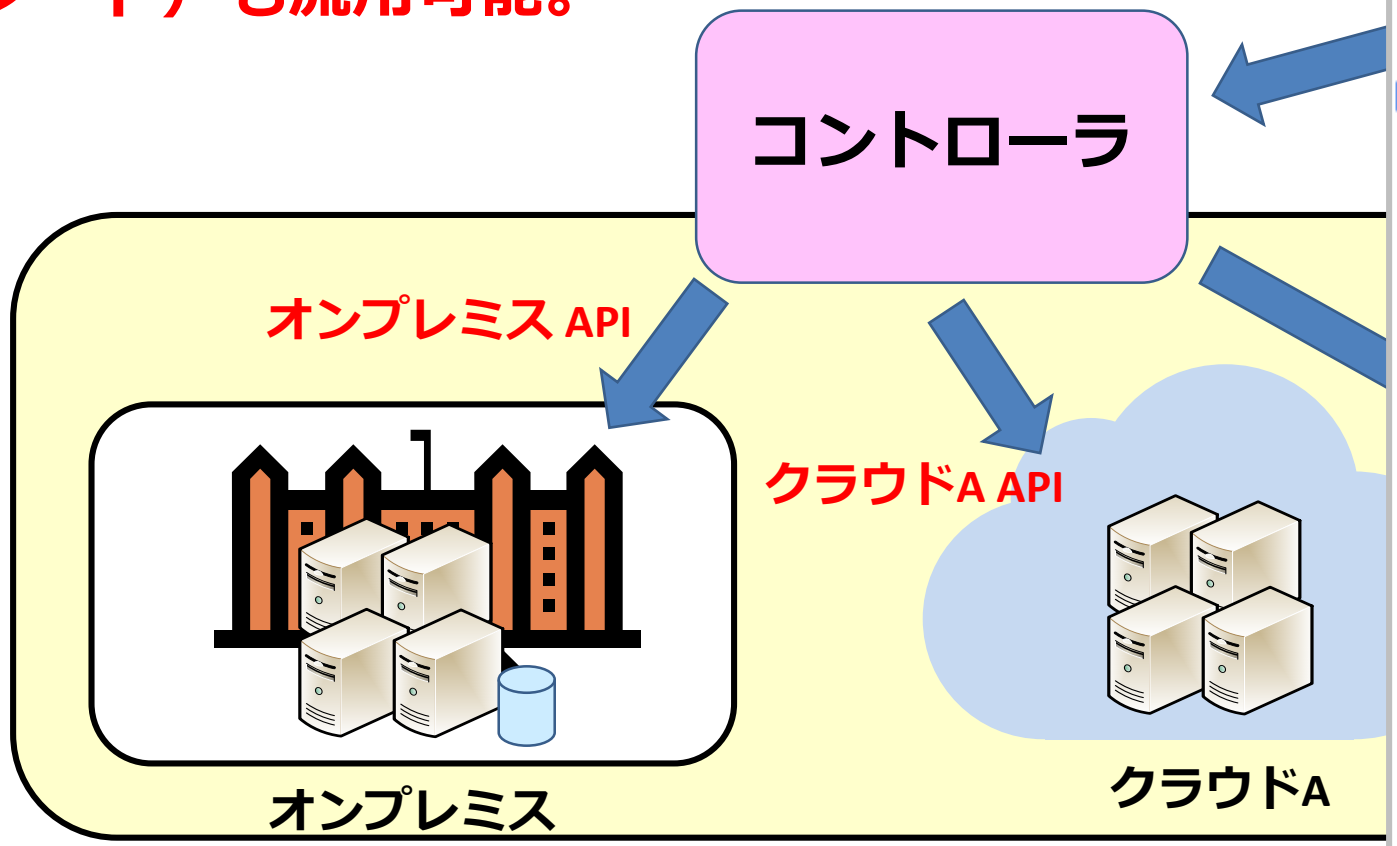


OCSの特徴(2)



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
vcc_access_token = "
testname = "TEST-2022-03-15"
```

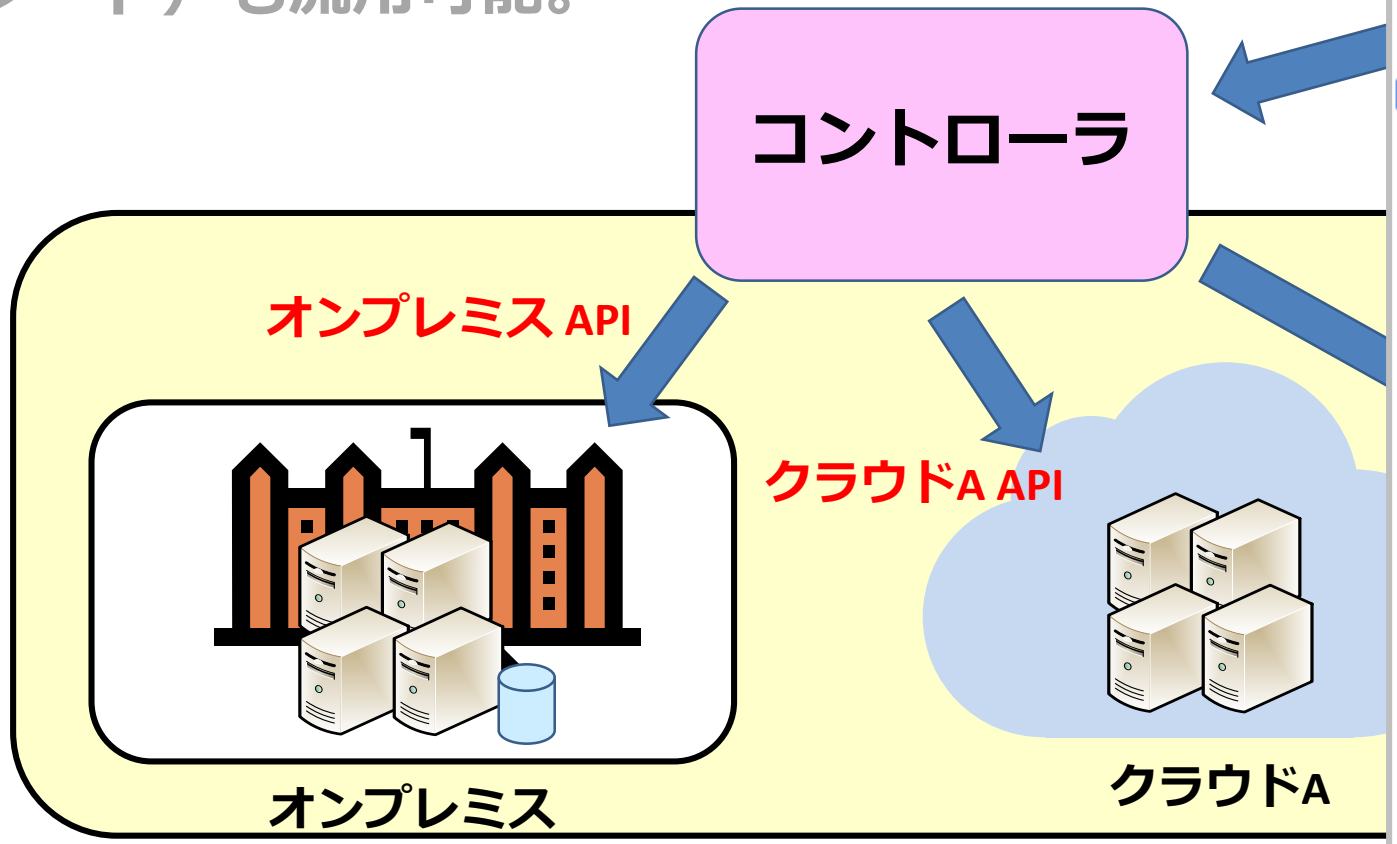
[2]:
```
from common import logsetting
from vcpsdk.vcpsdk import VcpSDK

