学認クラウドオンデマンド構築サービス（OCS）とOpen OnDemandの概要

２０２５年１０月０６日

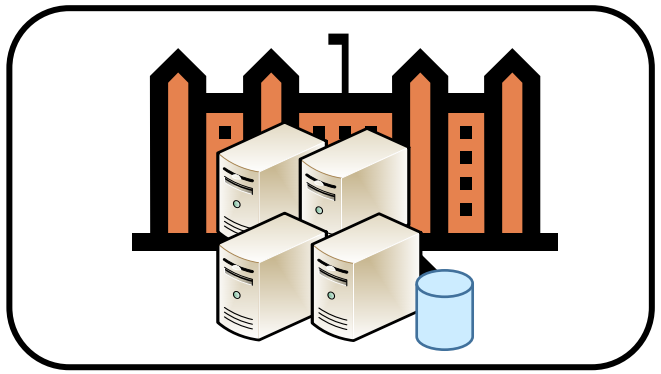
大江　和一

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



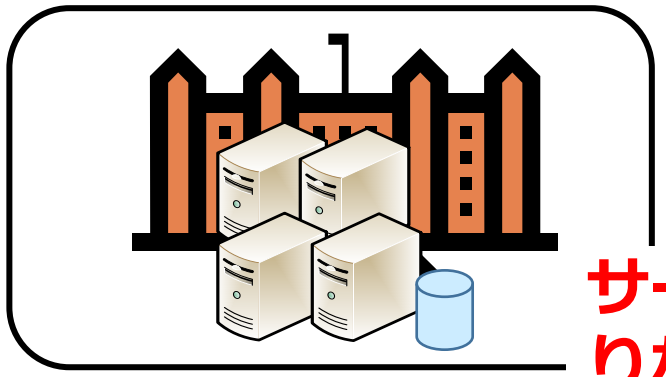
オンプレミス

クラウドB

OCS提供の背景（２）

JupyterHubを用いてPython演習環境を立ち上げたい

どの環境を選ぶべきか？

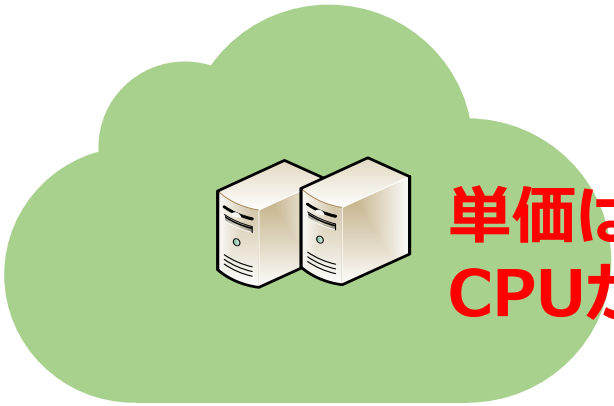


オンプレミス

サーバの空きが余りない ..

クラウドA

高速、だけど単価も高い ..



クラウドB

単価は安い、がCPUが旧世代 ..

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！

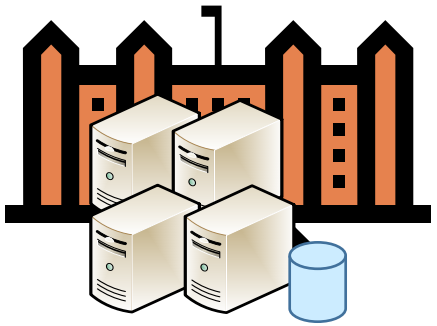
JupyterHubを用いてPython演習環境を立ち上げたい

クラウドA API

クラウドA

高速、だけど単価も高い ..

オンプレミス API



オンプレミス

サーバの空きが余りない ..

