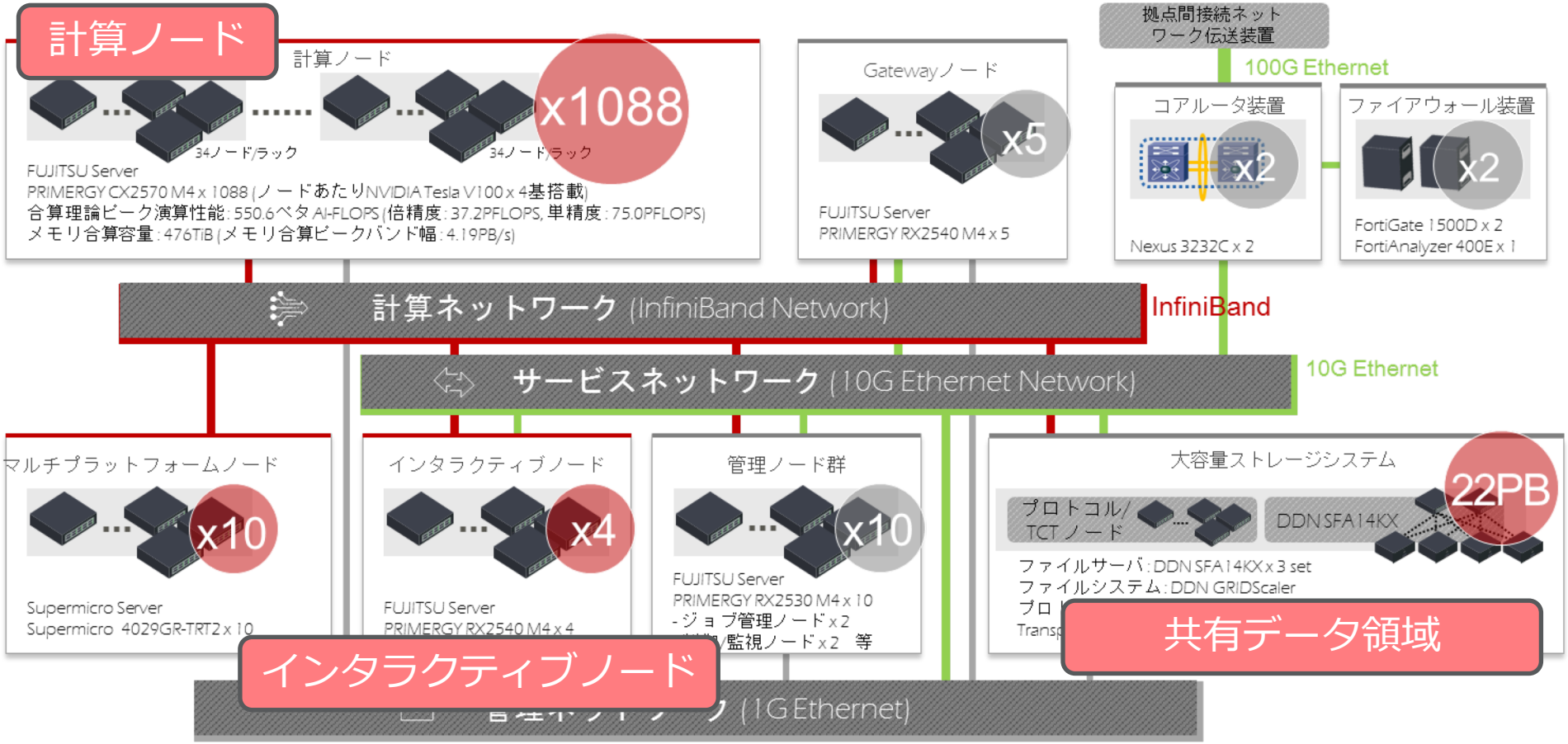
ABCI利用者講習会（初級編）

2019年2月7日
2019年3月25日（改定）

ABCIシステムの概要

- 1088台の計算ノードと22PBの大容量ストレージを高速ネットワークで接続した高性能計算システム
- 実際の利用は、インタラクティブノードと計算ノードを操作

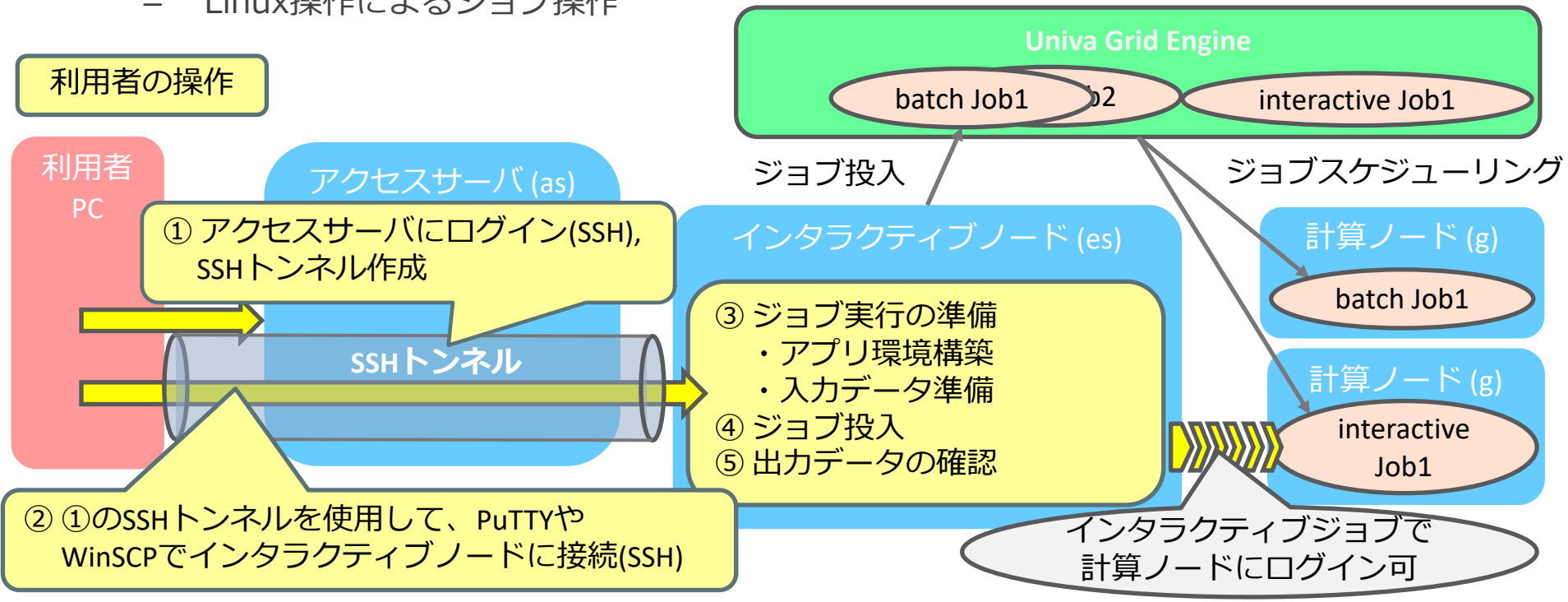


ABCIシステムの概要

- ABCIのノードの役割について
  - 計算ノード（ホスト名 g##### #は0001 ～ 1088までの4桁の数字）
    - 利用者のジョブ（プログラム）を実際に実行するサーバ群
    - ジョブスケジューラ（UGE:Univa Grid Engine）を介したジョブのみ実行可能
    - ジョブの実行方法は主に２つ
      - Spotサービス （バッチジョブ実行）
        UGEにジョブの実行を依頼し、UGEが確保した計算ノードで実行した結果を受け取り
      - On-demandサービス（インタラクティブジョブ実行）
        UGEが確保した計算ノードに利用者が直接ログインしプログラムを実行
  - インタラクティブノード （代表名：es, ホスト名 es1, es2, es3, es4 ）
    - インタラクティブノード(es)へログインすると、自動的にホスト名es1~ es4に振り分け
    - 利用者がログインしジョブ実行などの作業を行うABCIのサーバ
      - コンパイル、アプリのインストール、ジョブ実行スクリプトの用意
      - UGE にジョブ実行依頼
      - ABCI へのデータアップロード、ダウンロード
  - アクセスサーバ　（代表名：as）
    - インタラクティブノードにアクセスするため、最初にログインする操作不可の踏み台サーバ
    - コマンド操作をおこなうと、esなどへのSSHセッションが切れることに注意

利用のイメージ

- ABCIへのログイン
  - SSH接続とSSHトンネルを使用したアクセス
- インタラクティブノードでのジョブ実行
  - Linux操作によるジョブ操作

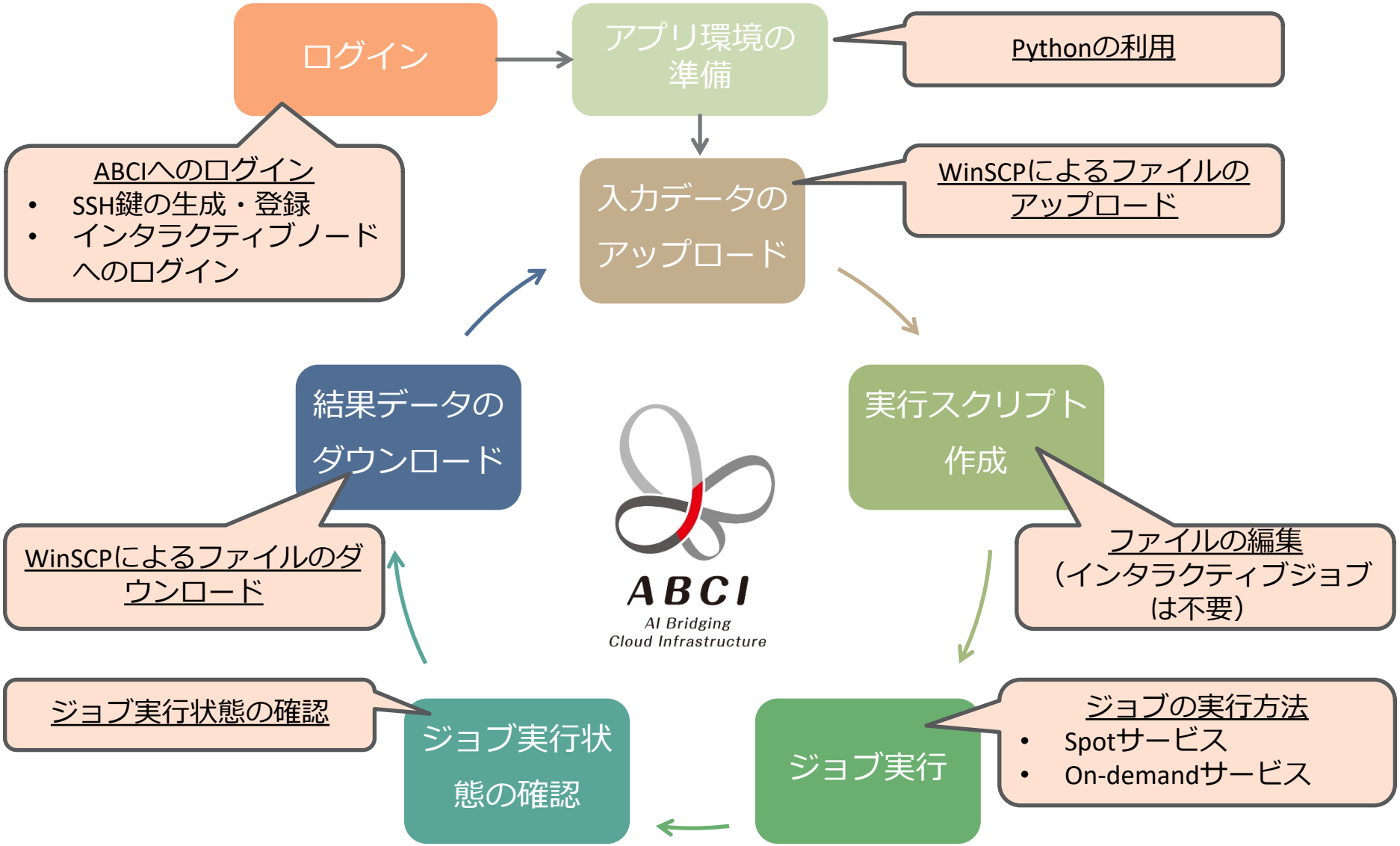


※インタラクティブノード利用時の注意事項

1. インタラクティブノード上では長時間資源（CPU、メモリ等）を占有する計算の実行は許可されておりませんが、アプリケーションのコンパイル・インストールなどの比較的低負荷の処理は可能です。
2. GPUを使用するアプリ環境の準備は、インタラクティブジョブとして計算ノードにログイン後、環境構築します。