#
# VCP SDK の初期化
#

sdk = VcpSDK(vcc_access_token)

# VCP SDK バージョン確認
sdk.version()

# UnitGroup作成
my_ugroup_name = "03_sample" + testname

ugroup = sdk.get_ugroup(my_ugroup_name)
if ugroup is None:
    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
vcc_access_token = "
testname = "TEST-2022-03-15"
```

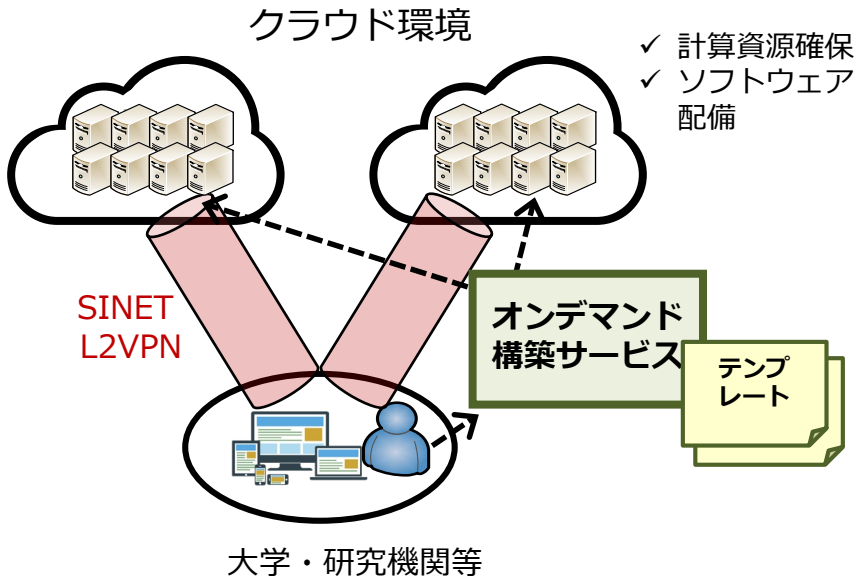
[2]:
```
from common import logsetting
from vcpsdk.vcpsdk import VcpSDK

#
# VCP SDK の初期化
#

sdk = VcpSDK(vcc_access_token)

# VCP SDK バージョン確認
sdk.version()

# UnitGroup作成
my_ugroup_name = "03_sample" + testname

ugroup = sdk.get_ugroup(my_ugroup_name)
if ugroup is None:
    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）

- 概要
  - テンプレート※を使って、クラウド（IaaS）上のアプリケーション実行環境構築を支援するサービス

- 利点
  - クラウド上のアプリ環境の構築・再構築の運用をシンプルにできる
  - 近年求められている研究環境の再現がしやすい
  - オンデマンドに構成変更し再構築できるためコスト低減を図れる
  - オンプレとクラウド、複数のクラウドをまたがる環境も作れる
  - 他者が作ったテンプレートも利用できる
  - 機関とクラウドの接続方法などの相談ができる