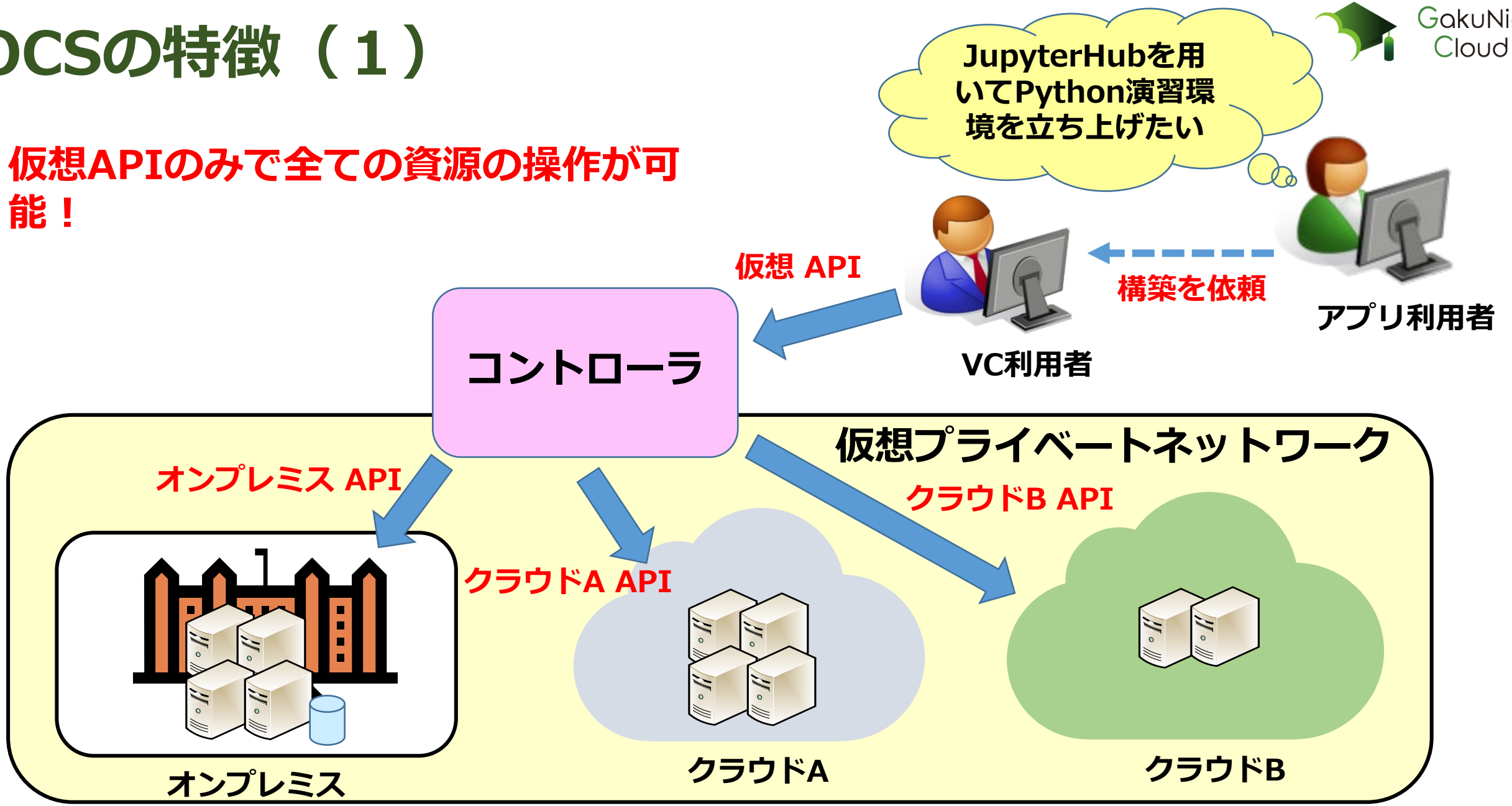
クラウドB API

クラウドB

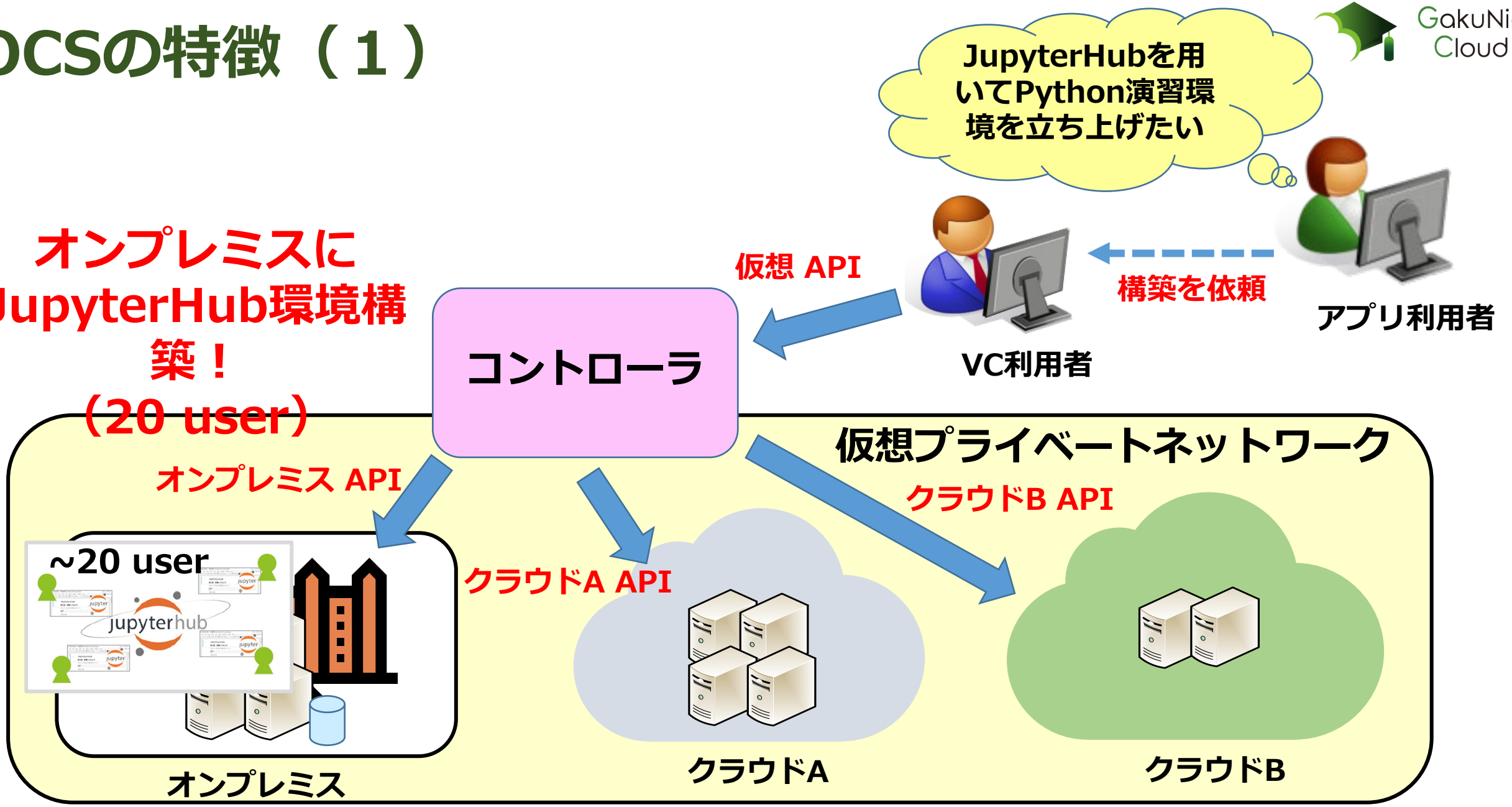
単価は安い、がCPUが旧世代 ..

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）

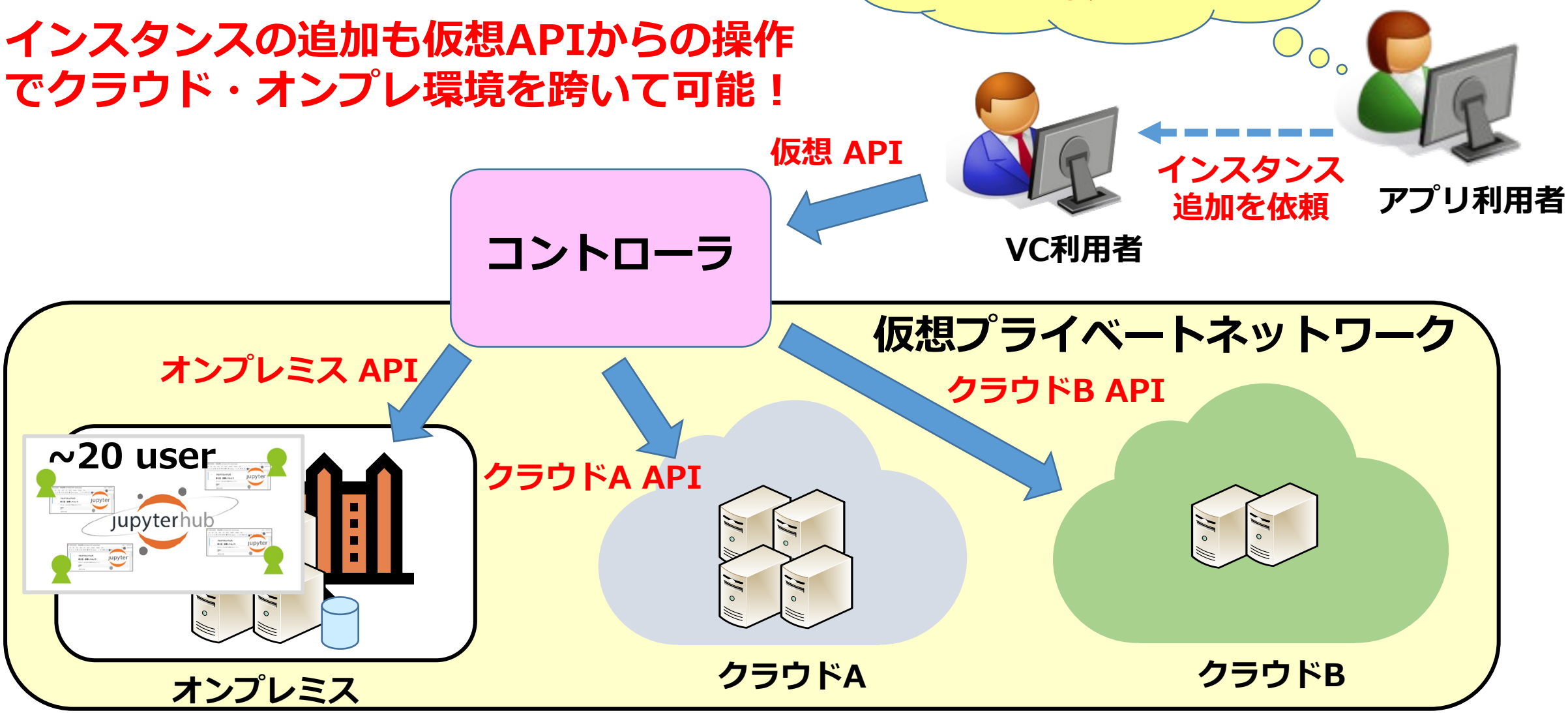

JupyterHubを用いてPython演習環境を立ち上げたい
アプリ利用者
構築を依頼
VC利用者
仮想 API
コントローラ
オンプレミスにJupyterHub環境構築！（20 user）
仮想プライベートネットワーク
オンプレミス API
クラウドA API
クラウドB API
~20 user
jupyterhub
オンプレミス
クラウドA
クラウドB