利用フローと本日の講習項目

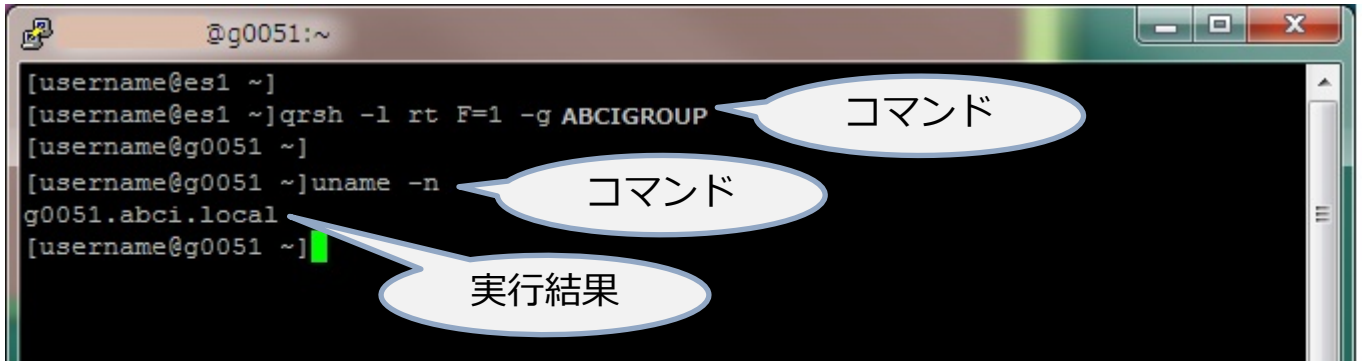
講習の項目



ご参考：ABCIでのオペレーション

- ABCIはLinux OSで構成されたシステム
  - コマンド実行による対話的な操作が主体
  - Windowsユーザはターミナルソフトが必要

    ex) PuTTY (https://www.putty.org/)



  - Mac, LinuxユーザはOS添付のターミナルが利用可能

定型の操作を覚えれば利用可能

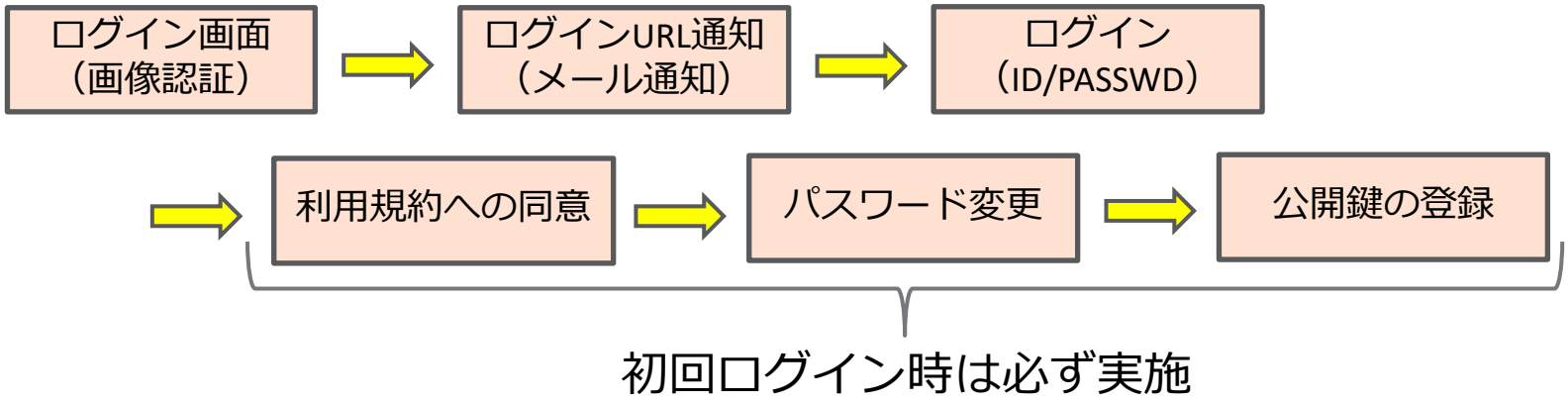
SSH鍵の生成・登録について

- SSH (Secure Shell)
  - ネットワークの通信経路を暗号化して安全に通信するプロトコル
  - 公開鍵認証を始めとする様々な認証方法をサポート
- 公開鍵認証をABCIでは採用
  - 公開／秘密鍵ファイルをあらかじめ作成し、公開鍵をABCIに登録
  - 認証は手元の秘密鍵とリモートの公開鍵をつき合わせて実施

SSH鍵の生成・登録のフロー

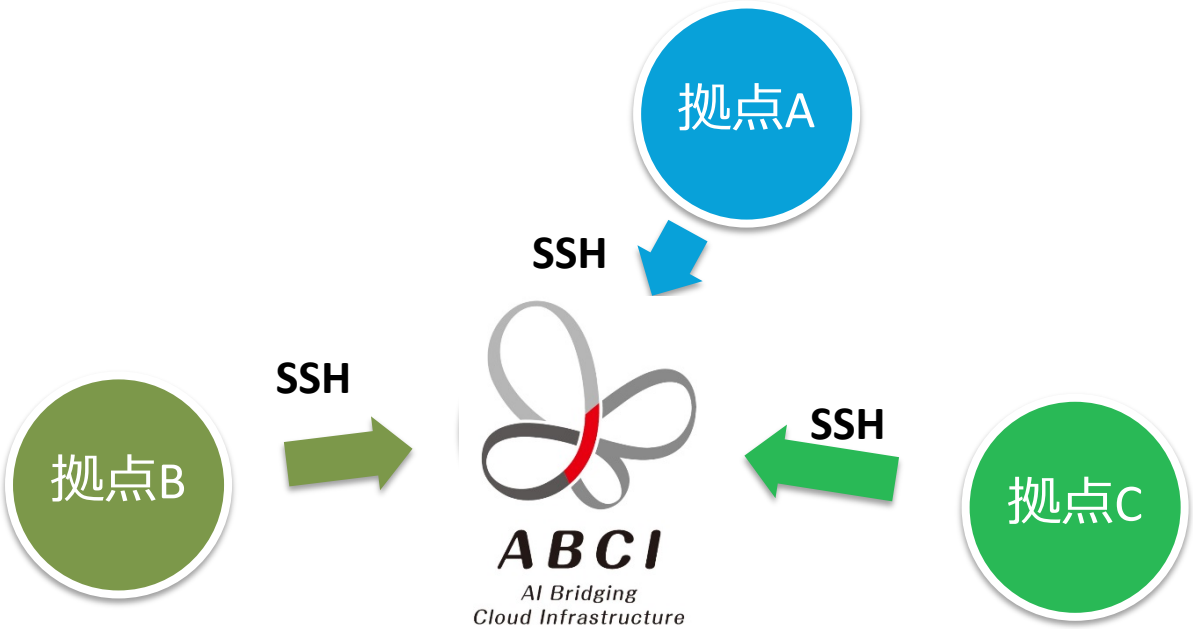
- SSH鍵の生成
  - SSH公開鍵／秘密鍵は、利用者の端末で作成
- ABCIへのSSH公開鍵の登録
  - SSH公開鍵の登録は利用者ポータルにて登録

    https://portal.abci.ai/user/?lang=ja
  - 利用者ポータル操作の流れ



ご参考：複数拠点からのアクセス

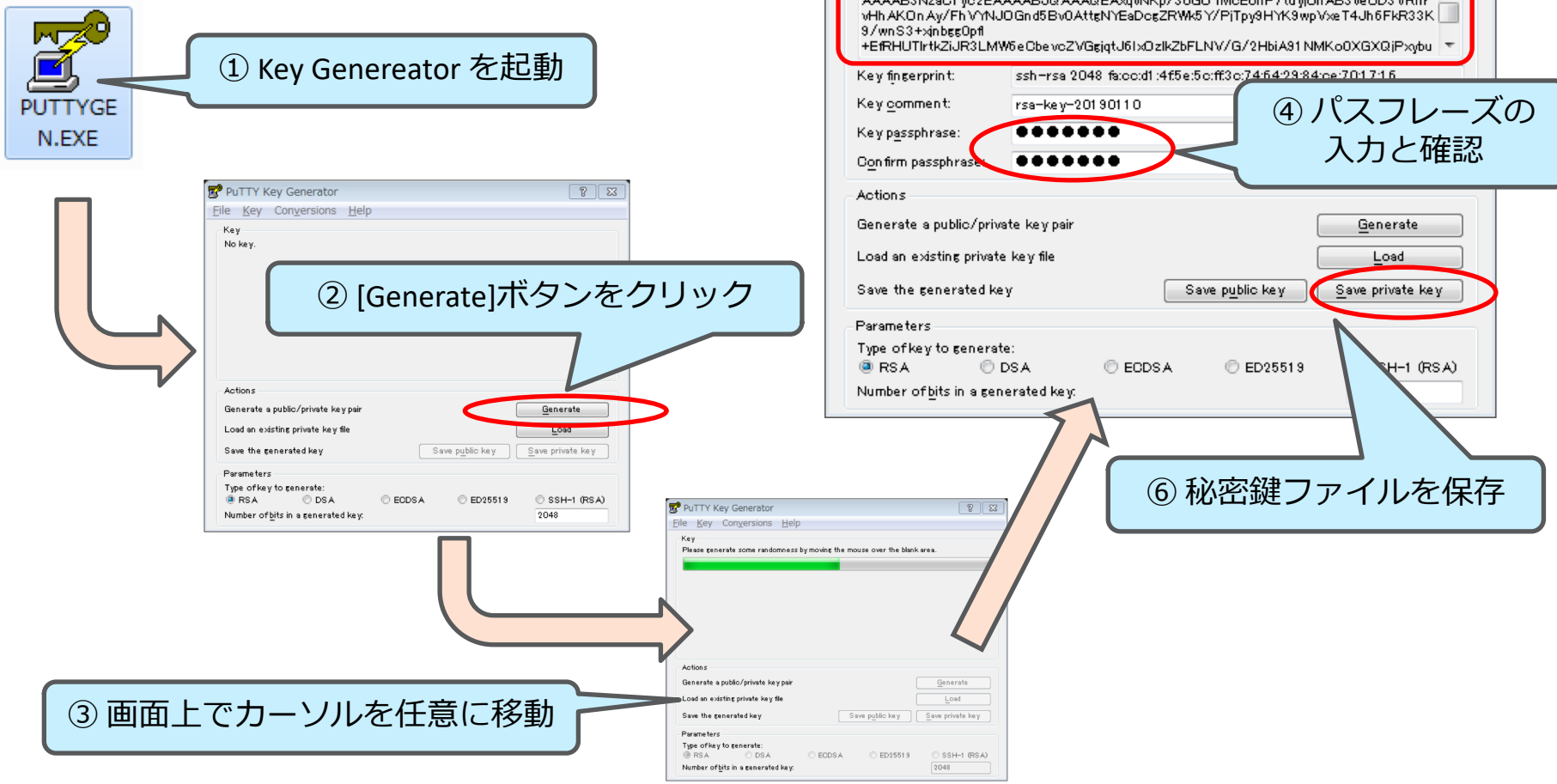
- ABCIはインターネットに公開されたサービス
  - インターネットに接続できれば、どこからでも利用可能
  - 暗号化した通信経路（SSH）を使用しセキュアに通信
  - 端末が異なる場合は端末ごとに公開鍵の登録が必要
    - 公開鍵の追加は利用者ポータルで登録