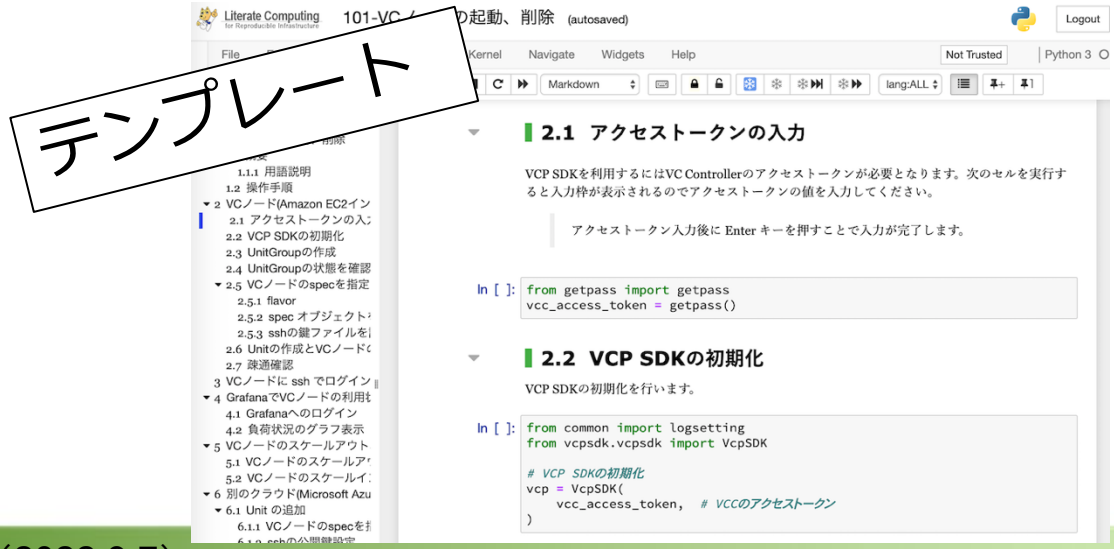


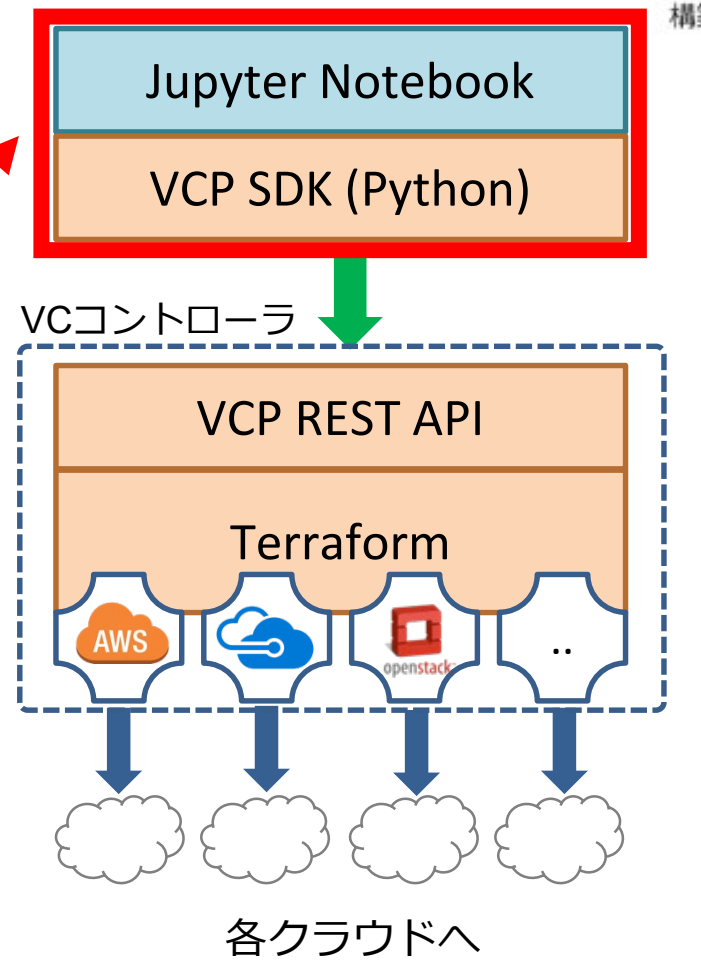
※テンプレート
- アプリ環境の構築ワークフローとドキュメントを記述したファイル
  - 実体は Jupyter Notebook ファイル
  - ドキュメントと構築スクリプトを一体化でき、説明と実態の乖離が起こりにくい
  - 図表、グラフ、画像なども利用可能
  - テンプレート内にスクリプトの実行結果も残しておくことが可能

OCSを支えるVCPの仕組み

管理ソフトウエアの概要(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





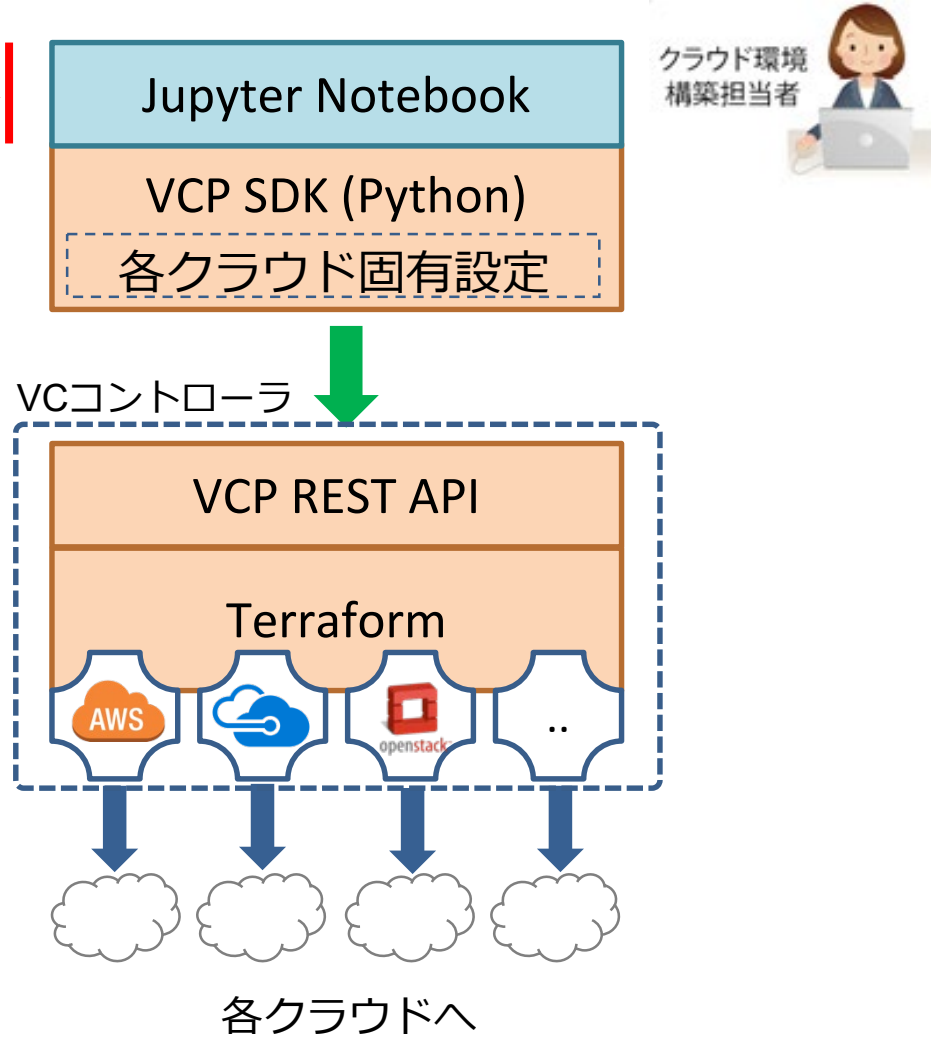
管理ソフトウエアの概要（2）

- 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

1.2 新規 server を作成

[5]:

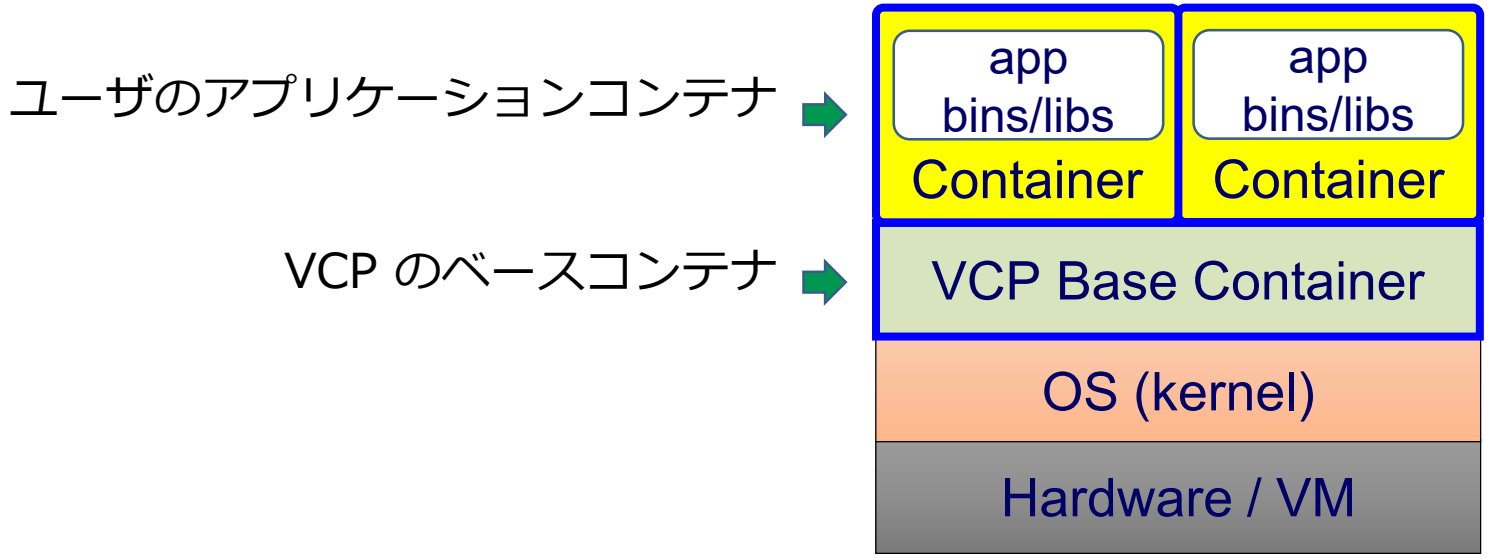
```
#
# 作成するserverのspec情報を作成
#
spec = sdk.get_spec("aws", "medium")

#
# 変更できること
#
# spec.num_nodes = 1
spec.num_nodes = 2
#spec.instance_type = 't2.medium'
# spec.params_v = ['/opt:/opt']
# spec.volume_size = 40
# spec.volume_type = "standard"  # standard|io1|gp2|sc1|st1
# spec.ip_addresses = ['起動するnodeの静的なIPアドレス']
# spec.image = 'vcp/base:1.5'   # base container
# 追加で使用するVolume
# spec.disks = ['vol-08cbb04b35c8c9545']

# cloud上のタグ設定
spec.set_tag('key1', 'value1')
spec.set_tag('key2', 'value2')
```



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

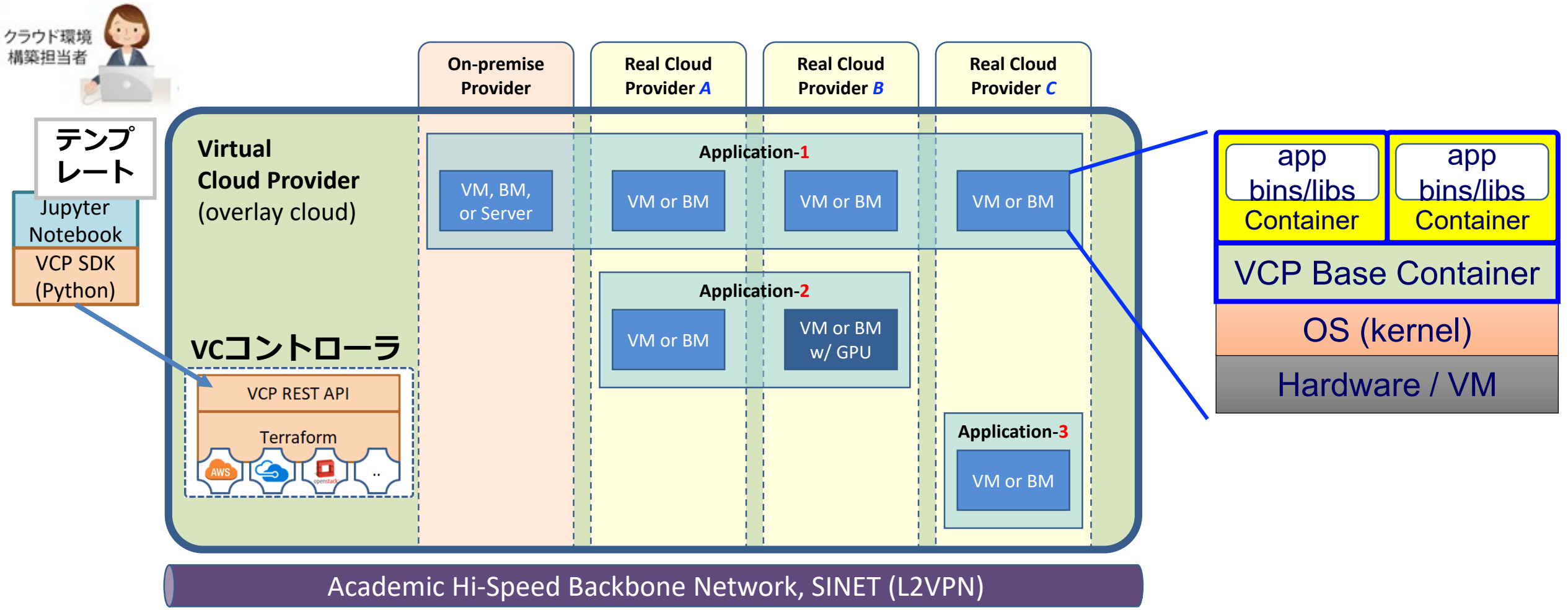
- ベースコンテナ、アプリコンテナのモニタリング情報を提供
- アプリケーションの収容設計を支援