OCSの特徴（２）

GakuNin Cloud

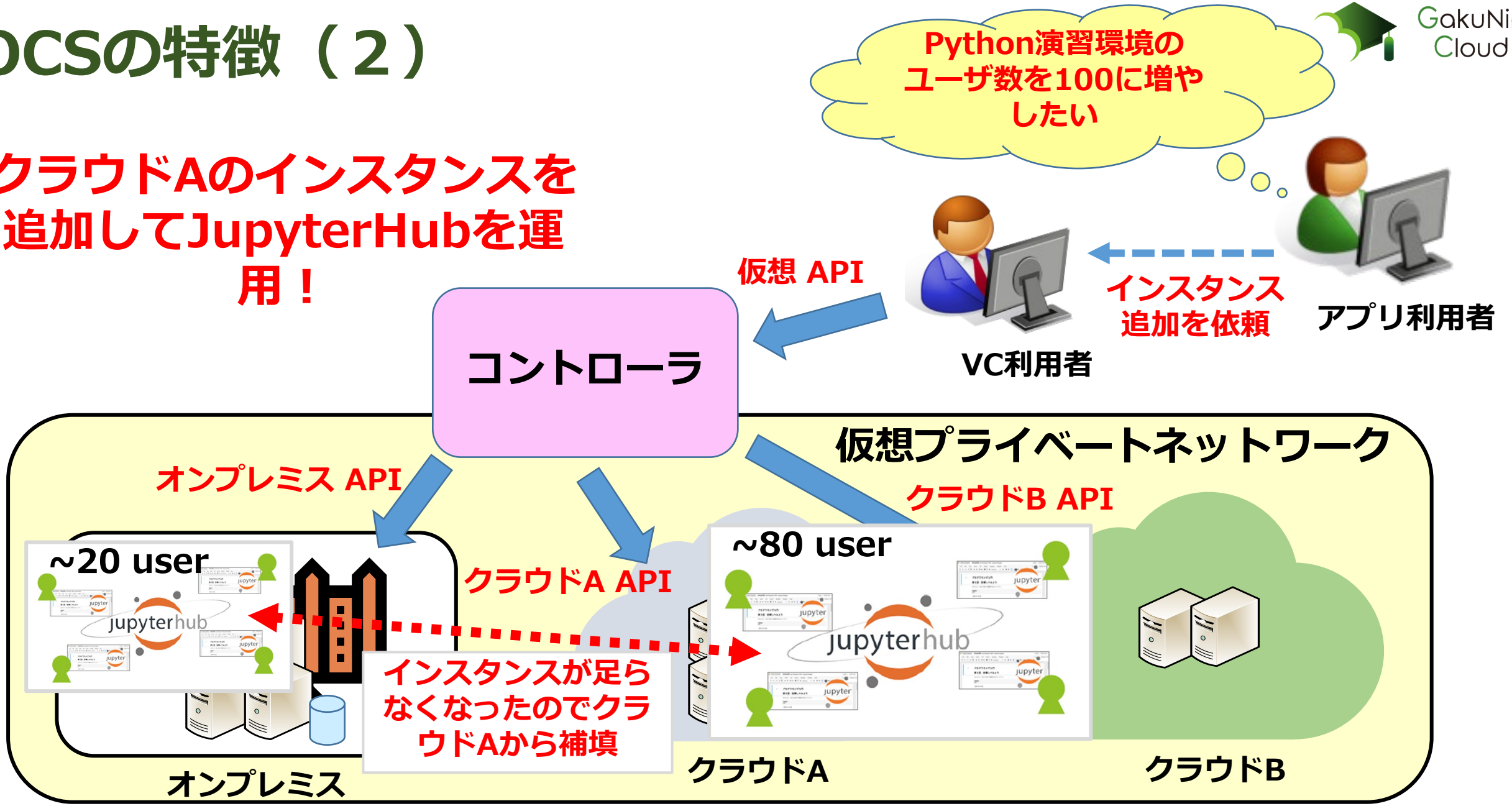
Python演習環境のユーザ数を100に増やしたい

インスタンスの追加も仮想APIからの操作でクラウド・オンプレ環境を跨いで可能！



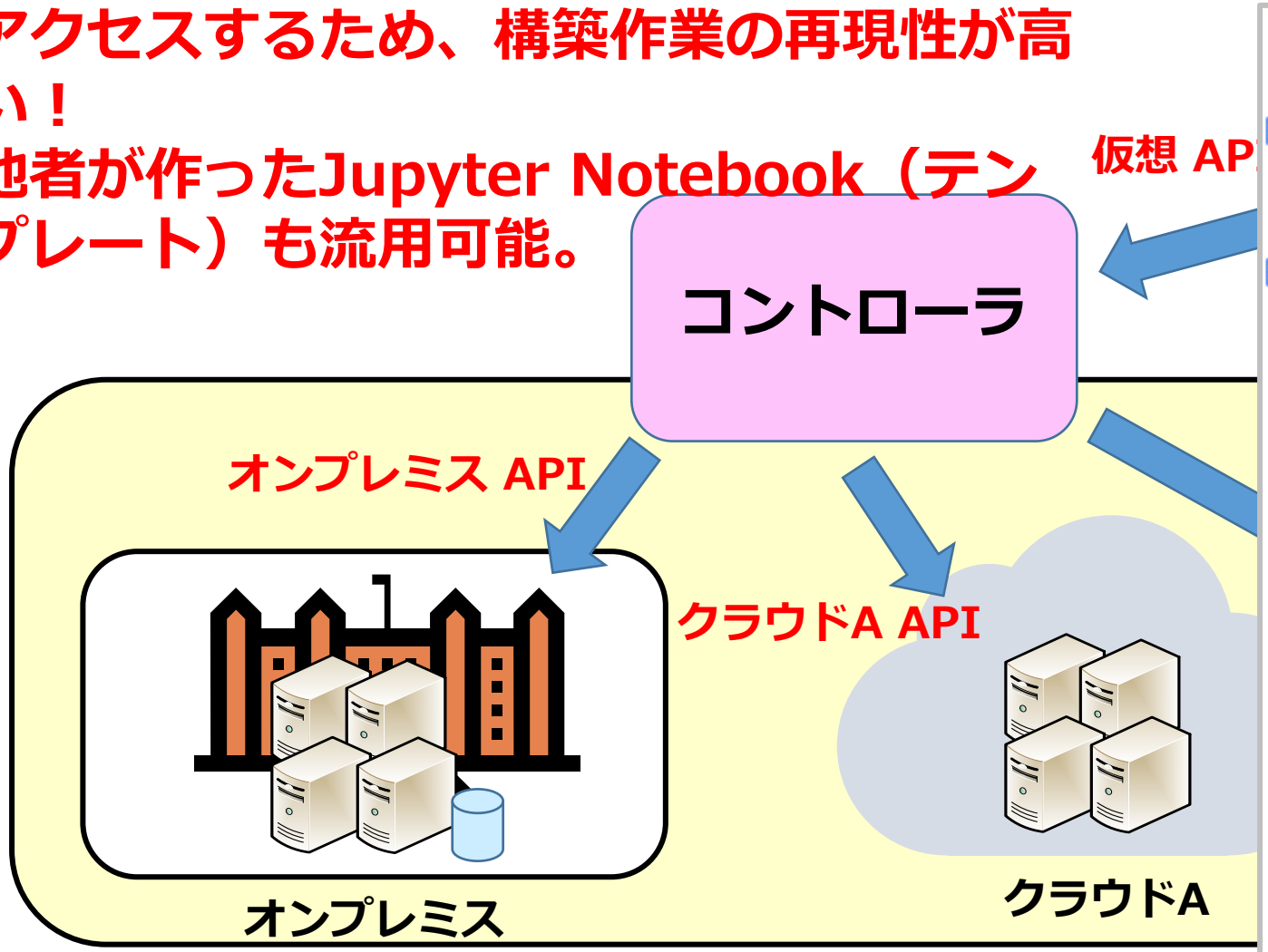
OCSの特徴（２）

クラウドAのインスタンスを追加してJupyterHubを運用！



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。



Jupyter Notebookの記述例

1.1 初期化

[1]:
parameters

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

[2]:

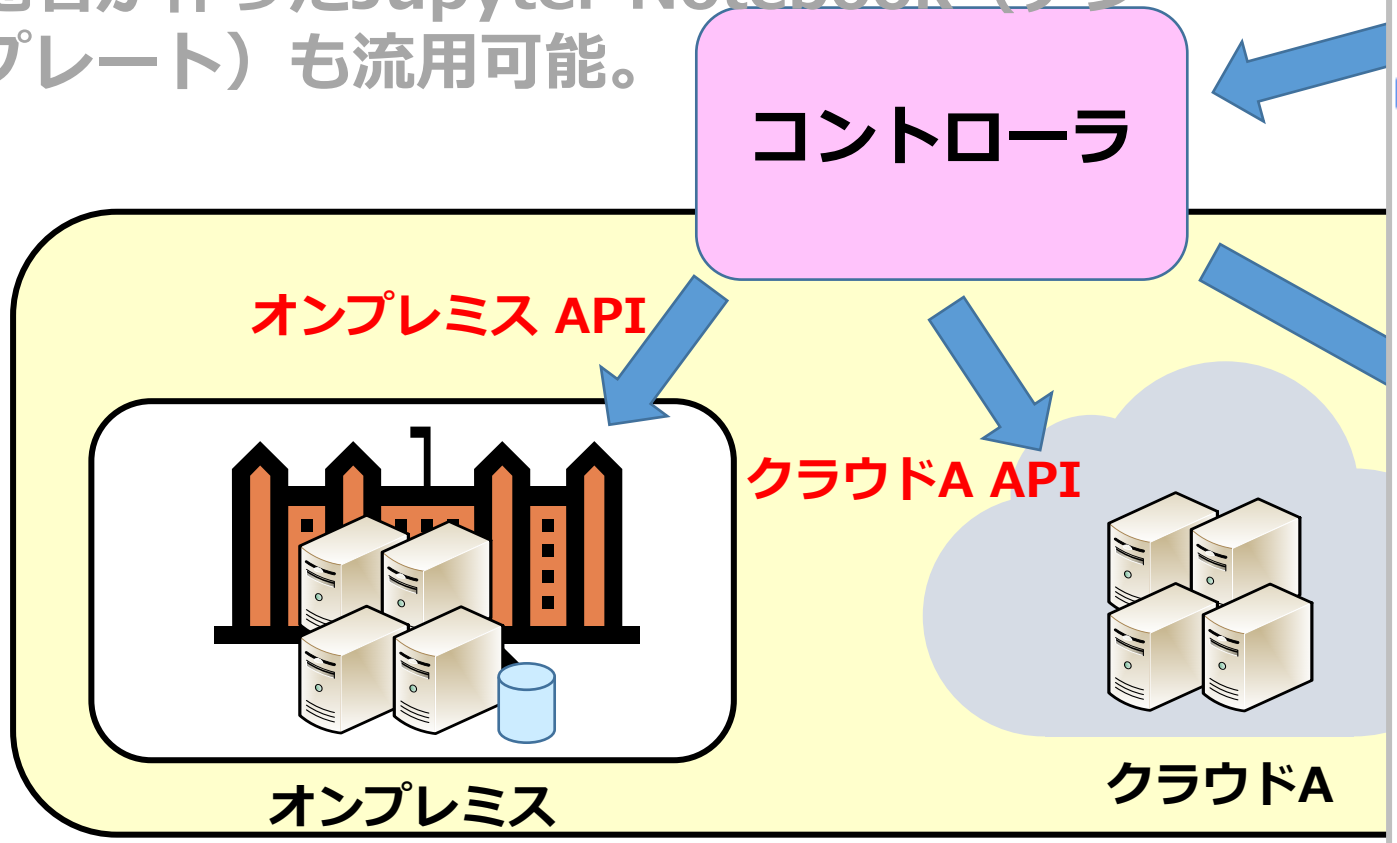
```
from common import logsetting
from vcpsdk.vcpsdk import VcpSDK

#
# 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！



Jupyter Notebookの記述例

1.1 初期化

[1]:
parameters

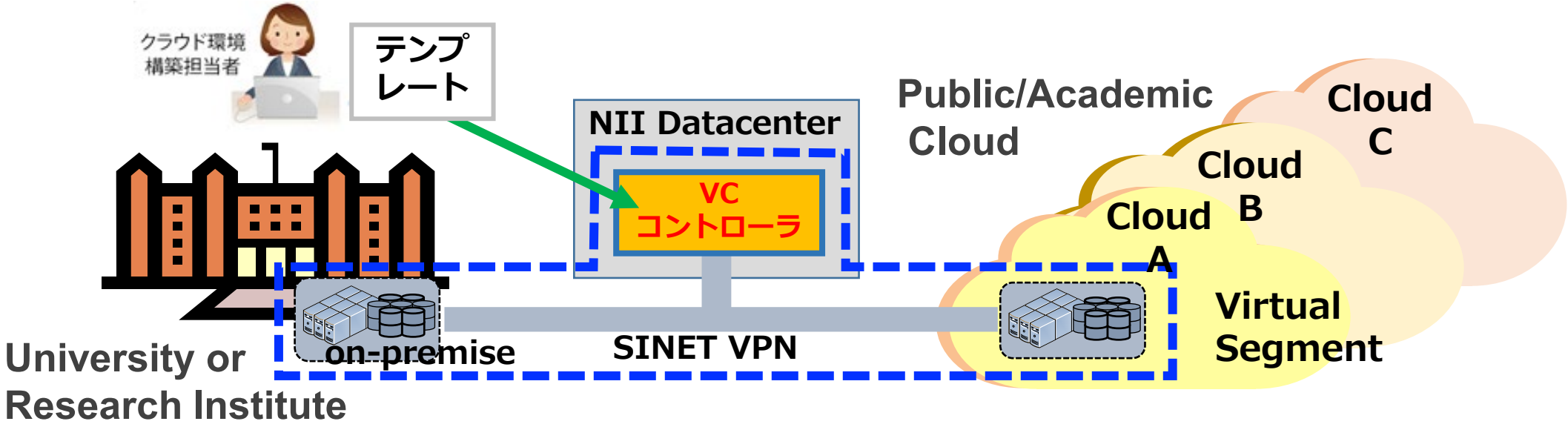
```
vcc_access_token = "
testname = "TEST-2022-03-15"
```

[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）



■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）

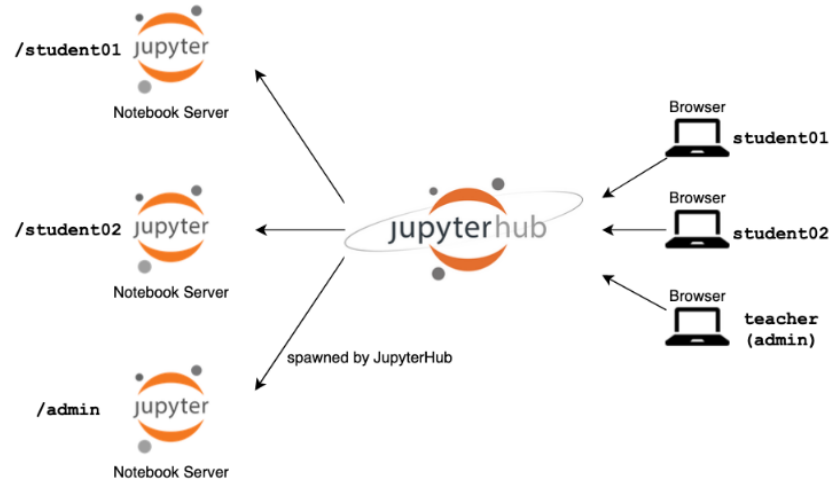
他者が作ったテンプレートの流用も可能

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JupyterNotebook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★

```
In [ ]:
##################################################
### 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名
ugroup_name = 'handson403'