インターネットに接続可能かつ鍵登録済みPCあれば様々な拠点から利用可能

SSH鍵の生成・登録（Windows）

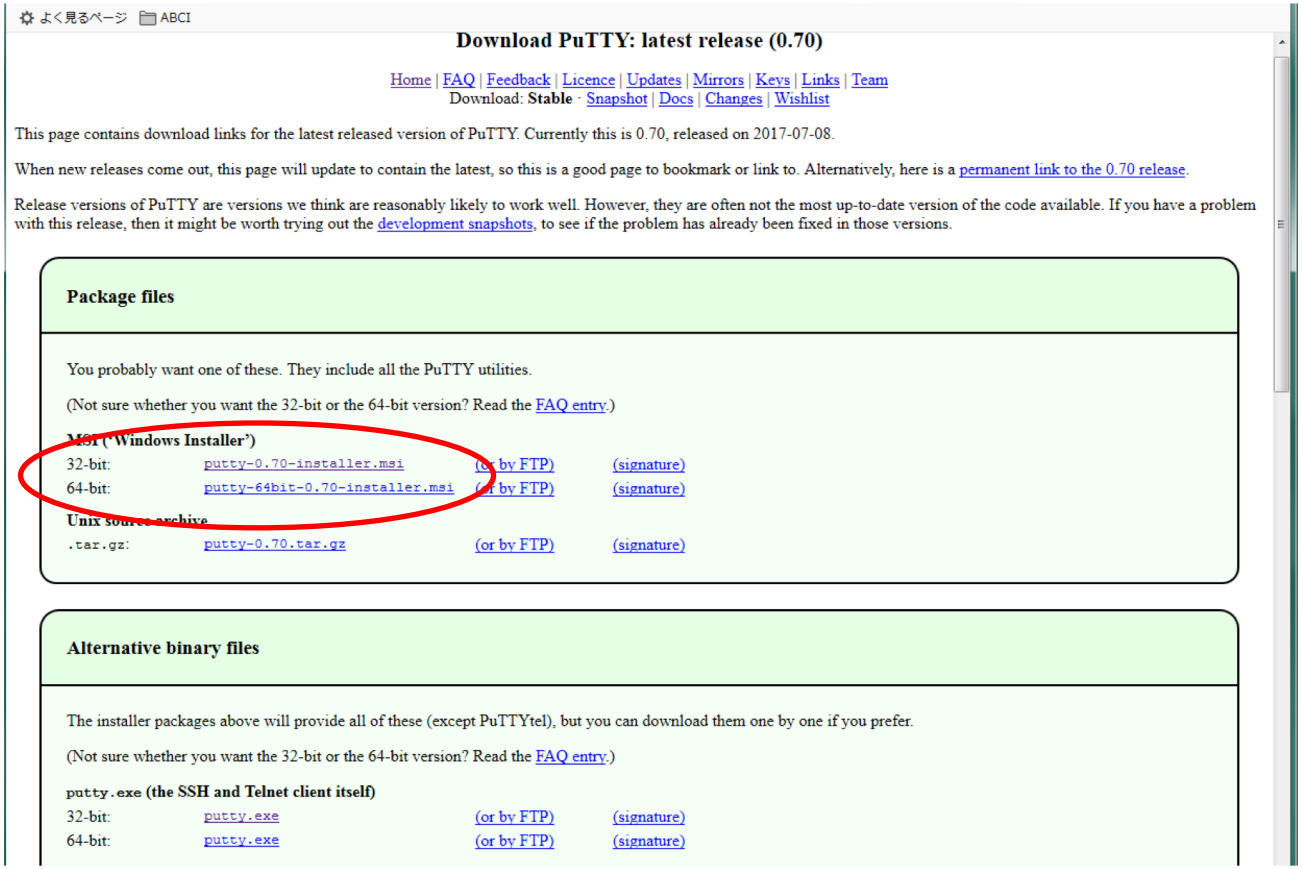
- SSH鍵の生成（PuTTY）



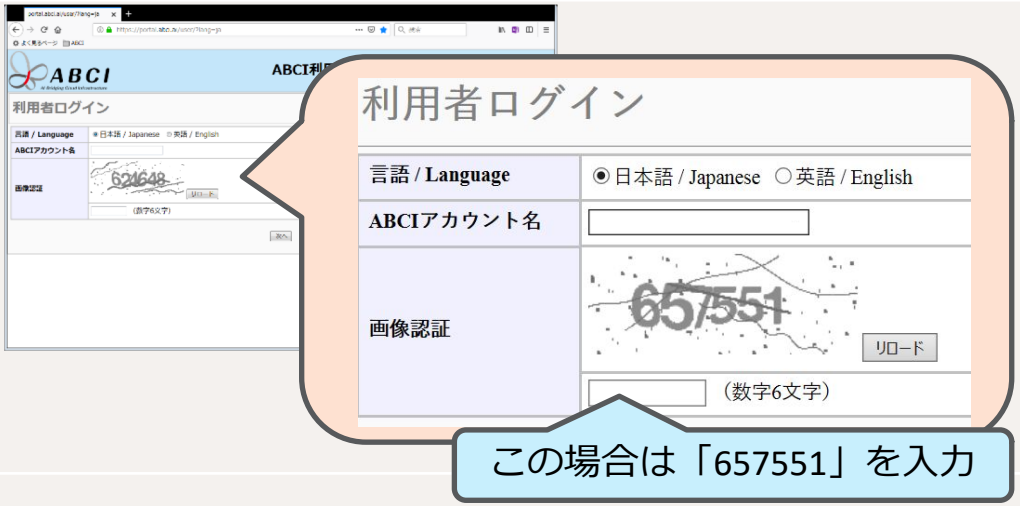
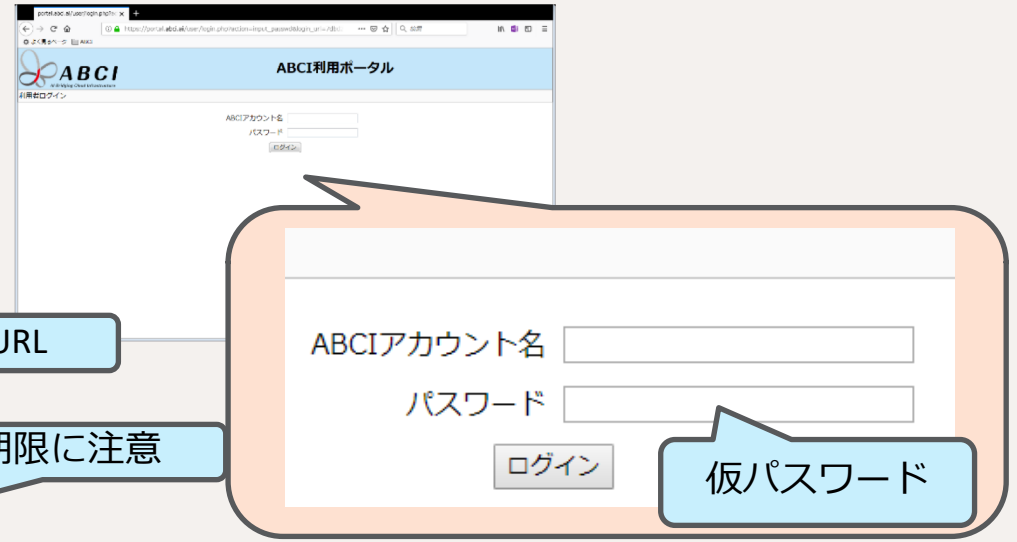
ご参考：PuTTYの入手先

- URL

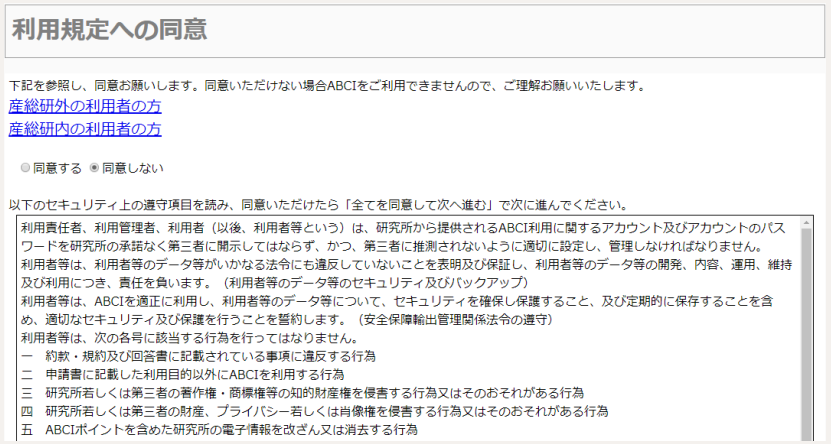
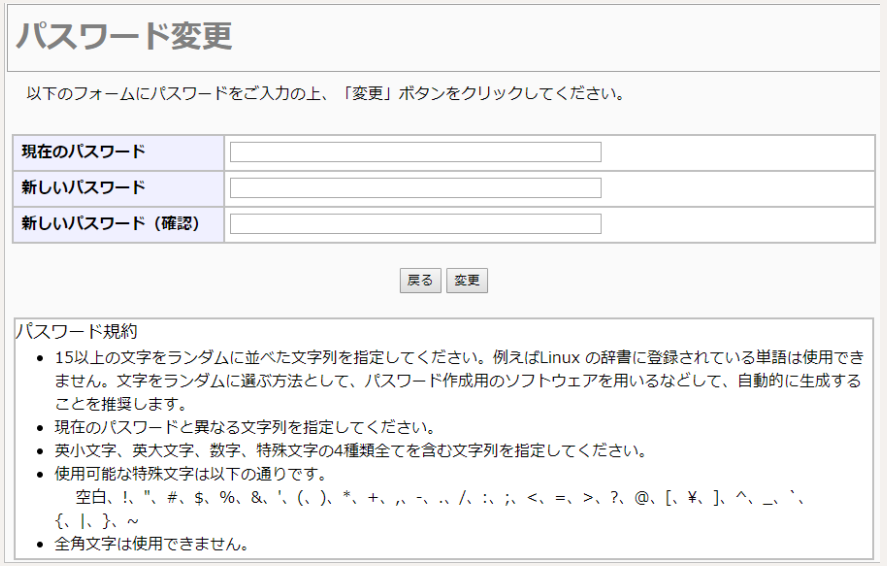
  https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html/



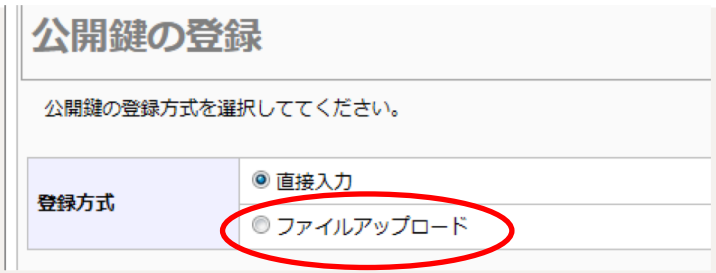
公開鍵の登録 (1/3)

操作説明	画面
利用者ポータル（画像認証）にアクセス • ABCIアカウント名を入力 • 画像認証の数値をテキストボックに入力 利用者ポータル https://portal.abci.ai/user/	
メール通知されたURLにアクセスし利用者ポータルにログイン • ABCIアカウント名を入力 • 仮パスワードを入力 通知メール文例 xxxxx 様 このメールはABCI利用ポータルよって送信されました。 以下のURLからログインしてください。 https://portal.abci.ai/user/login.php?action=input_passwd&login_url=xxxxxxxxxxxxxxxxxxxx ※このURLの有効期限は "2019/01/10 12:02" までとなりますのでご注意ください。	

公開鍵の登録 (2/3)

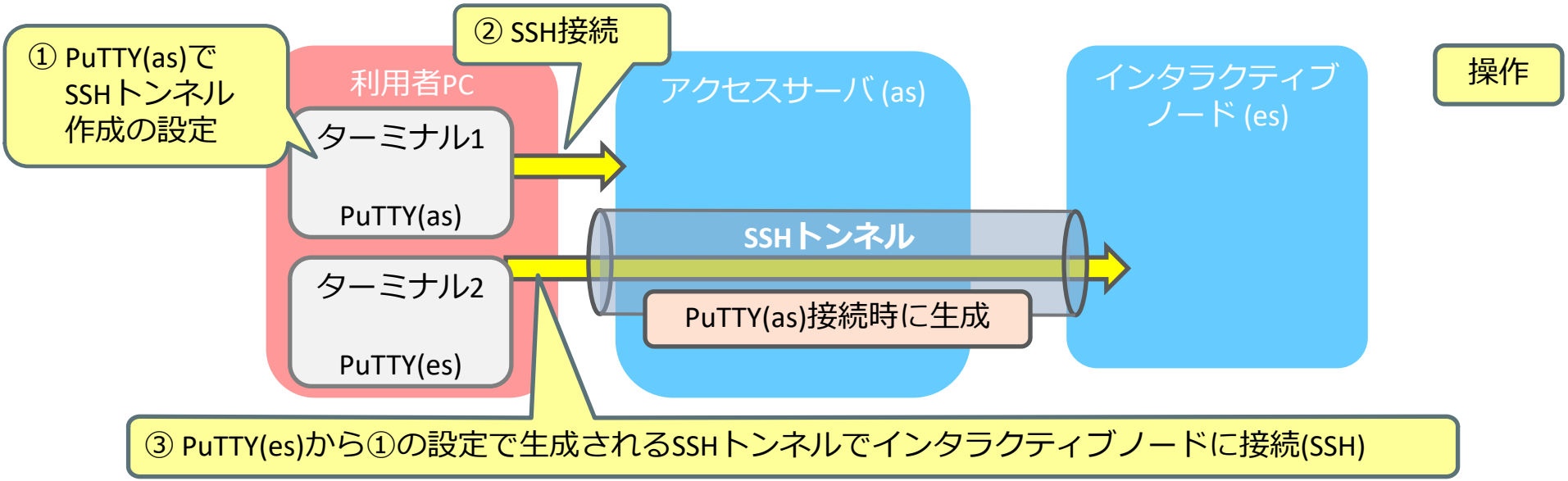
操作説明	画面
利用規定への同意（初回ログイン時のみ） • 利用規定を読み、同意する場合は「全てを同意して次へ進む」をクリック	
パスワード変更（初回ログイン時のみ） • 仮パスワードと新しいパスワードを入力し、「変更」をクリックすると確認ダイアログが表示されますので「OK」をクリックします。 ※ パスワードは初回設定後も利用者ポータルで変更可能です。	