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

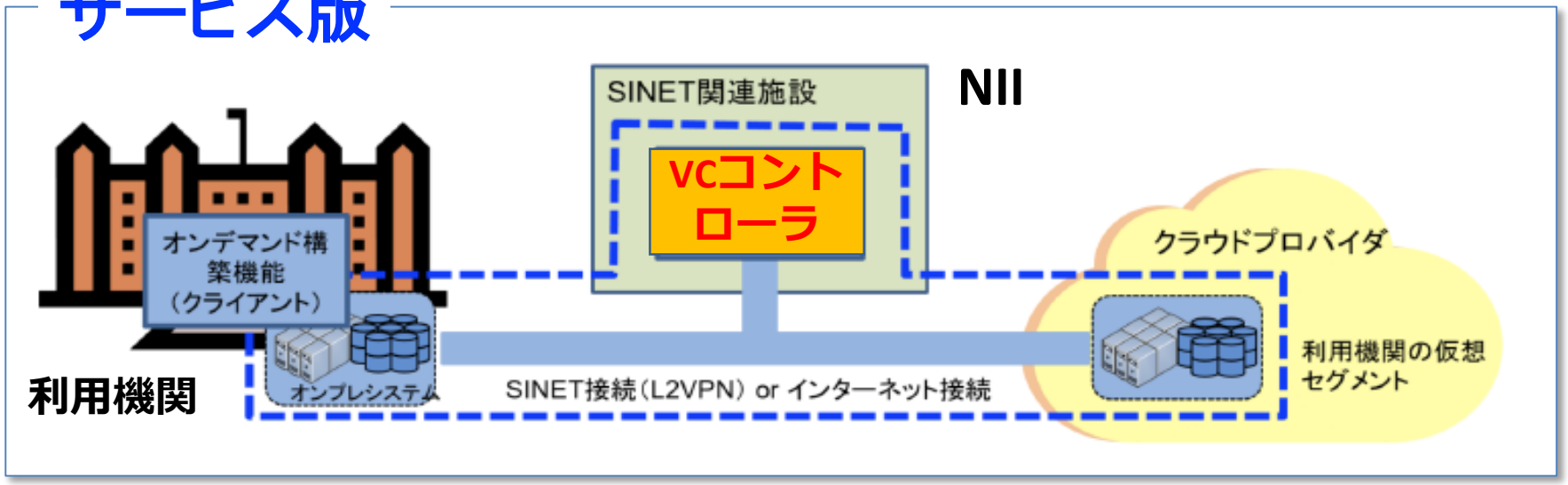
- オンプレ・複数の実クラウドを跨ってのアプリケーション配備が可能！



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

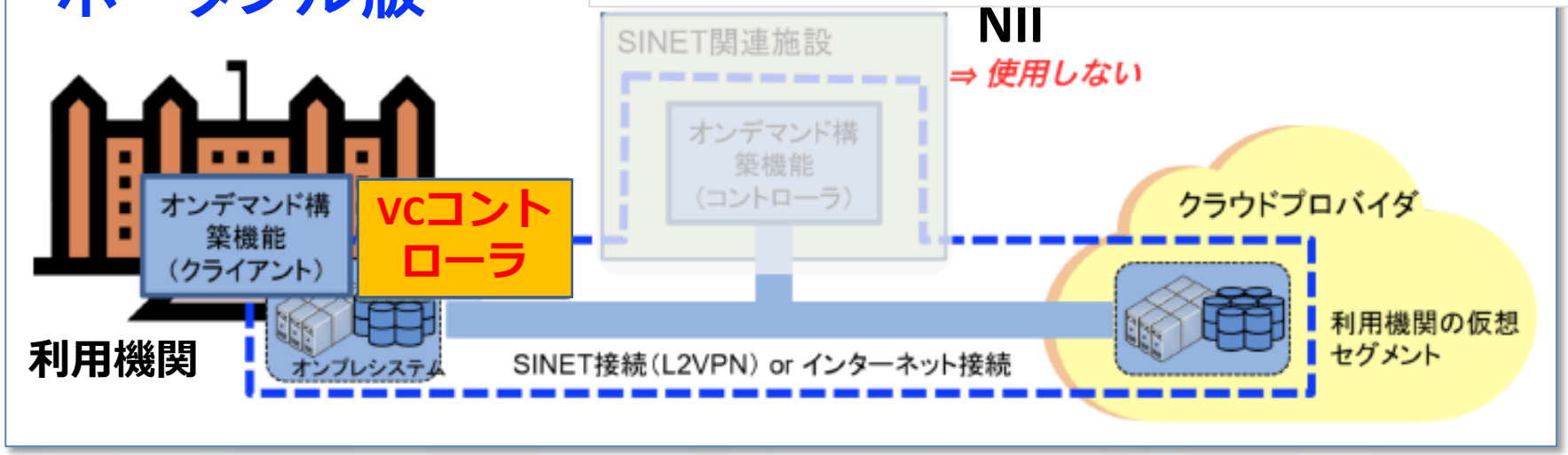
サービス版