# プロバイダ
vc_provider = 'aws'
```

図表を組み合わせた説明を挿入できる

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

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

# VCノード spec
spec = vcp.get_spec(vc_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、`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

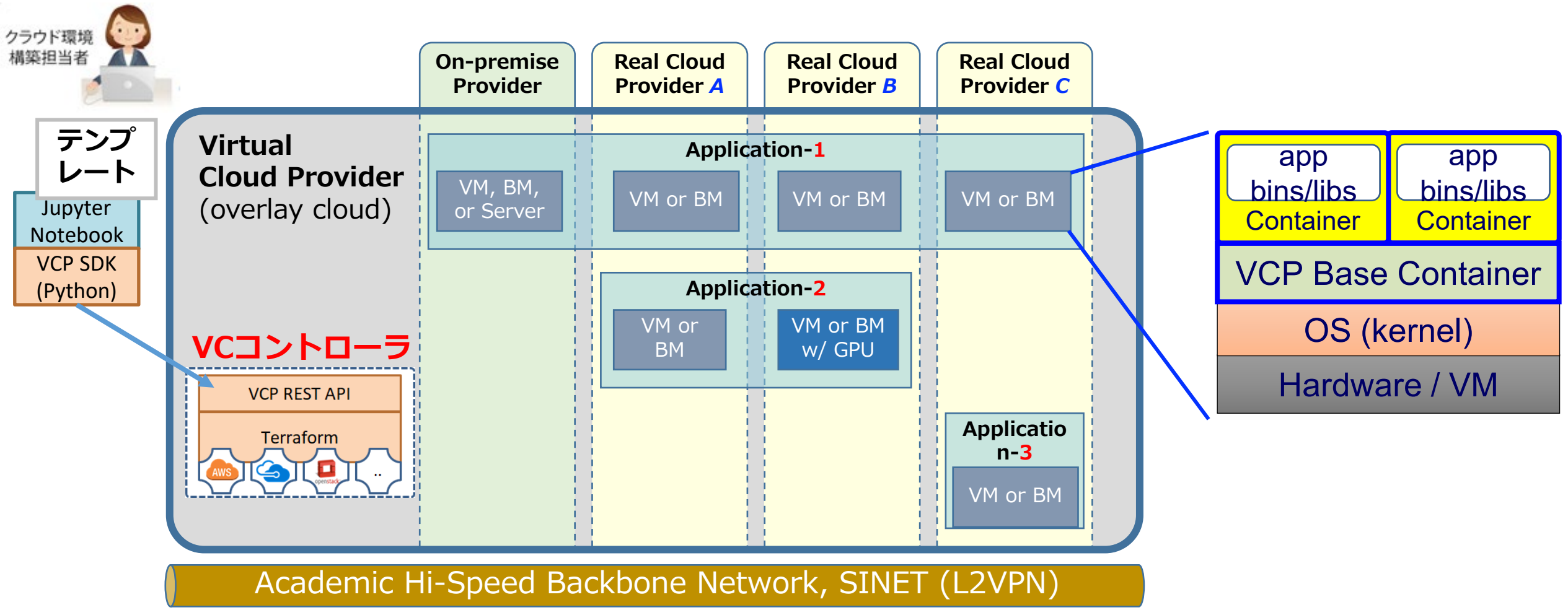
VCノード上に、本ハンズオン用に用意したThe Littlest JupyterHubのコンテナイメージを使用して環境を構築します。