公開鍵の登録（3/3）

操作説明	画面
公開鍵の登録① • 登録方式に[ファイルのアップロード]を選択	
公開鍵の登録② • [参照]をクリックしファイルのアップロード画面を表示 • 事前に作成した公開鍵ファイルを選択 • [登録]をクリックし公開鍵を登録 ※ 公開鍵は初回設定後も利用者ポータルで追加、削除が可能です。	

インタラクティブノードへのログイン（Windows）

- ターミナルアプリは PuTTY を使用
- インタラクティブノードへの接続の流れ
  - SSHポートフォワーディングによるトンネル設定
  - 最初にターミナルからアクセスサーバにログイン
  - 次に別のターミナルでインタラクティブノードにログイン



SSHトンネル設定（PuTTY）

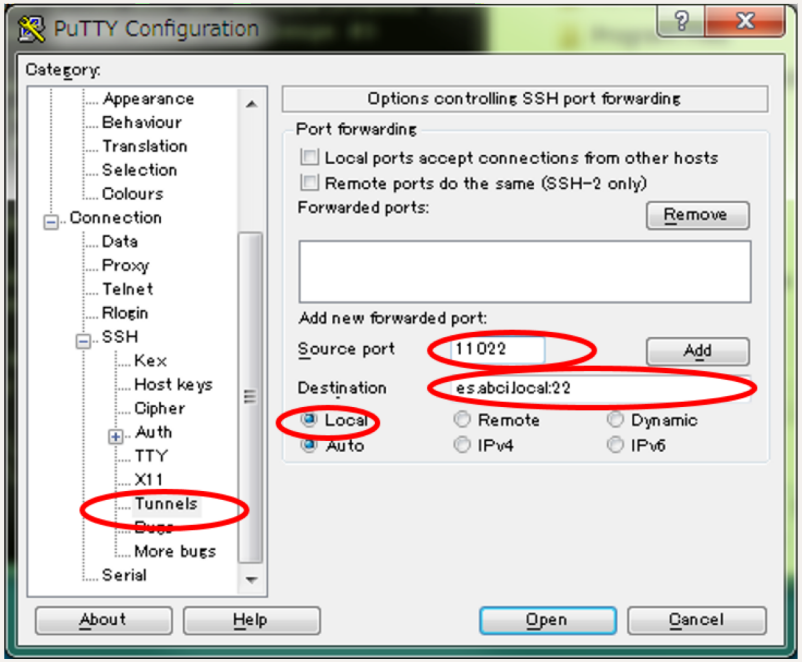
操作説明	画面

SSHポート転送の設定

- PUTTY.exe をダブルクリックし起動
- [Category]から、 [Connection] > [SSH] > [Tunnels] を選択しメニューを開く
- 以下表の設定を入力する

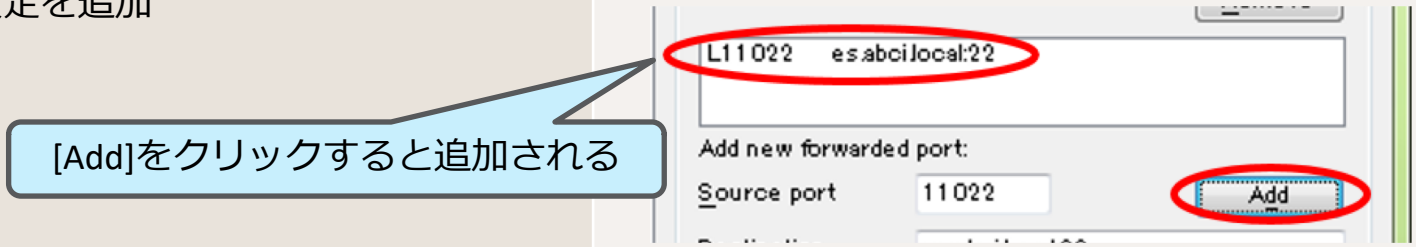
設定項目	設定値
Source port（変更可）	システムで許容されるポート番号 ex) 11022
Destination（固定値）	・アクセスサーバからの接続先とポート番号 es.abci.local:22 または es:22 ・Tunneling の方式 Local

※ Source port の設定値は後のインタラクティブノードへの接続に使用するため記録を推奨

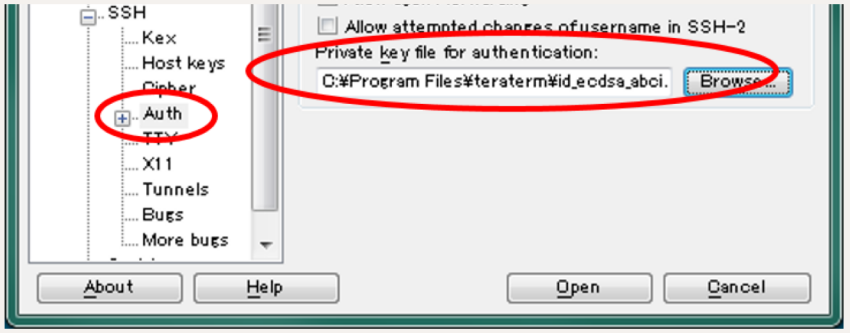
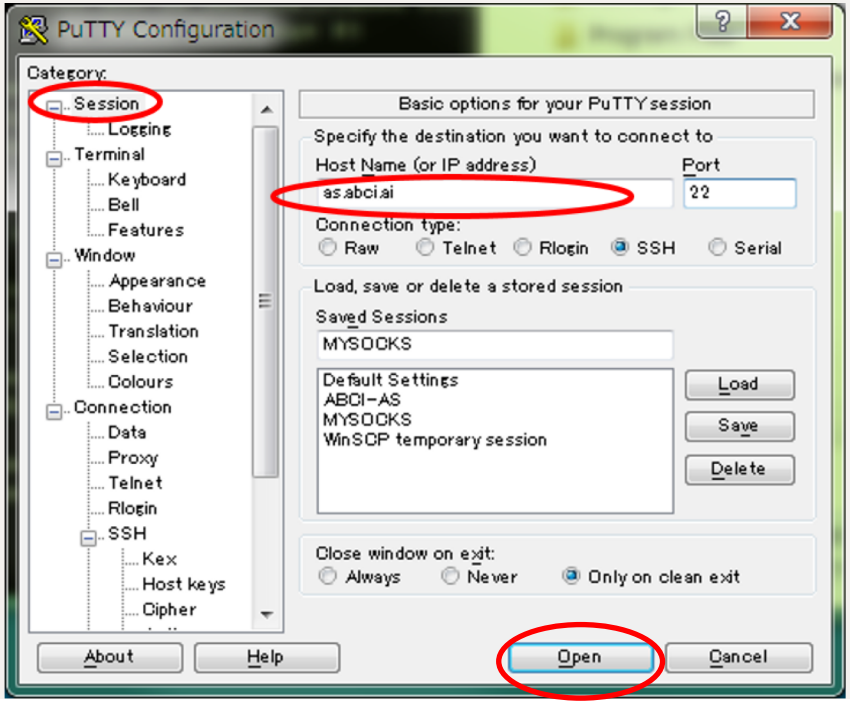


設定の登録

[Add] をクリックして設定を追加

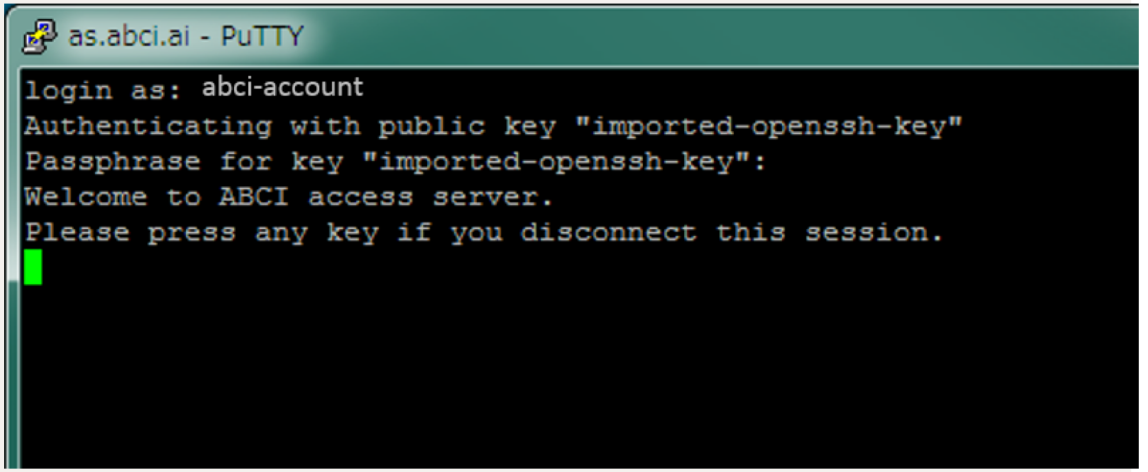


アクセスサーバへの接続 (1/2)

操作説明	画面
SSH秘密鍵の指定 • [Category]から、 [Connection] > [SSH] > [Auth] > を選択しメニューを開く • [Browse]ボタンをクリックし、秘密鍵ファイルを選択する。	
アクセスサーバの接続設定 • [Category]から、 [Session]を選択してアクセスサーバの接続情報を入力 • [Open]をクリックしアクセスサーバに接続開始	

設定項目	設定値
Host Name	as.abci.ai
Port	22

アクセスサーバへの接続 (2/2)

操作説明	画面
アクセスサーバとのSSH認証 • ABCIアカウント名を入力 • SSH鍵を作成時に設定したパスフレーズを入力	
ログインに成功 ※ 画面上でキー操作を行うとSSH接続が切断されるため、注意すること ※ ログインに成功後は接続設定の保存を推奨。（次項参照）	