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

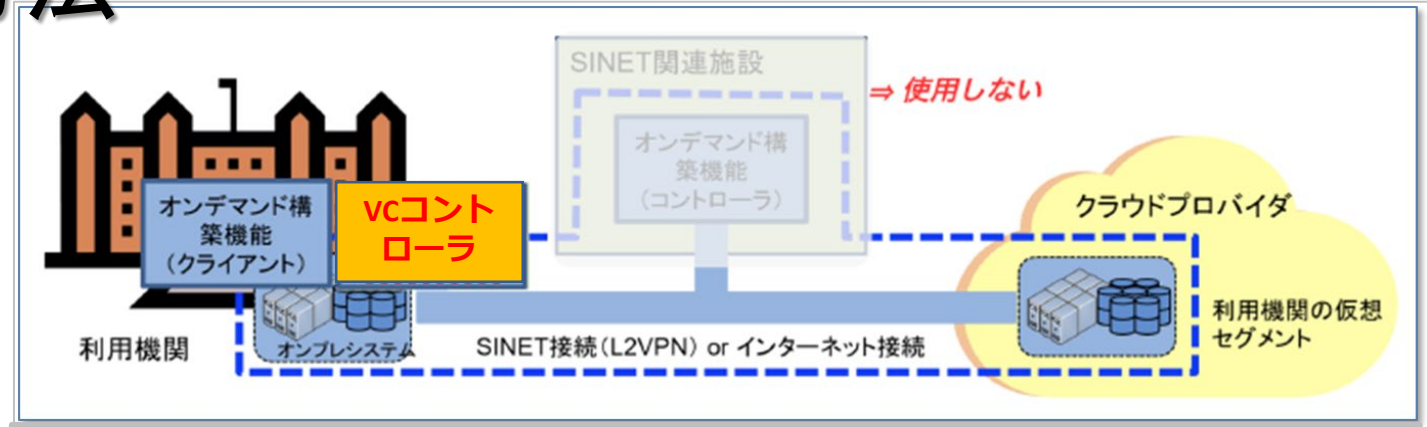
ポータブル版

本日のハンズオンはmdx上でポータブル版を使用

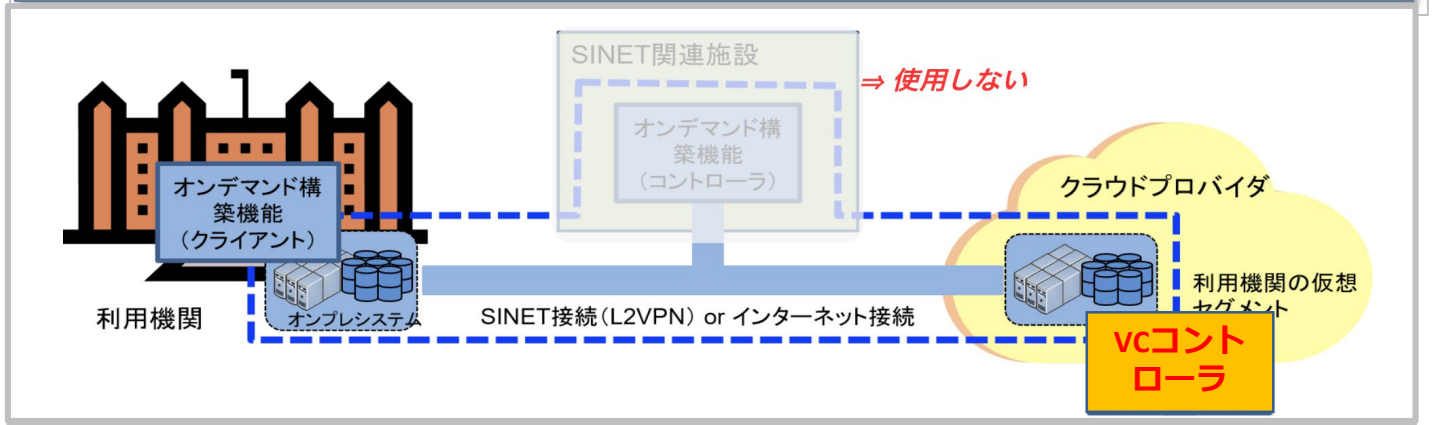


ポータブル版の構成方法

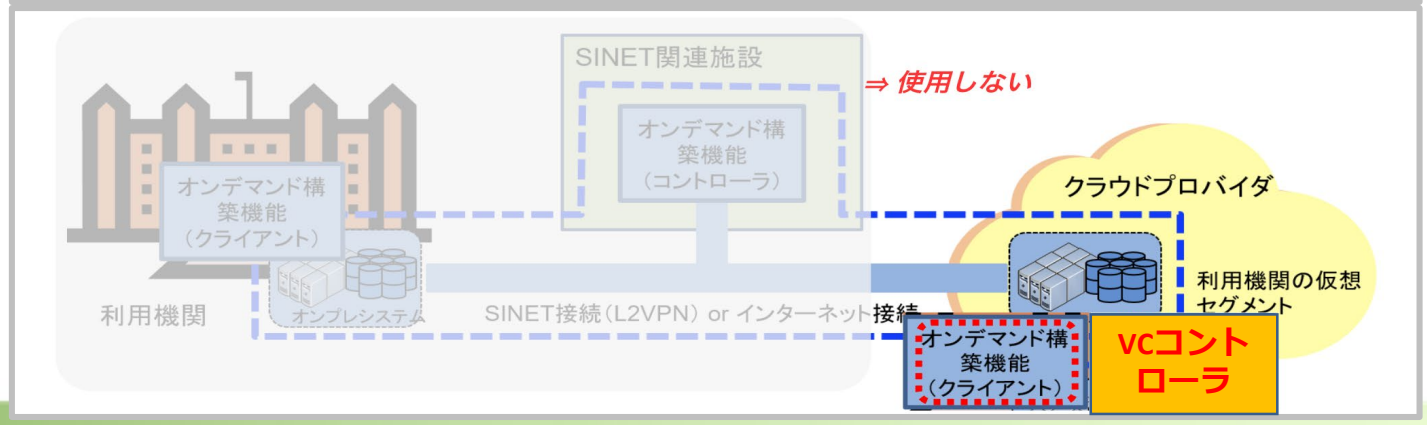
vcコントローラ： 利用機関
JupyterNotebook： 利用機関
（クライアント）