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。

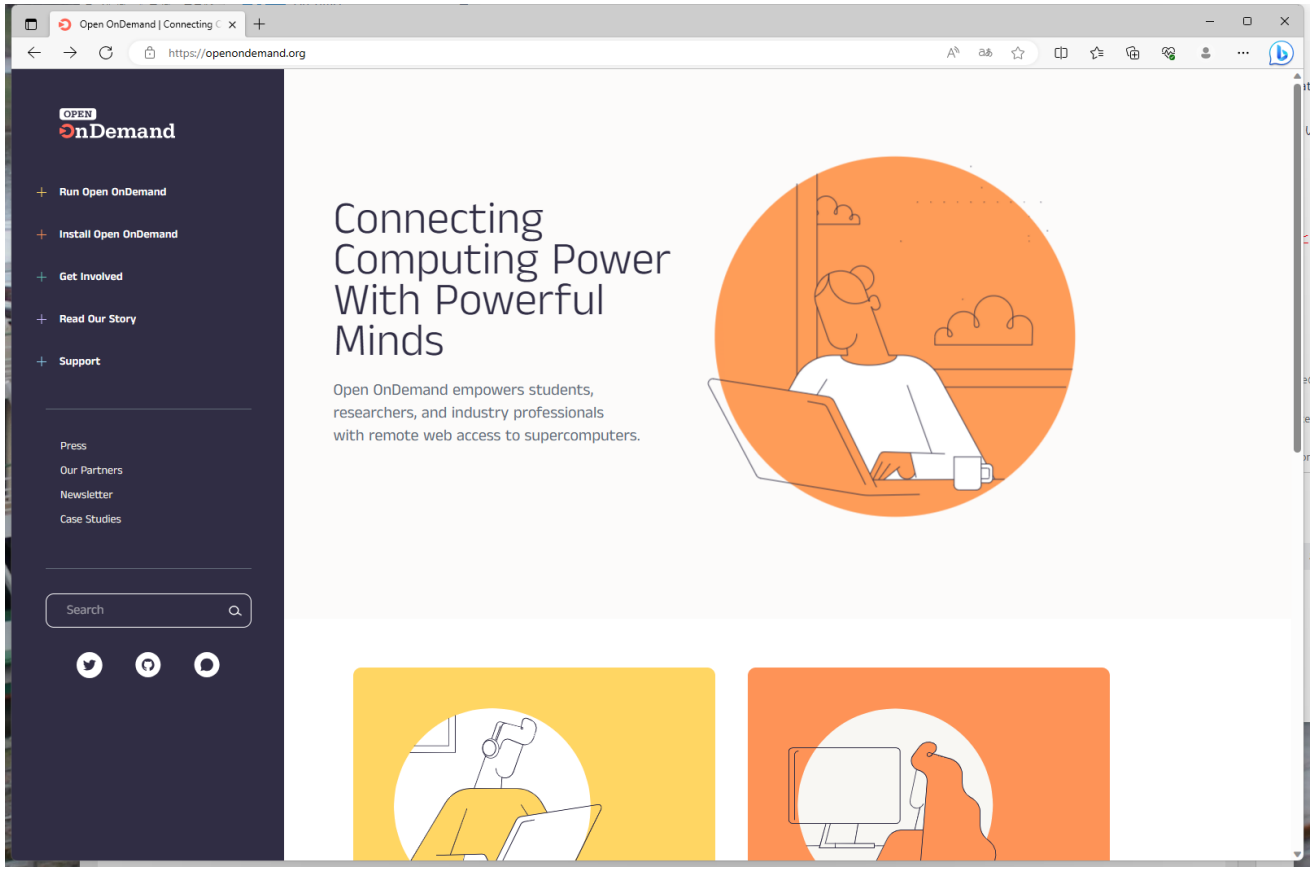
OCSを利用したアプリケーション配備例

■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！



Open OnDemandの概要

Open OnDemandとは

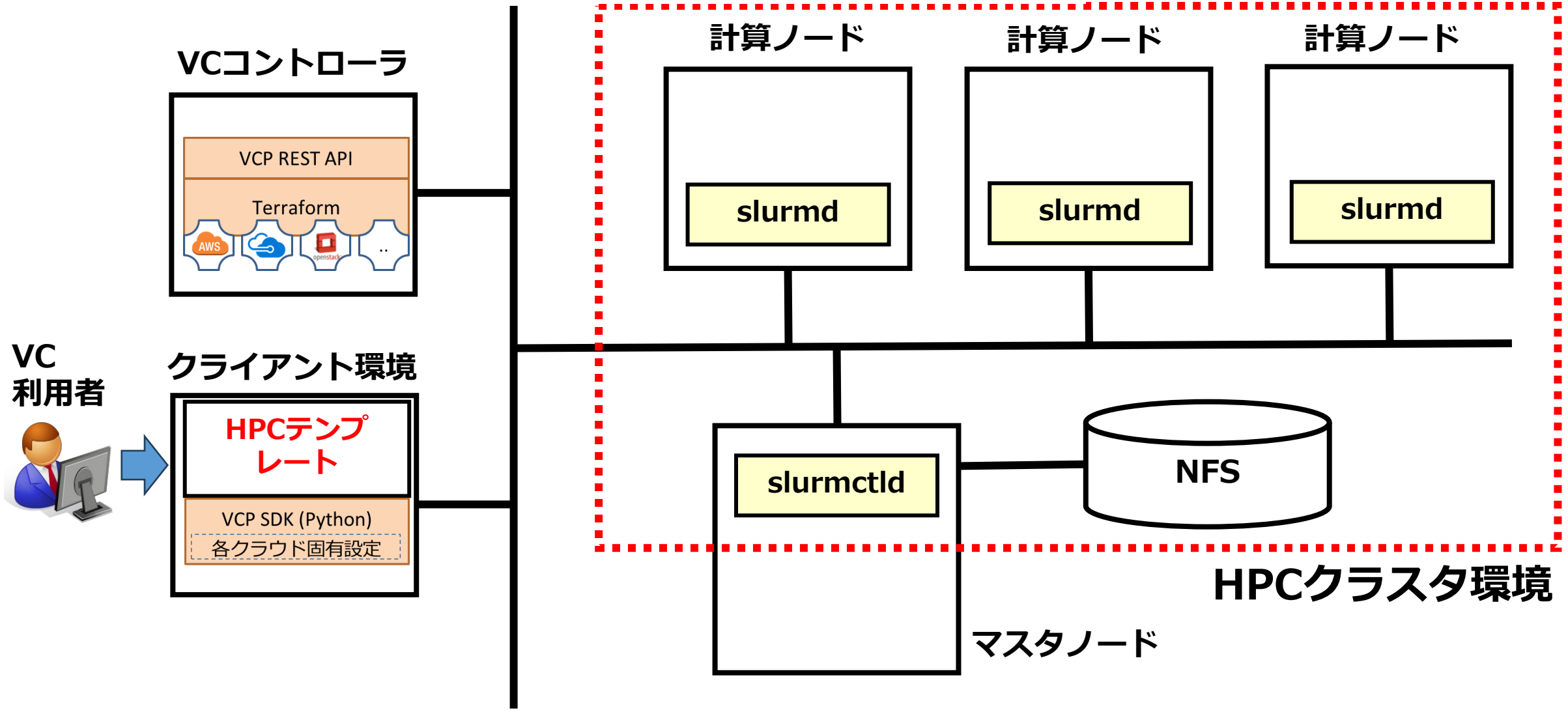


https://openondemand.org/

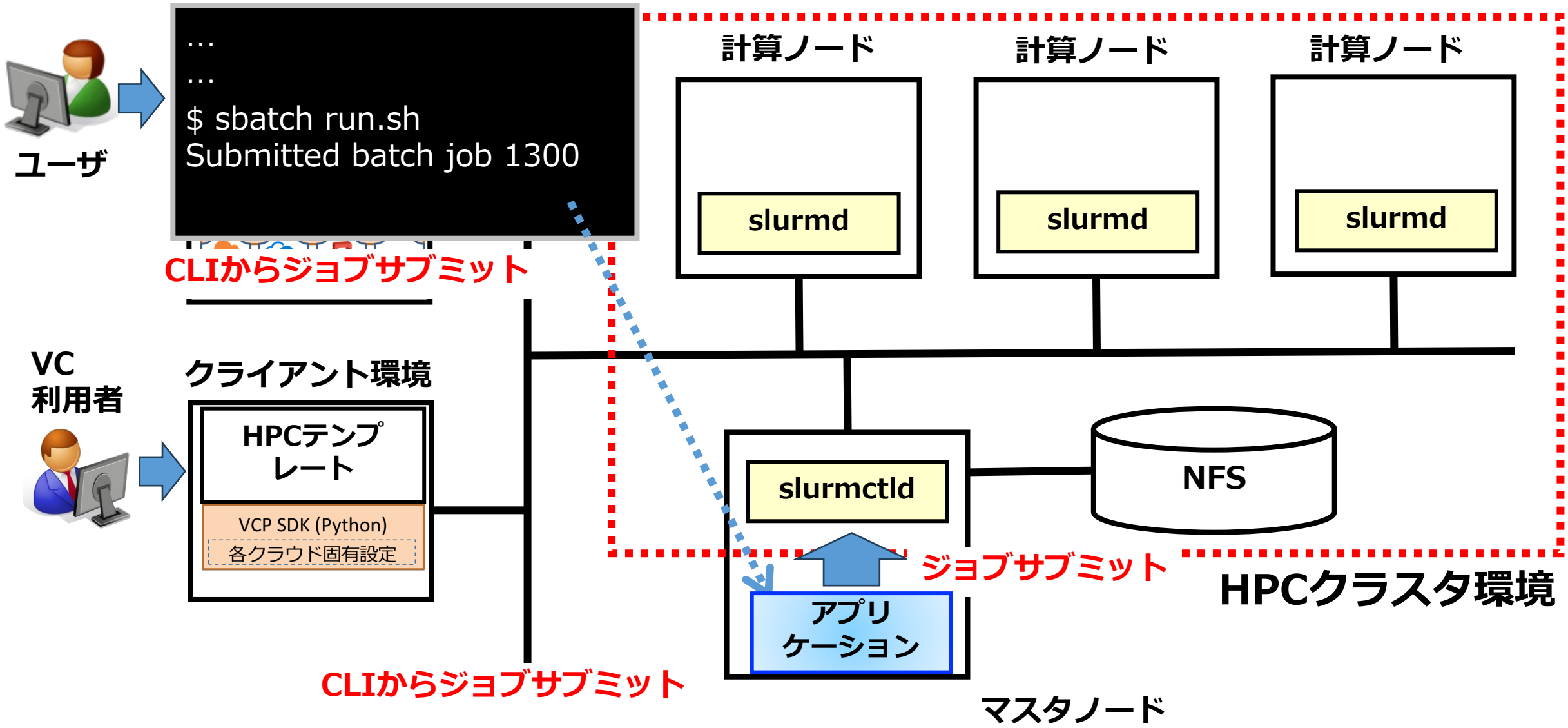
- HPCクラスタを容易に利用可能とするWebポータル
- これまでの経緯