ご参考：PuTTYの設定保存

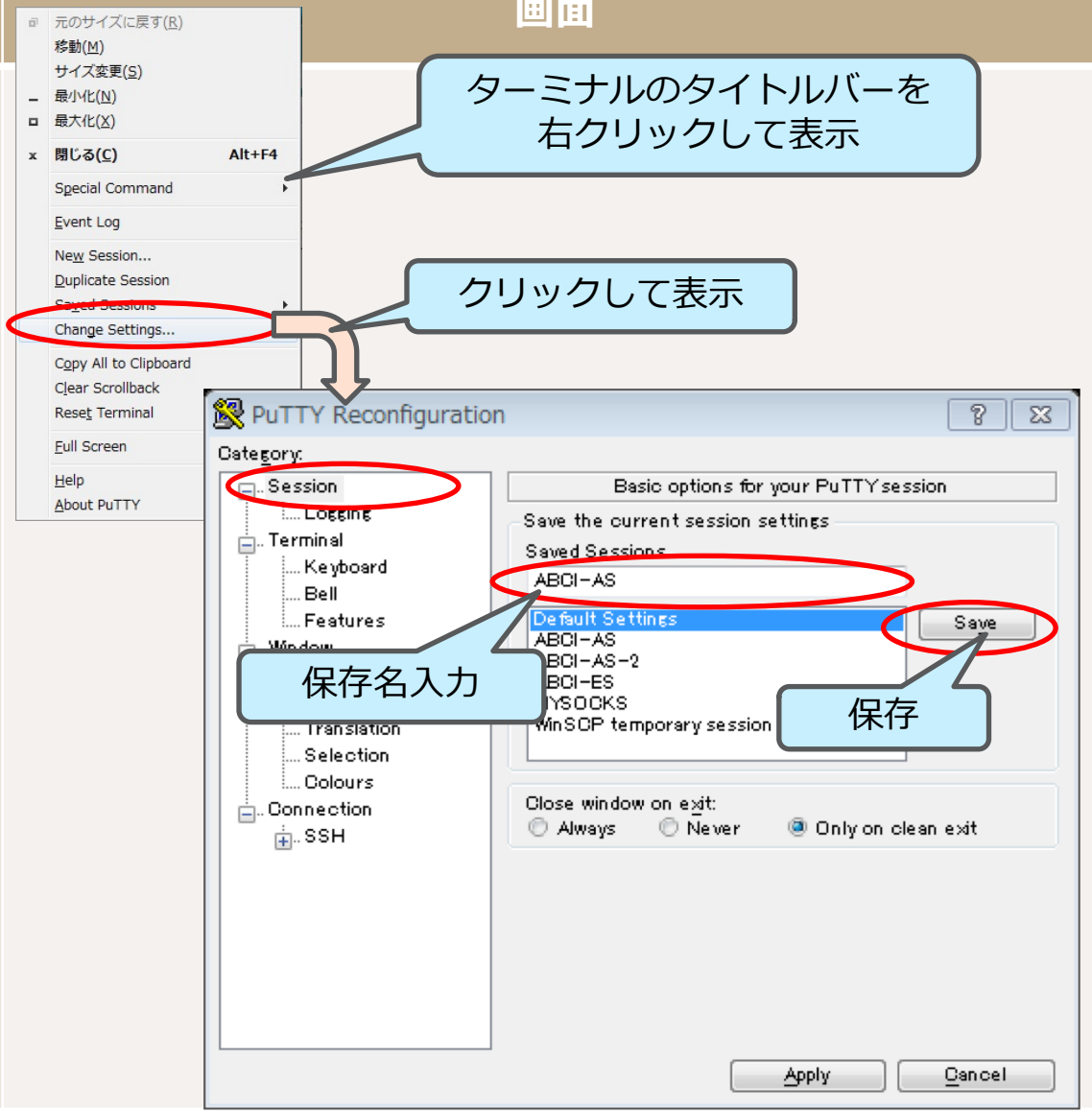
操作説明

接続設定の保存
- 接続済みターミナルのタイトルバーを右クリック

- Save Sessions に保存名を入力
- [Save]ボタンを押してで設定保存。

接続設定の呼び出し
- 保存名を選択
- [Load]ボタン

画面

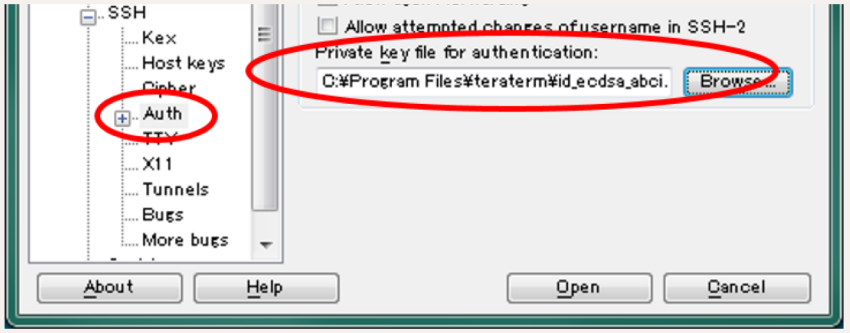


インタラクティブノードへの接続（1/2）

操作説明	画面

SSH秘密鍵の指定

- PUTTY.exe をダブルクリックし起動
  ※ アクセスサーバに接続したものとは別起動
- Category]から、 [Connection] > [SSH] > [Auth] > を選択しメニューを開く
- [Browse]ボタンをクリックし、秘密鍵ファイルを選択する



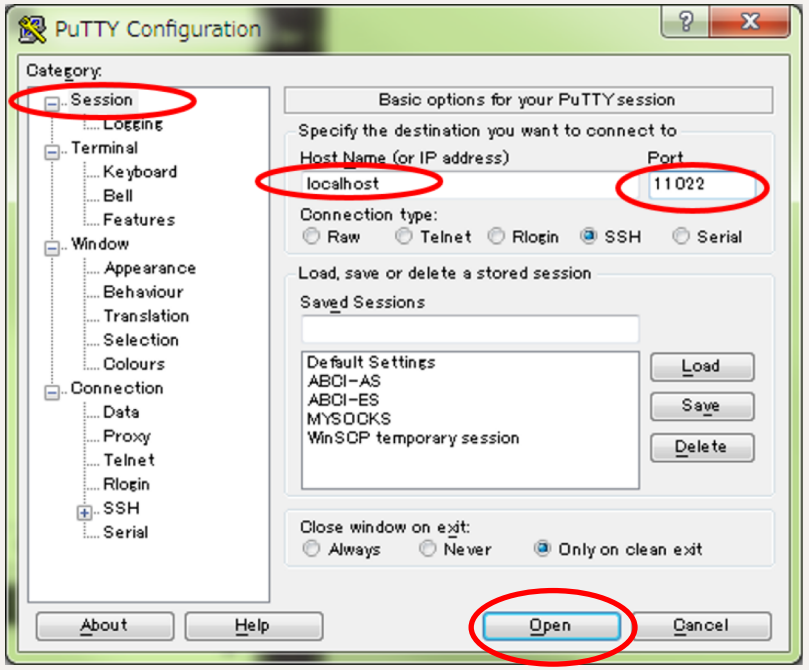
インタラクティブノードの接続設定

- [Category]から、 [Session]を選択してメニューを開く
- インタラクティブノードにポートフォワーディングする接続情報を入力

設定項目	設定値
Host Name	localhost
Port	SSHトンネル設定で指定した値 Ex) 11022

※ SSHトンネルの設定で使用した設定値

- [Open]をクリックしインタラクティブノードに接続開始



インタラクティブノードへの接続 (2/2)

操作説明

アクセスサーバとのSSH認証
- ABCIアカウント名を入力
- SSH鍵を作成時に設定したパスフレーズを入力

※ ログイン時に以下WARNING発生時は[はい]をクリック

※ ログインに成功後は接続設定の保存を推奨。

画面

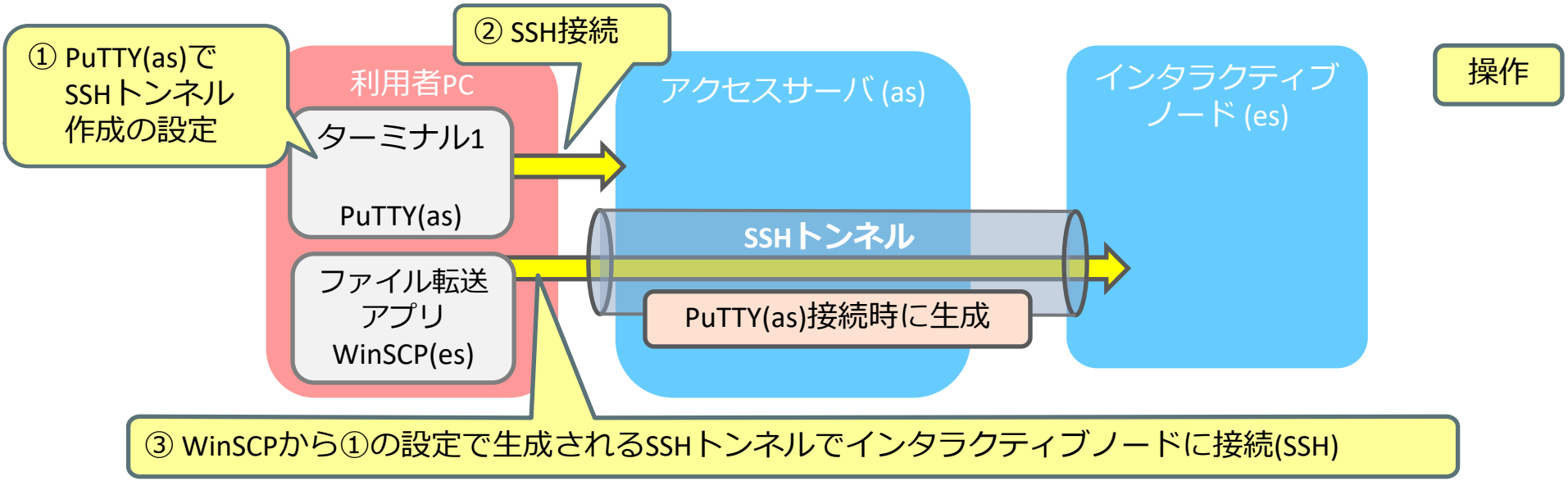
```
login as: abci-account
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
```

パスフレーズを入力（非表示）

```
login as:
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Last login: Thu Jan 10 12:14:38 2019 from 163.220.128.222
------------------------------------------------------------------------------
  ABCI Information                                        Date: Nov 29, 2018
------------------------------------------------------------------------------

  Welcome to ABCI system

  - Operation Schedule
    01/28(Mon) 11:00 - 01/31(Thu) 13:00 Grand Challenge #3

    For more information, please see
    - https://abci.ai/en/about_abci/info.html (In English)
    - https://abci.ai/ja/about_abci/info.html (In Japanese)

  - How to use
    ABCI Users Guide can be found at the ABCI User Portal.

    - https://portal.abci.ai/docs/en/ (In English)
    - https://portal.abci.ai/docs/ja/ (In Japanese)

  If you have any questions or need for further assistance,
  please refer to the following URL and contact us:

    - https://abci.ai/en/how_to_use/user_support.html (In English)
    - https://abci.ai/ja/how_to_use/user_support.html (In Japanese)
```

ABCIとのファイル転送

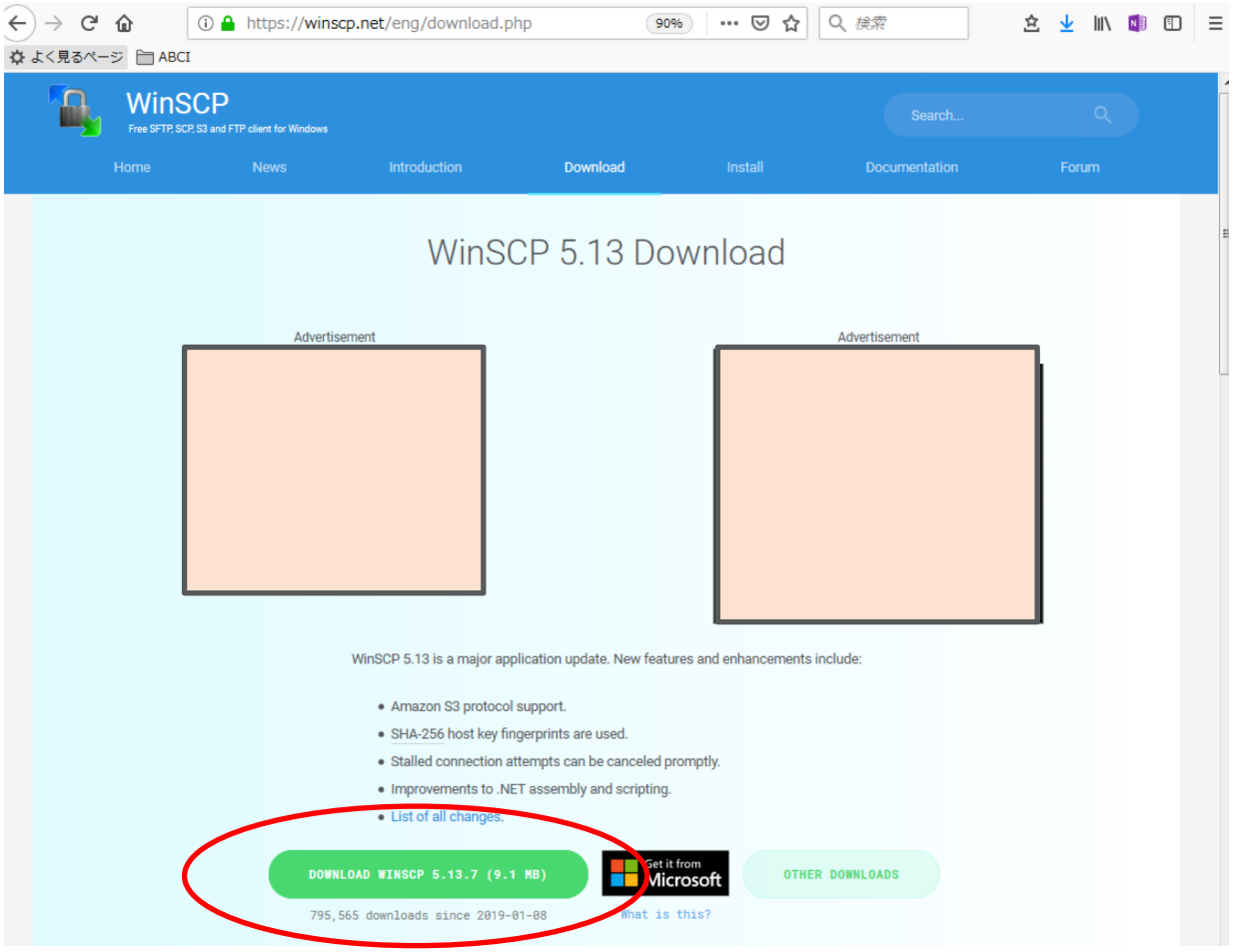
- WinSCPによるファイルのアップ／ダウンロード
  - SSHトンネルを利用してファイルの転送が可能
    - アクセスサーバへの接続が事前に完了していること
    - PuTTYで作成したSSH鍵ファイルが必要