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）



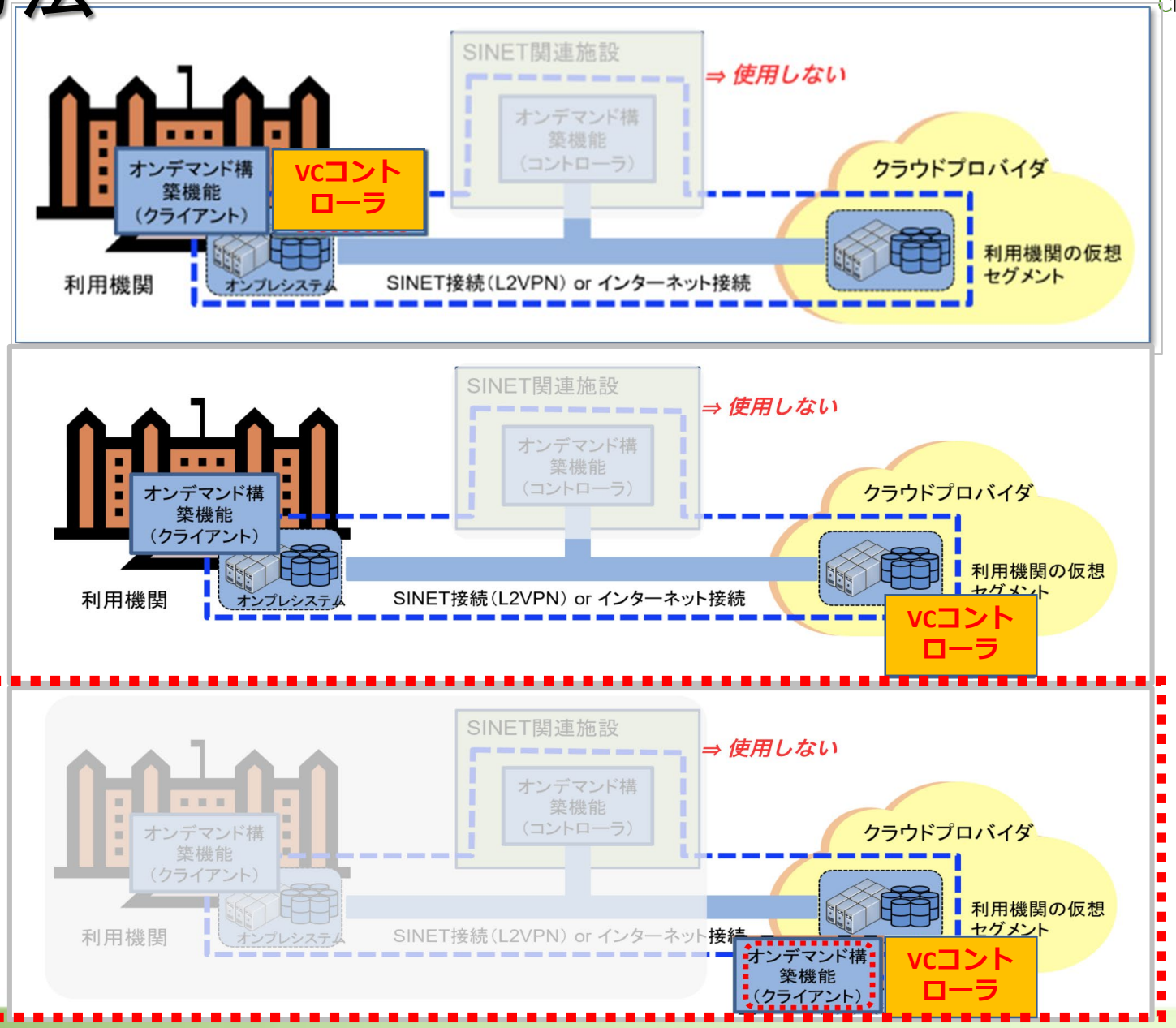
ポータブル版の構成方法

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

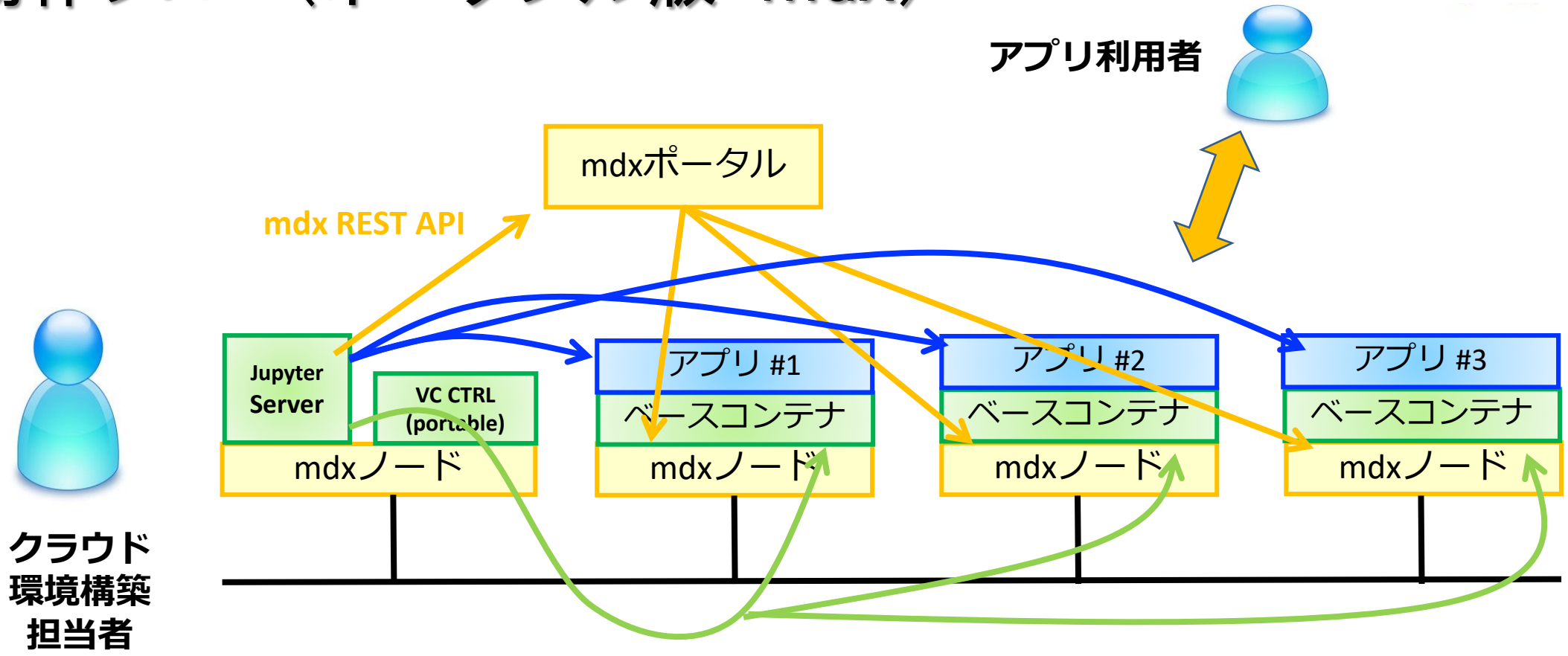
vcコントローラ： クラウド
JupyterNotebook： 利用機関
（クライアント）