  - 2007年より開発
  - 2013年から提供開始
  - 2022年時点で250以上のインストール
- コミュニティ
  - https://github.com/OSC/ondemand
  - https://discourse.openondemand.org/

OCSテンプレートによるOpen OnDemand環境の構築（１）

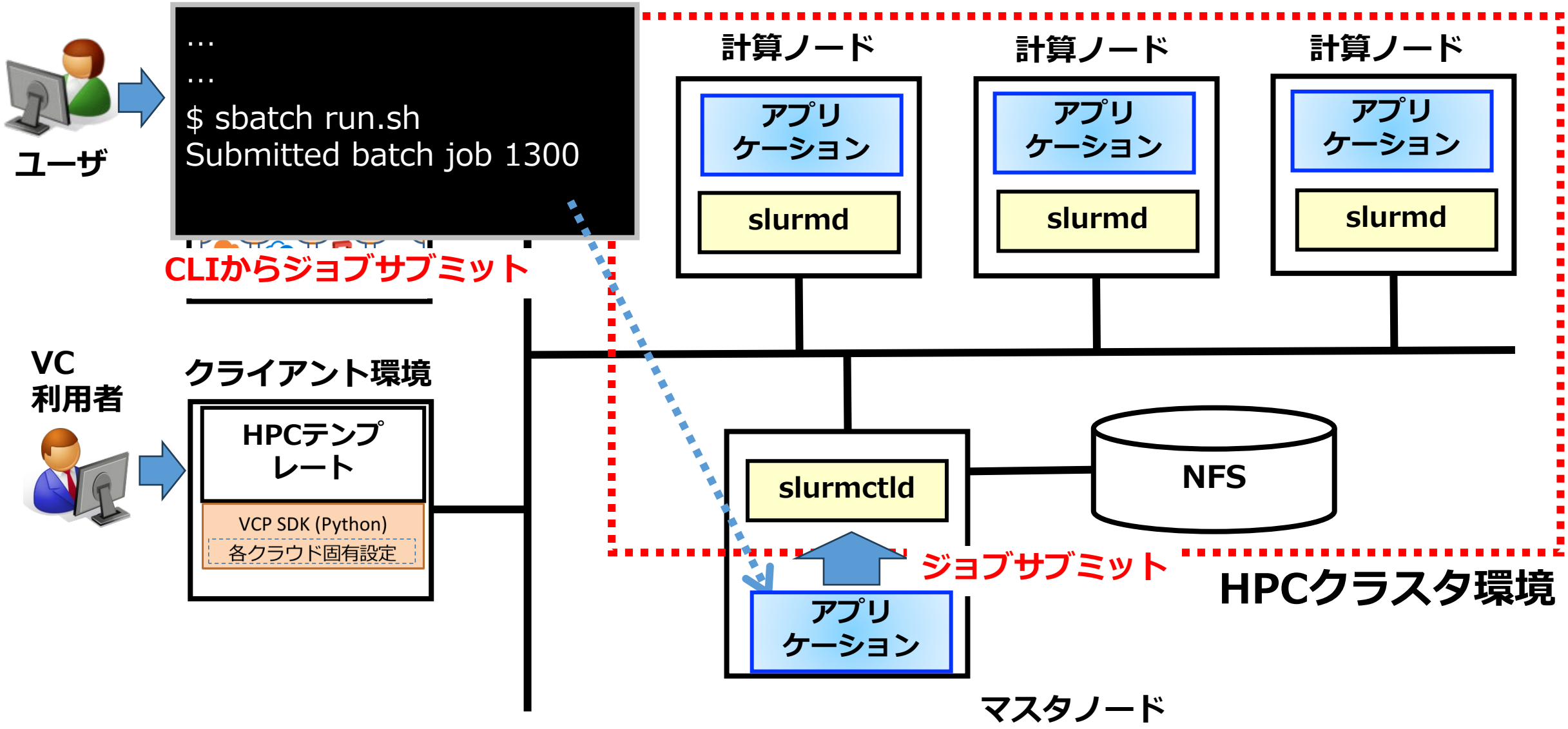
HPCテンプレートを用いて、HPCクラスタ環境を構築



OCSテンプレートによるOpen OnDemand環境の構築（２）

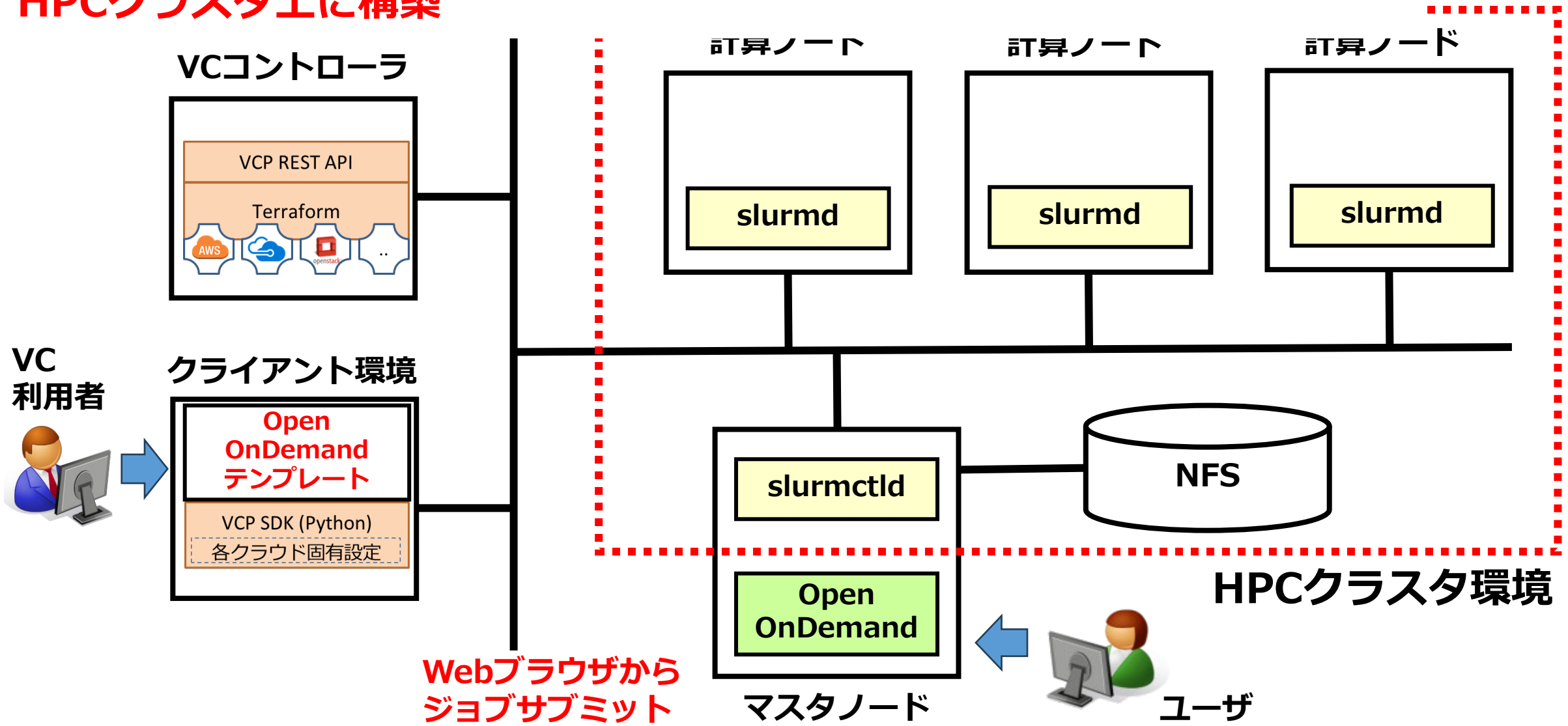


OCSテンプレートによるOpen OnDemand環境の構築（3）

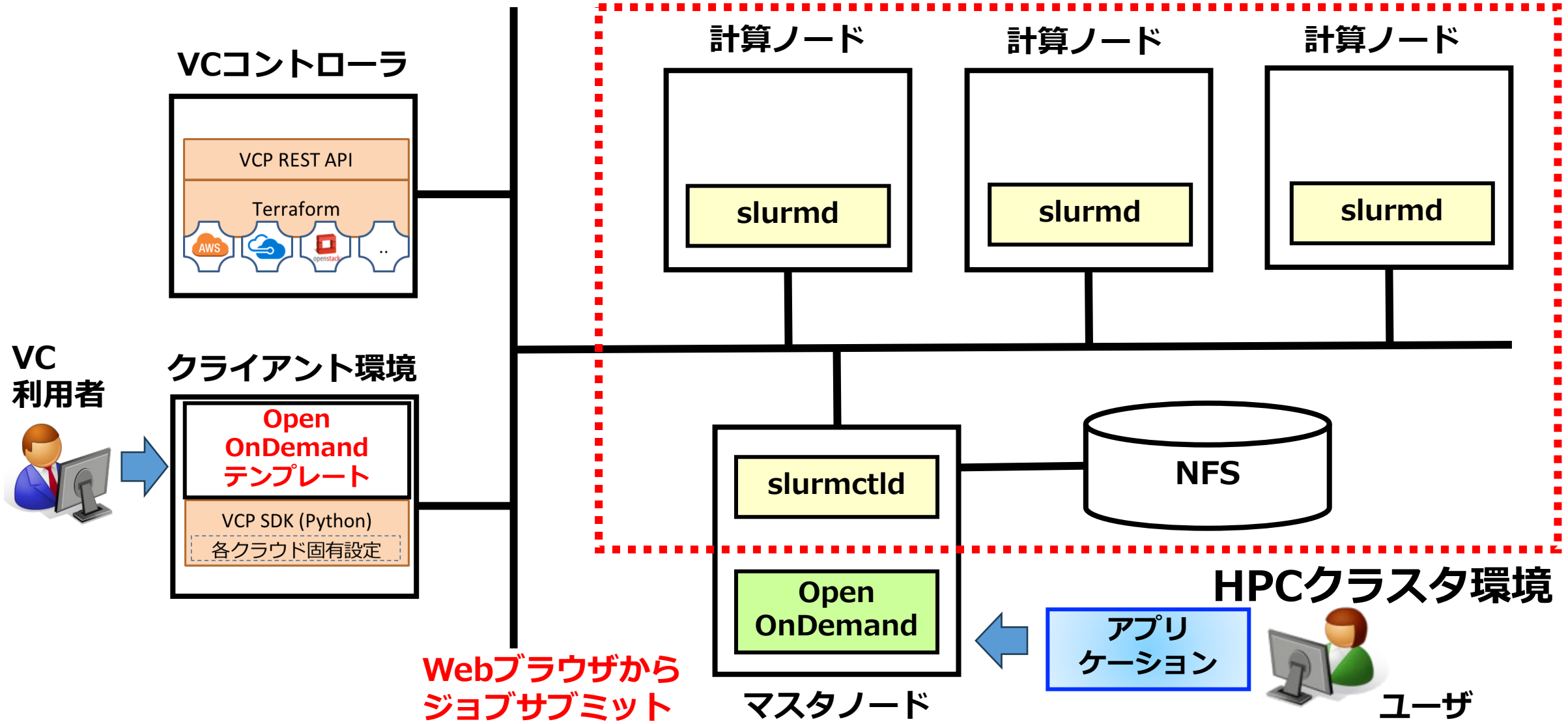