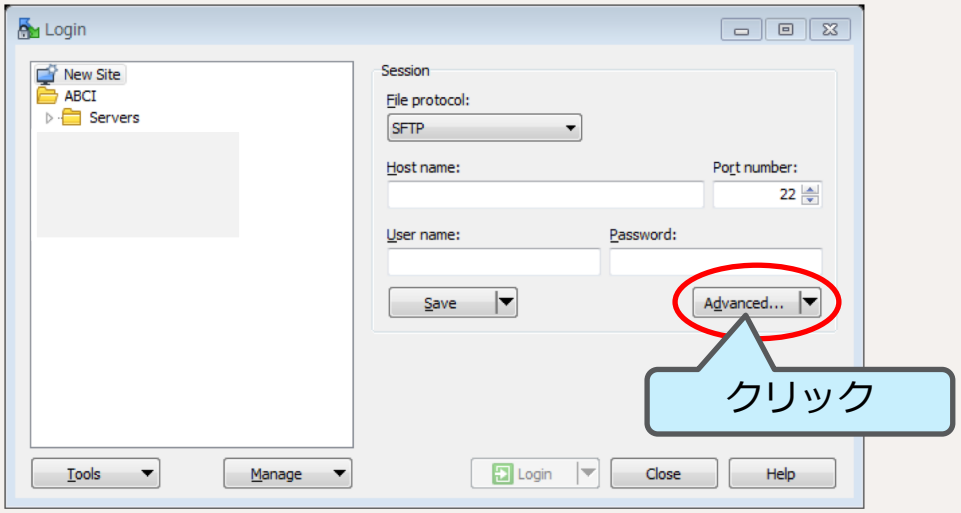
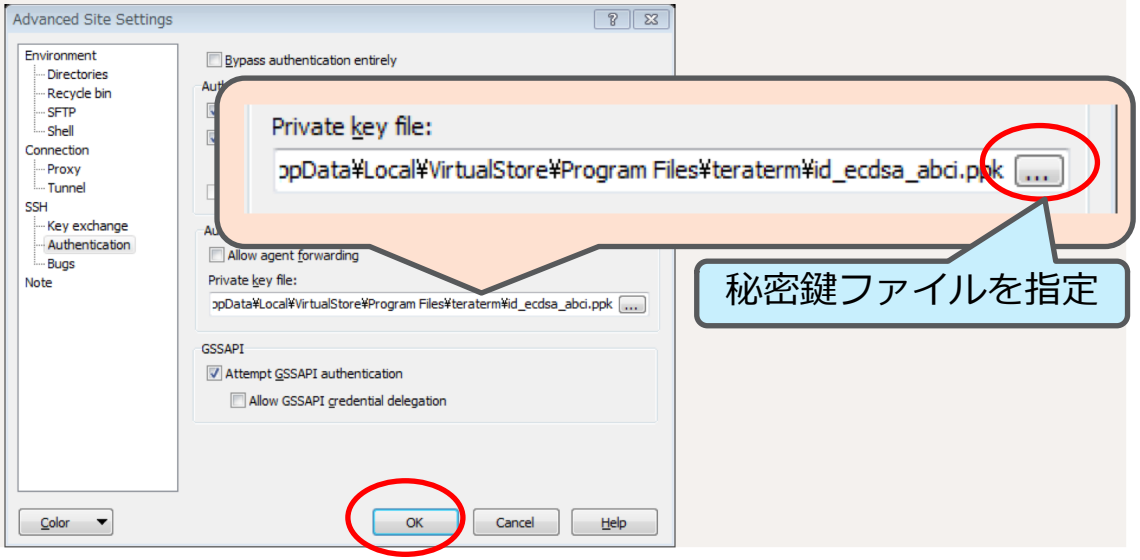
ご参考：WinSCPの入手先

- URL

  https://winscp.net/eng/download.php/



WinSCPでのログイン (1/3)

操作説明	画面
WinSCPのAdvanced設定画面の表示 • WinSCPを起動 • [Advanced|▼]をクリック	
SSH秘密鍵を設定 • Advanced site Settings 画面左ペイン [SSH] > [Authentication] を選択 • [...]をクリックして秘密鍵ファイルを指定 • [OK]をクリックして設定終了	

WinSCPでのログイン(2/3)

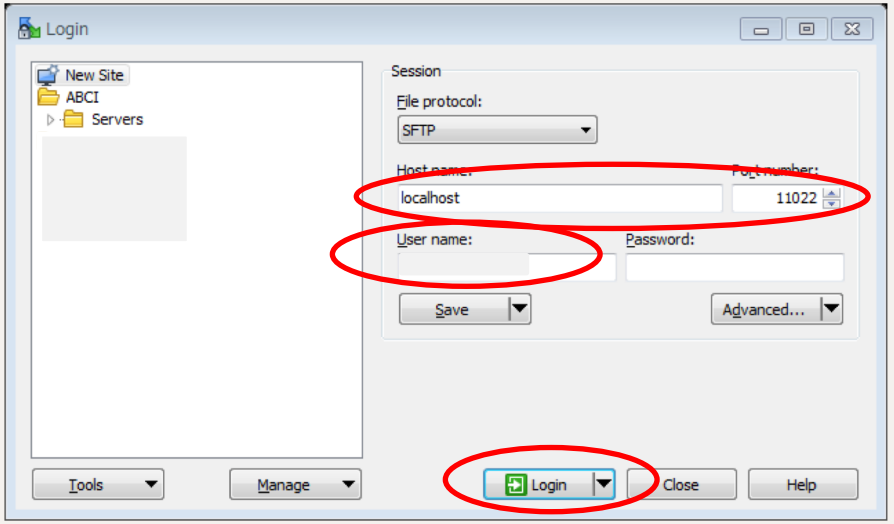
操作説明	画面

インタラクティブノードへの接続設定

- SSHトンネルを使用してインタラクティブノードに接続する以下表の設定

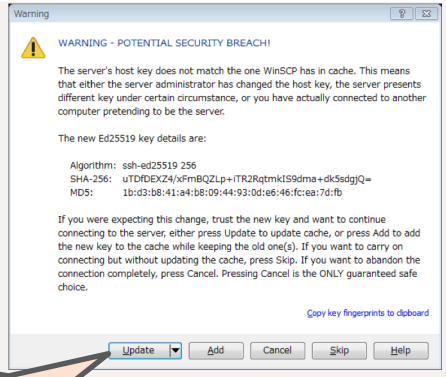
設定項目	設定値
Host name	localhost
Port number	SSHトンネル設定の設定値 ex) 11022
User name	ABCIアカウント名

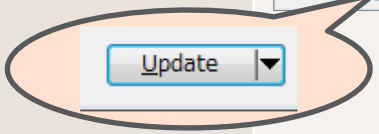
- [Login]をクリックして接続開始

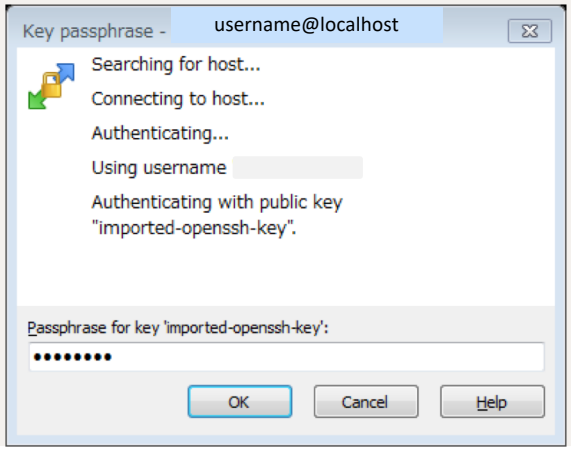


SSH秘密鍵を設定

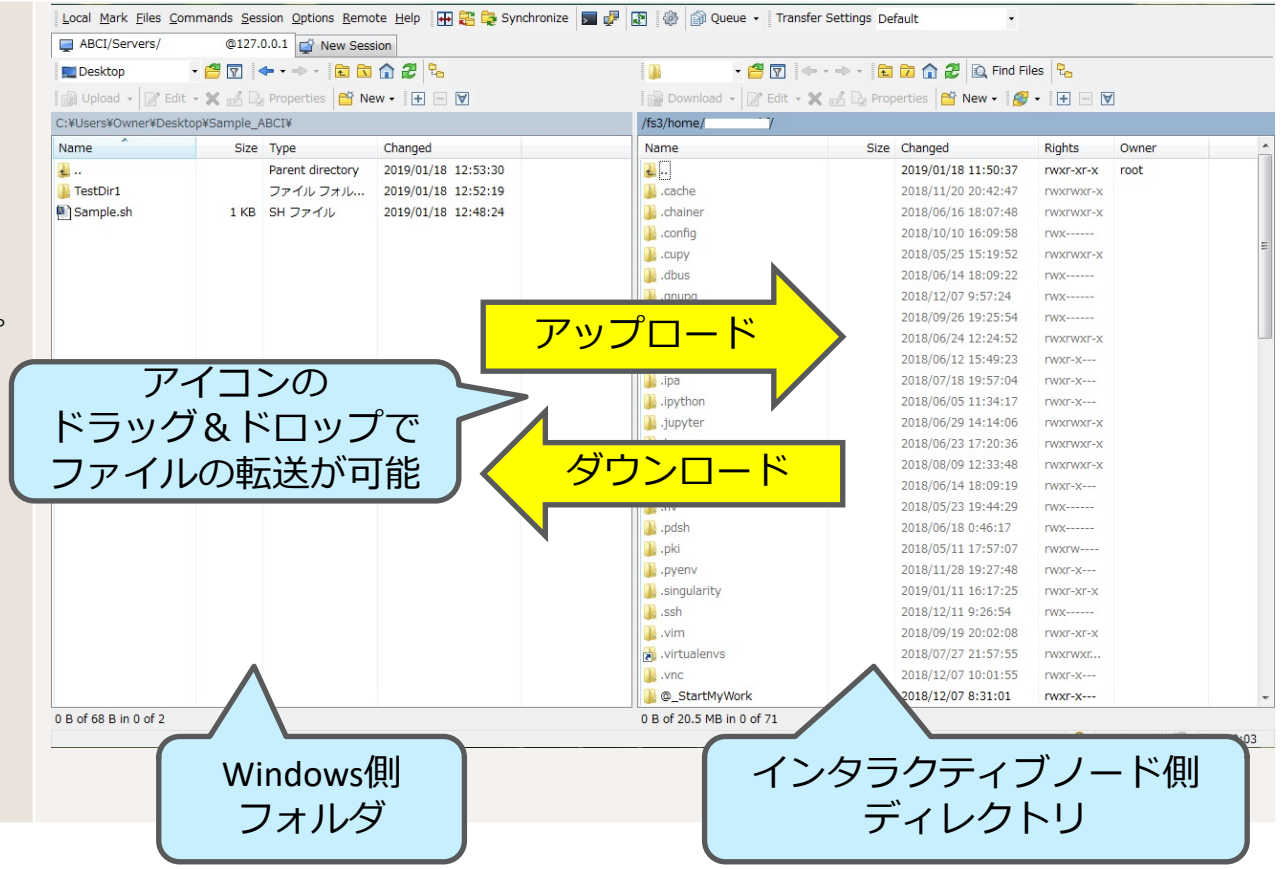
- WARNINGが表示された場合は[Update]
- SSH鍵生成時のパスフレーズを入力
- [OK]をクリック