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+mdx）



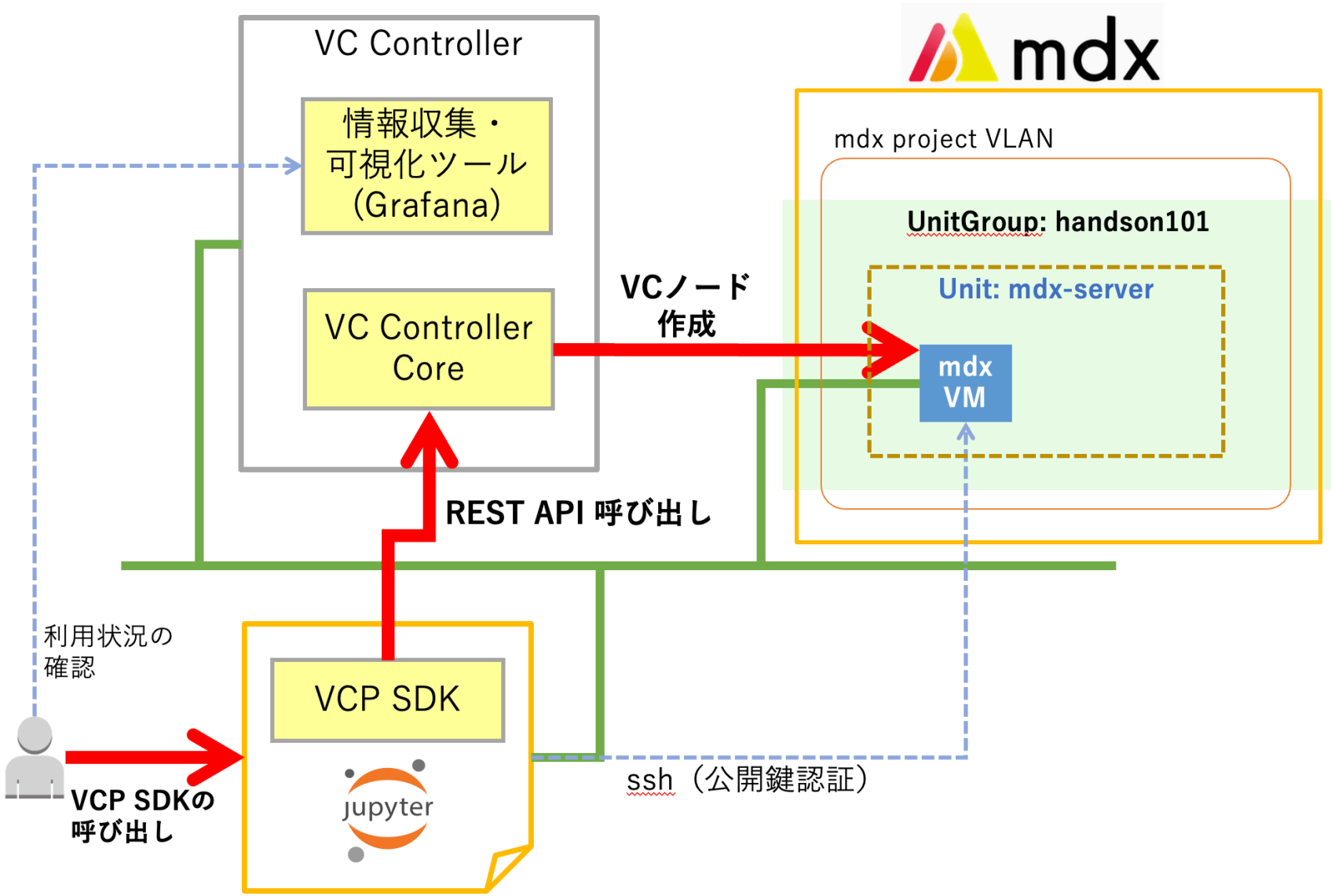
① OCSイメージ（Jupyter + VC CTRL）を起動

② mdx REST APIでOCS用のノードを確保

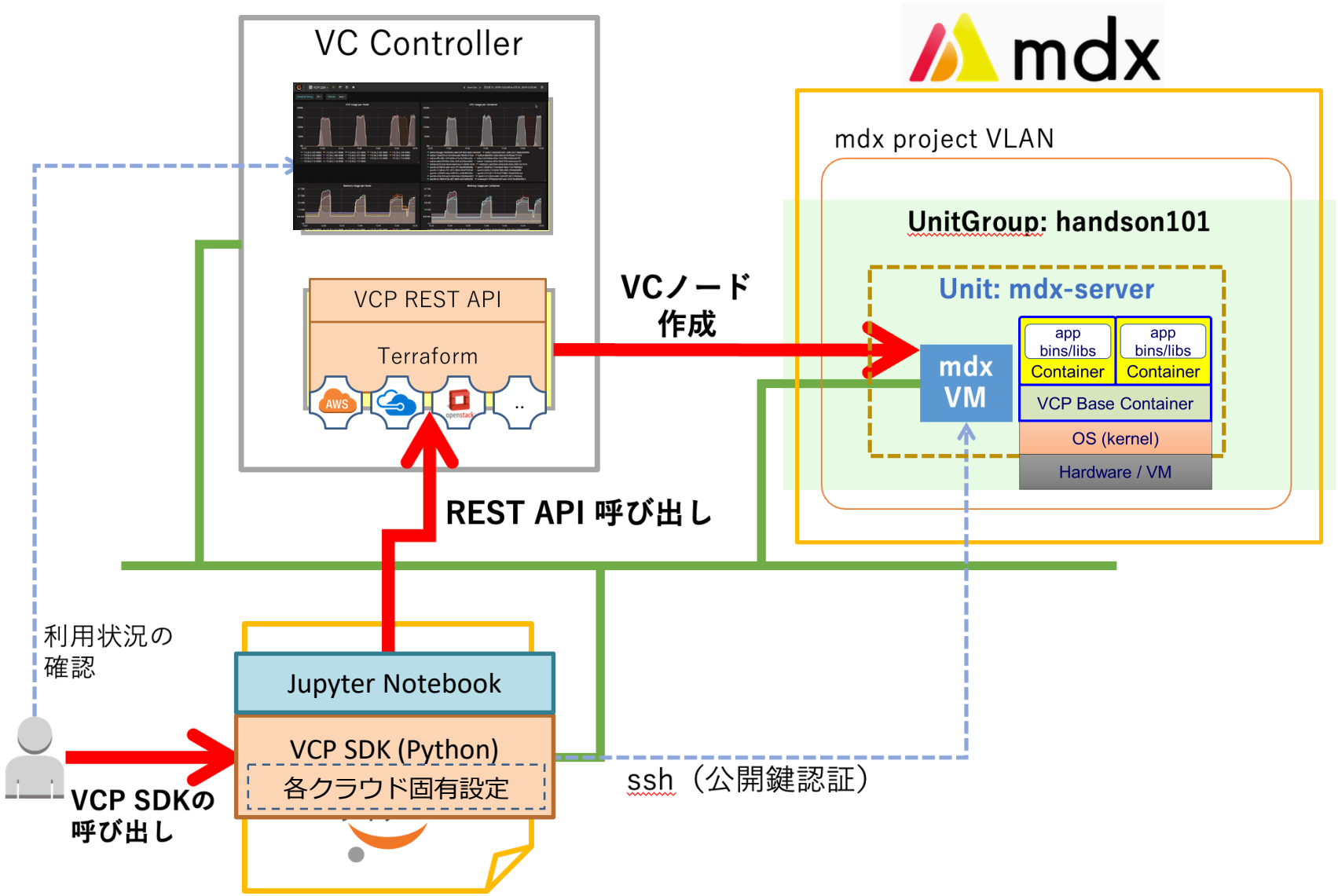
③ VC CTRL経由でベースコンテナを起動

④ 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

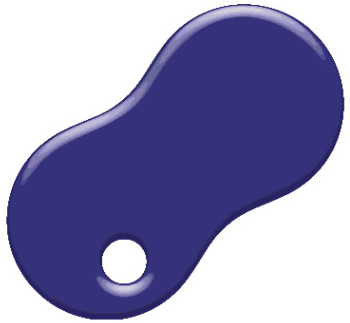


ハンズオン教材とのマッピング



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

GakuNin Cloud



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