OCSテンプレートによるOpen OnDemand環境の構築（４）

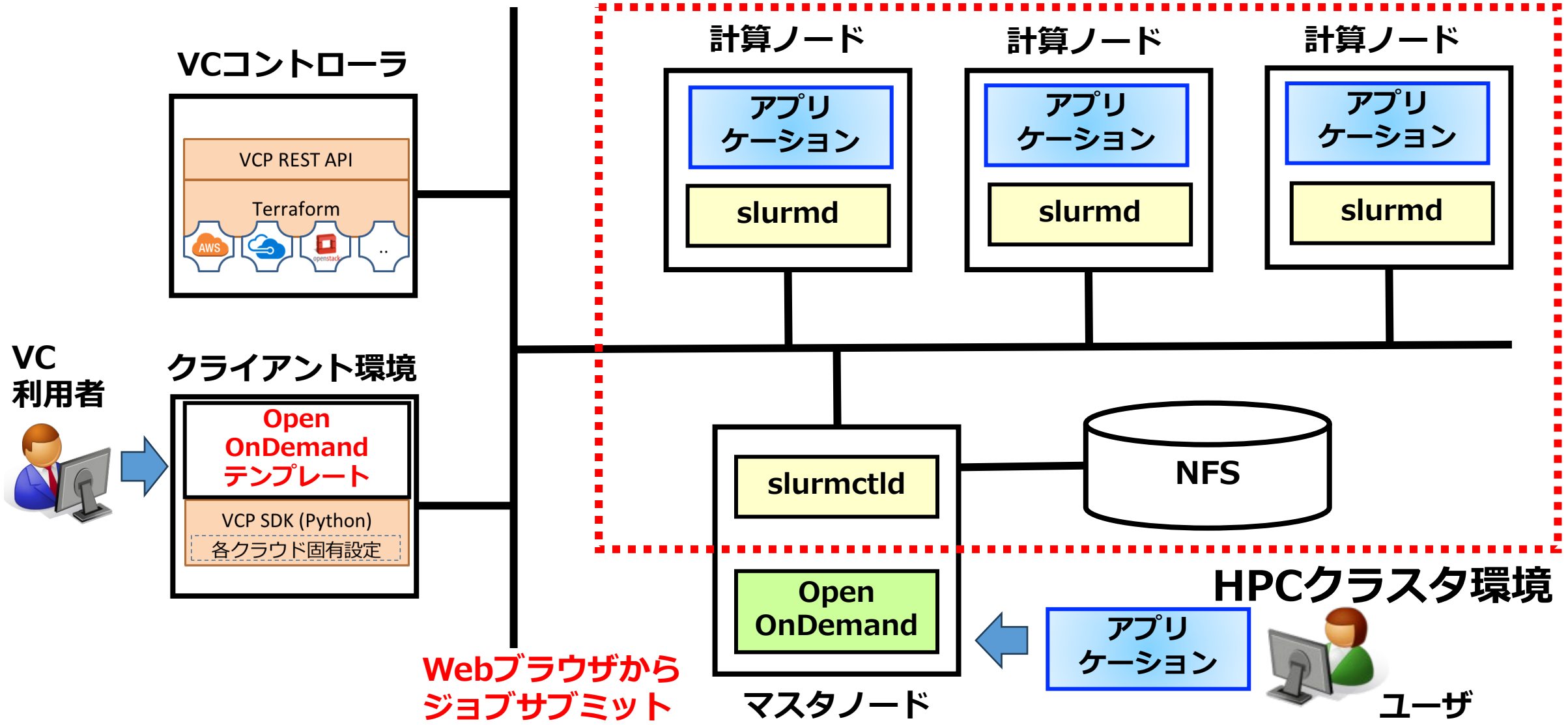
Open OnDemandテンプレートを用いて、Open OnDemand環境を HPCクラスタ上に構築



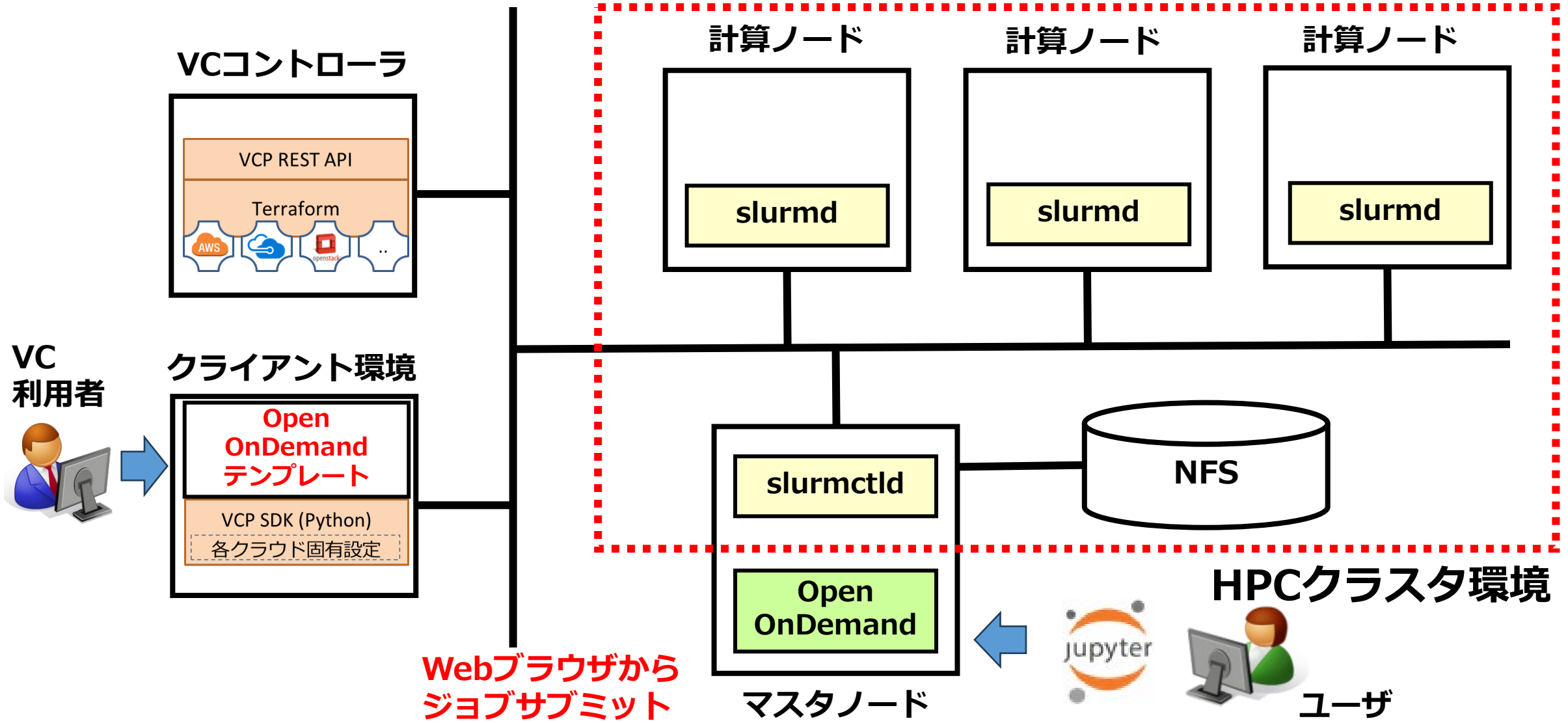
OCSテンプレートによるOpen OnDemand環境の構築（５）



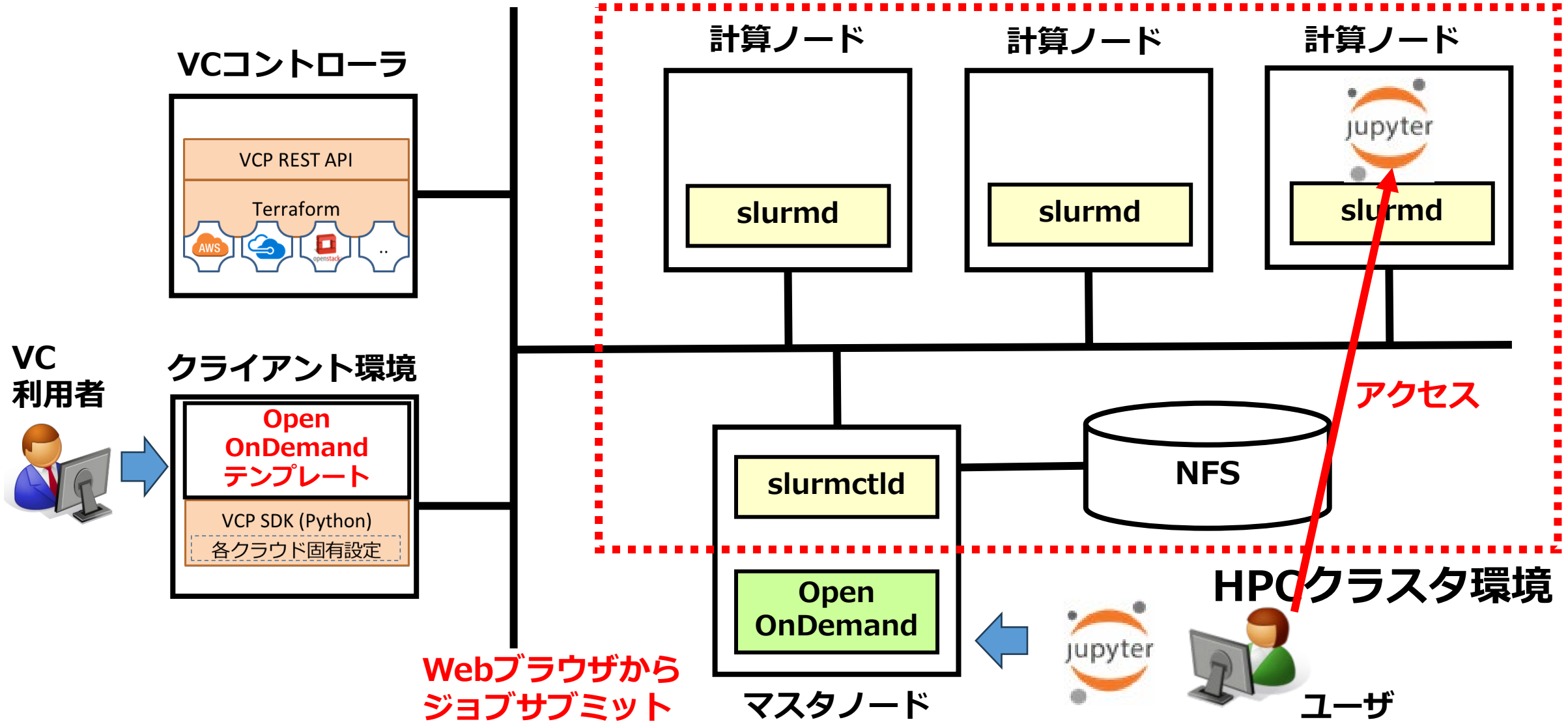
OCSテンプレートによるOpen OnDemand環境の構築（6）



OCSテンプレートによるOpen OnDemand環境の構築（7）



OCSテンプレートによるOpen OnDemand環境の構築（7）



GakuNin Cloud

Thank You.