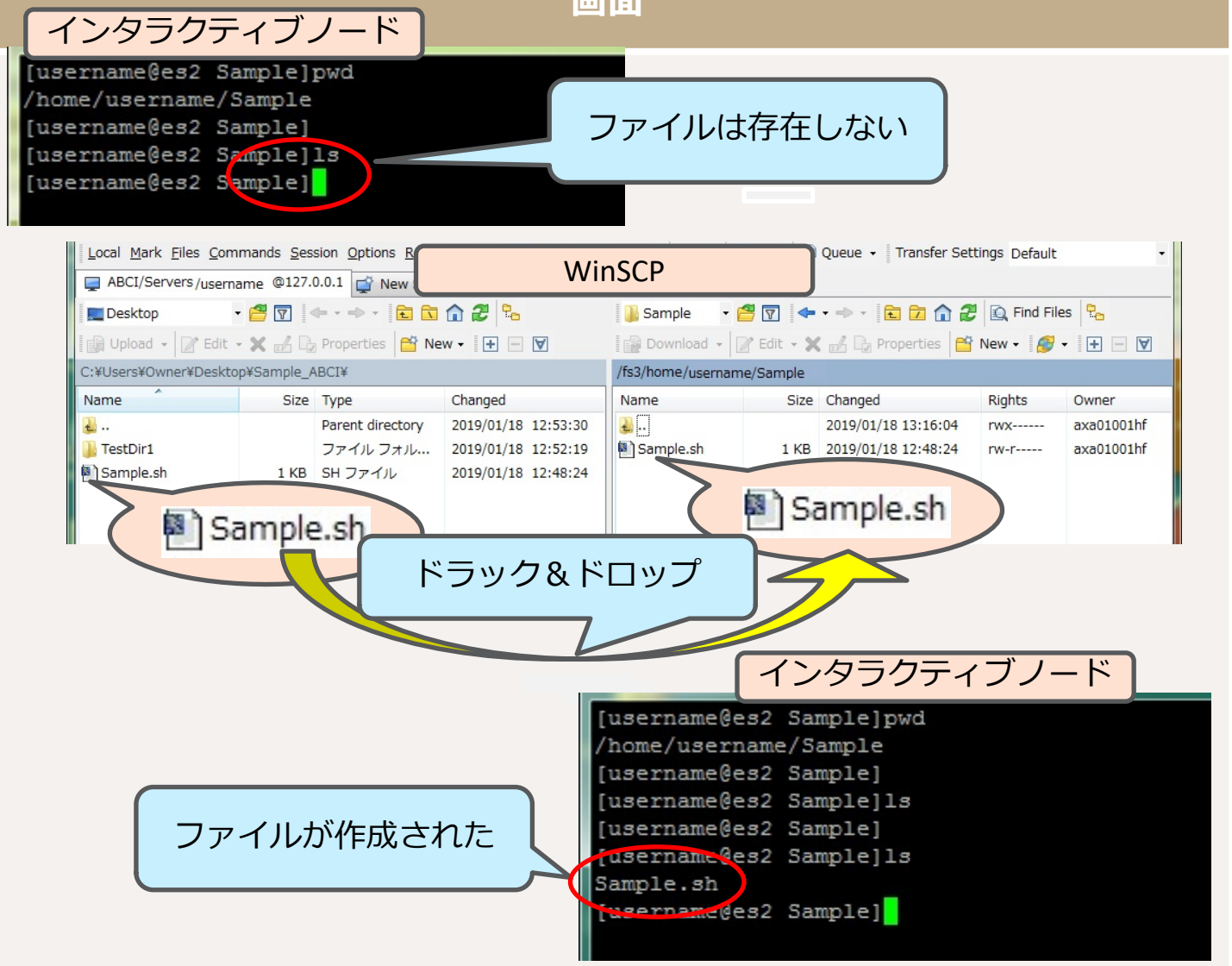




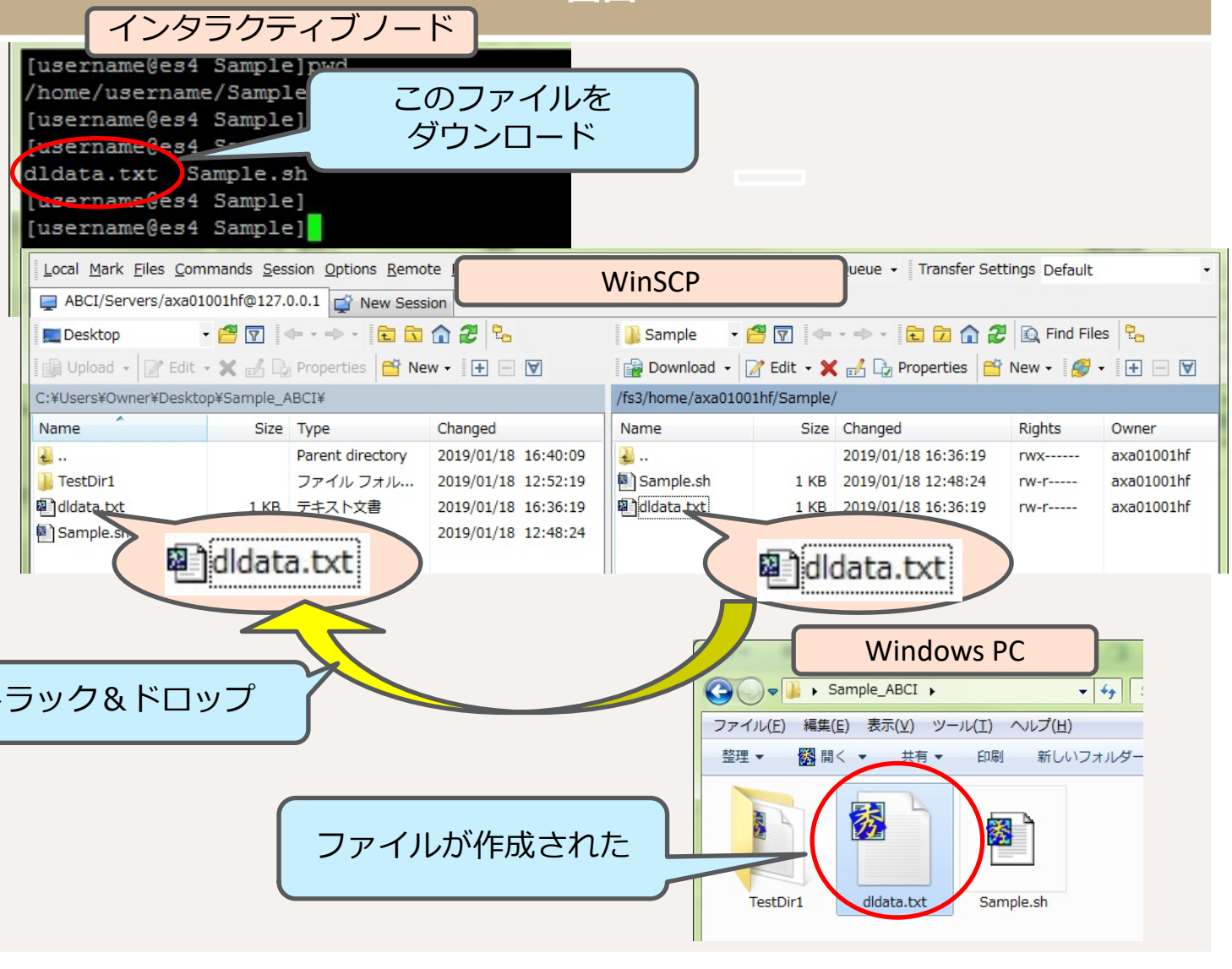
WinSCPでのログイン(3/3)

操作説明	画面
ログイン成功 ログインに成功すると右画面が表示される。 • 左：利用者Windowのフォルダ • 右：インタラクティブノード(es#)のディレクトリ • アイコンのドラッグ＆ドロップでファイルの転送が可能 - 左→右 ：アップロード 利用者Win→ es - 左←右 ：ダウンロード es → 利用者Win	

WinSCPによるファイルアップロード

操作説明	画面
アップロード ABCI インタラクティブノード ↑ 利用者Windows	

WinSCPによるファイルダウンロード

操作説明	画面
ダウンロード ABCI インタラクティブノード ↓ 利用者Windows	

ジョブの実行

- ジョブの実行サービスは３つ

今回ご紹介の範囲

  - Spotサービス
    - ジョブ実行スクリプト作成しジョブスケジューラにバッチ処理依頼
  - On-demandサービス
    - ジョブスケジューラに計算ノードの確保を依頼し、計算ノードでプログラムを実行。
  - Reservedサービス
    - 事前に計算ノードの予約をジョブスケジューラに依頼するサービス
- ジョブ実行時に指定が必須なパラメータ
  - 自分が所属するABCIグループ
  - 利用する計算リソースの量

ジョブの実行

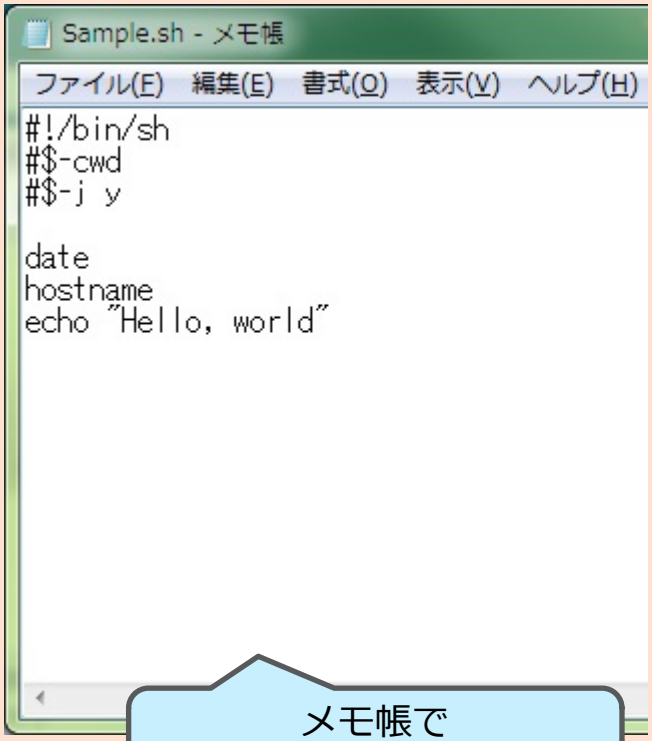
- ジョブ計算リソース
  - ジョブが必要とする資源量を資源タイプ名と数量で指定

資源タイプ	資源タイプ名	説明	割り当て物理CPUコア数	割り当てGPU数	メモリ(GiB)	ローカルストレージ(GB)	資源タイプ課金係数
Full	rt_F	ノード占有	40	4	360	1440	1.00
G.large	rt_G.large	ノード共有 GPU利用	20	4	240	720	0.90
G.small	rt_G.small	ノード共有 GPU利用	5	1	60	180	0.30
C.large	rt_C.large	ノード共有 CPUのみ利用	20	0	120	720	0.60
C.small	rt_C.small	ノード共有 CPUのみ利用	5	0	30	180	0.20

ジョブの実行（ Spotサービス）(1/4)

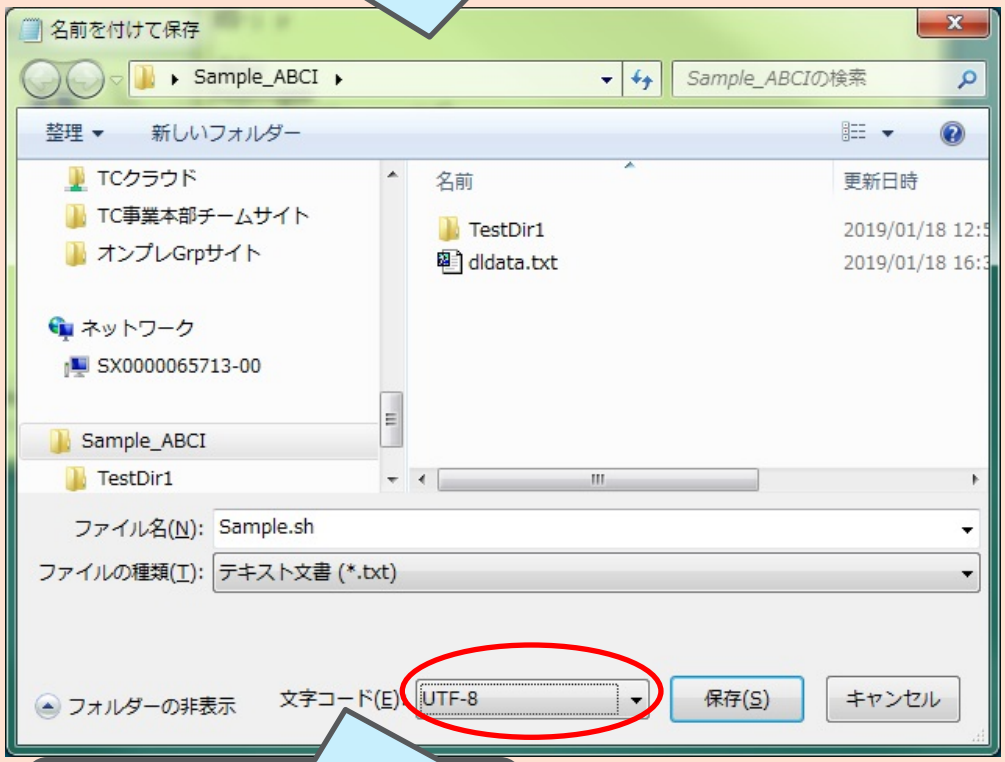
- ジョブ実行スクリプトの作成
  - 各自の端末で実行スクリプトファイルを作成

利用者PC（Win）



メモ帳で
バッチスクリプトを作成

名前をつけて保存
「Sample.sh」



文字モードを
「UTF-8」で保存

ジョブの実行（ Spotサービス）(2/4)

- ジョブ実行スクリプトのアップロード
  - インタラクティブノードにジョブ実行用のディレクトリを作成
  - 利用者PCからWinSCPで上記ディレクトリにスクリプトファイルをアップロード
  - インタラクティブノードで、アップロードしたファイルの改行コードを変更

インタラクティブノード

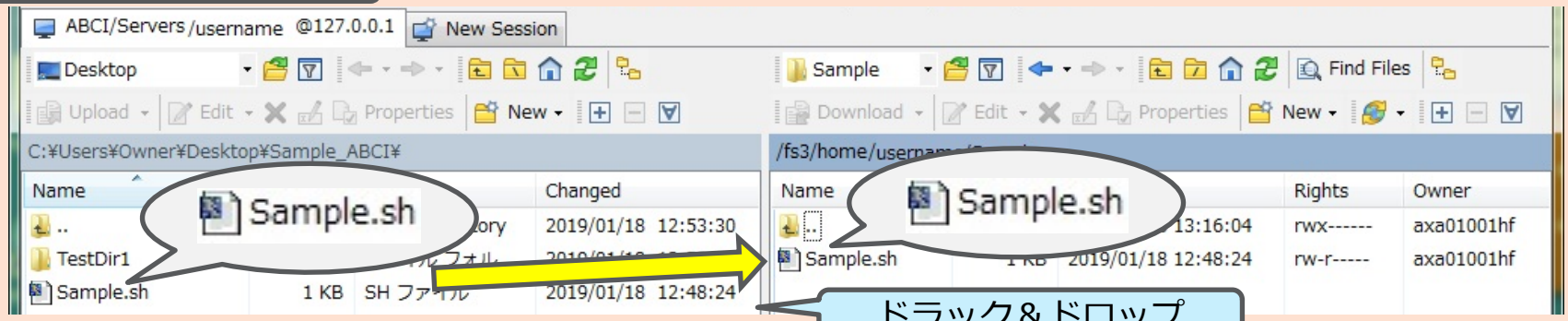
```
[username@es3 ~]$ mkdir Sample
[username@es3 ~]$ cd Sample
[username@es3 Sample]$
```

バッチジョブ実行用のディレクトリを作成
ディレクトリ名：Sample

~/Sampleディレクトリに移動

利用者PC（Win）

※ WinSCPの使用方法は前章「ABCIとのファイル転送」を参照



インタラクティブノード

```
[username@es3 Sample]$ nkf –Lu ./Sample.sh > sample.sh
```

改行コードをWin→Linuxに変更し
sample.sh に保存する。

ジョブの実行（ Spotサービス）(3/4)

- バッチジョブの実行

コマンド

$ qsub –lリソースタイプ=数量 –g グループ名 実行スクリプトのパス

インタラクティブノード

実行例

```
[username@es3 Sample]$ cat sample.sh
#!/bin/sh
#$-cwd
#$-j y

date
hostname
echo "Hello, world"
[username@es3 Sample]$ chmod u+x ./sample.sh
[username@es3 Sample]$ ls -l
total 0
-rwxr--r-- 1 username usergroup 61 Jan 11 13:12 sample.sh
[username@es3 Sample]$
[username@es3 Sample]$  qsub -l rt_G.small=1 -g usergroup ./sample.sh
Your job 114539 ("sample.sh") has been submitted
```

ジョブ実行スクリプト

実行スクリプトに実行権を付与

実行権”x”がついていること

リソース量の指定

グループの指定

ジョブの実行（ Spotサービス）(4/4)

- 実行結果の確認

インタラクティブノード

実行例

```
[username@es3 Sample$ ls -ltr        ← ディレクトリ内の一覧表示 (新しいものが下)
total 0
-rw-r----- 1 username groupname 23 Jan 18 16:36 dldata.txt
-rw-r----- 1 username groupname 66 Jan 18 19:04 Sample.sh
-rwxr----- 1 username groupname 59 Jan 18 19:05 sample.sh
-rw-r--r-- 1 username groupname   59 Jan 18 19:12 sample.sh.o 114539
[username@es3 Sample$
[username@es3 Sample$ cat sample.sh.o 114539        ← 実行結果の確認
Fri Jan 18 19:12:59 JST 2019
g0015.abci.local
Hello, world
[axa01001hf@es3 Sample]$
```

ジョブの標準／エラー出力ファイル

計算ノードで実行した結果

ジョブの実行（ On-demand サービス）

- インタラクティブジョブの実行

コマンド

```
$ qrsh –lリソースタイプ=数量 –g グループ名
```

実行例

```
[username@es3 Sample]$ qrsh -l rt_G.small=1 –g usergroup
[username@g0004 ~]$
[username@g0004 ~]$ uname -n
g0004.abci.local
[username@g0004 ]$ cd Sample
[username@g0004 Sample]$ ls -l
total 0
-rwxr--r-- 1 username usergroup 61 Jan 11 13:12 sample.sh
[username@g0004 Sample]$
[username@g0004 Sample]$ ./sample.sh
Fri Jan 11 13:59:51 JST 2019
g0004.abci.local
Hello, world
[username@g0004 Sample]$
```

リソース量の指定

グループの指定

計算ノードにログインしている

前項のサンプルスクリプト

計算ノードで実行した結果

ジョブの実行状態の確認

- インタラクティブジョブの実行

コマンド

```
$ qstat
```

qstatステータス	説明
r	ジョブ実行中
qw	ジョブはスケジューラに受け付けられているが、実行待ち
Eqw	何らかのエラーによりジョブ実行開始不可。（ポイント不足等）

実行例（バッチジョブ）

```
[username@es3 Sample]$ qstat
job-ID  prior   name       user         state submit/start at     queue                          jclass                         slots ja-task-ID
------------------------------------------------------------------------------------------------------------------------------------------------
 114535 0.25586 sample.sh  username     r     01/11/2019 13:12:55 gpu@g0016                      10
```

実行例（インタラクティブジョブ）

```
[@es1 mnist]$ qstat
job-ID  prior   name       user         state submit/start at     queue                          jclass                         slots ja-task-ID
------------------------------------------------------------------------------------------------------------------------------------------------
 114632 0.28027 QRLOGIN    username     r     01/11/2019 13:56:57 gpu@g0004                      10
```

environment moduleの利用

- アプリ実行環境の適用
  - ABCIで用意されたアプリ実行環境（環境変数）を設定可能

コマンド

$ module [avail] [load MODULE] [unload MODULE] [list]

オプション	説明
avail	利用可能な environment module の一覧を表示。av に省略可。
load	module の読み込み
unload	module の解除
list	現在読み込んでいる module をリスト表示

実行例

```
[username@g0004 Sample]$  which python3
/usr/bin/which: no python3 in (/apps/.....)
[username@g0004 Sample]$  module load python/3.6/3.6.5
[username@g0004 Sample]$
[username@g0004 Sample]$ which python3
/apps/python/3.6.5/bin/python3
```

Python3環境は未設定

python 3.6.5 環境をロード

Python3の環境が設定された

Pythonの利用 (1/5)

- ABCIで利用可能なPython環境
  - module コマンドで確認が可能

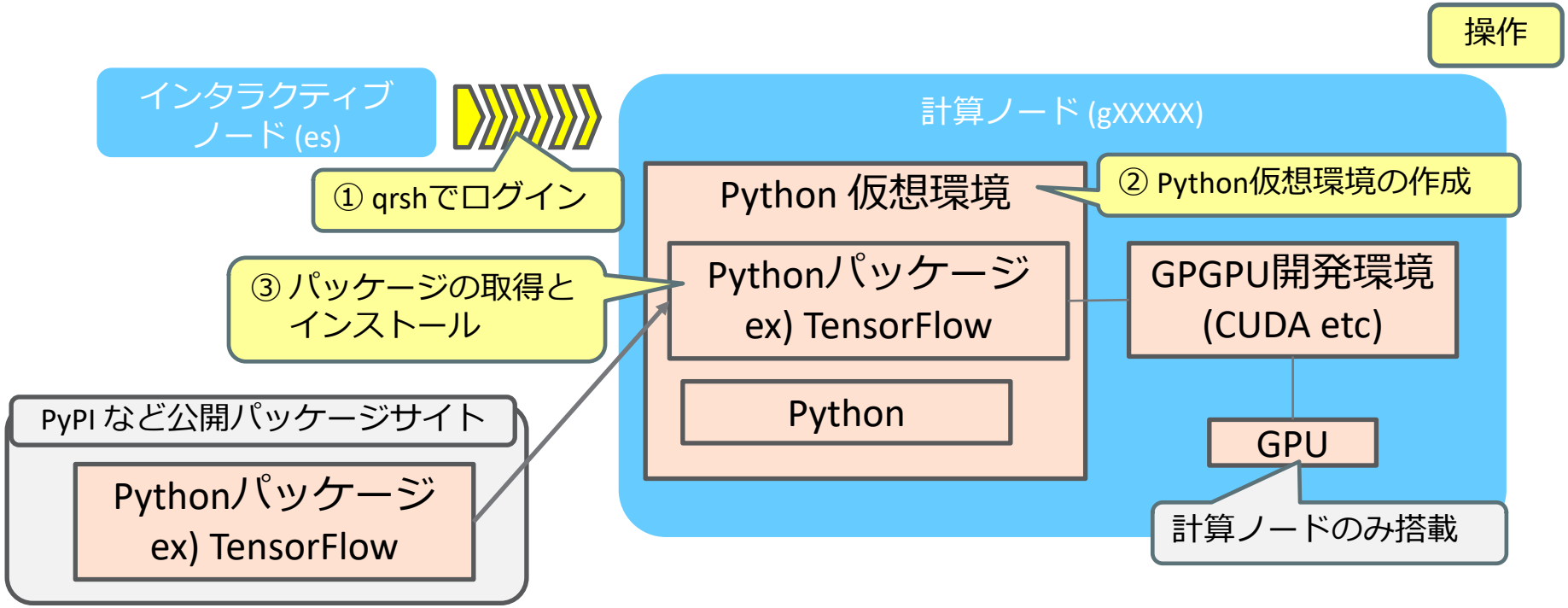
実行例

```
$ module avail
---------------------- /apps/modules/modulefiles/devtools -----------
cmake/3.11.4            openjdk/1.7.0.141     python/3.5/3.5.5
go/1.11.2               openjdk/1.8.0.131     python/3.6/3.6.5
intel-mkl/2018.2.199    python/2.7/2.7.15     R/3.5.0
openjdk/1.6.0.41        python/3.4/3.4.8
```

- Pythonパッケージをインターネットから取得可能
  - インタラクティブノードおよび計算ノードでは、PyPIなどインターネット公開されたPythonパッケージの取得可能
  - AIフレームワークを自分の環境に構築可能

Pythonの利用 (2/5)

- アプリケーションの実行環境は計算ノードで作成を推奨
  - GPGPUは計算ノードに搭載しているため
  - インタラクティブジョブとして計算ノードにログインし環境構築
- Python仮想環境の使用を推奨
  - 用途ごとに個別のアプリ環境を作成可能



Pythonの利用 (3/5)

- TensorFlow GPUのインストールと実行

Python 仮想環境の作成
- ここでは venv を使用

TensorFlow GPUのインストール
- pip install コマンドでインターネットからパッケージを取得、インストール

```
[username@es3 ~]$ qrsh -l rt_G.small=1 -g groupname
[username@g0003 ~]$
[username@g0003 ~]$ module load python/3.6/3.6.5
[username@e0003 ~]$ python3 -m venv ~/Sample/v_tf_gpu
[username@g0003 ~]$ source ~/Sample/v_tf_gpu/bin/activate
(v_tf_gpu) [username@g0003 ~]$ which python
/fs3/home/groupname/Sample/v_tf_gpu/bin/python
(v_tf_gpu) [username@g0003 ~]$
(v_tf_gpu) [username@g0003 ~]$ module load cuda/9.0/9.0.176.4
(v_tf_gpu) [username@g0003 ~]$ module load cudnn/7.4/7.4.2
(v_tf_gpu) [username@g0003 ~]$
(v_tf_gpu) [username@g0003 ~]$ pip3 install --ignore-installed --upgrade
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.12.0-cp36-cp36m-linux_x86_64.whl
                              :
Successfully built ... tensorflow-gpu-1.12.0 …
(v_tf_gpu) [username@g0003 ~]$ deactivate
```

インタラクティブジョブの投入
計算ノードが割り当てられた
仮想環境の作成
仮想環境の開始
pythonのパスが作成された
GPGPU使用環境の読み込み
tenforflow-gpu-1.12.0 のインストール
仮想環境の終了

Pythonの利用 (4/5)

サンプルコード (https://www.tensorflow.org/tutorials/?hl=ja)

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Pythonの利用 (5/5)

- PythonでTensorFlowを実行してみる

```
[username@g0001 ~]$ python3 -m venv ~/Sample/v_tf_gpu
[username@g0001 ~]$ source ~/Sample/v_tf_gpu/bin/activate
(v_tf_gpu) [axa01001hf@g0003 ~]$ python
Python 3.6.5 (default, Jun  2 2018, 15:49:50)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> mnist = tf.keras.datasets.mnist
>>> (x_train, y_train),(x_test, y_test) = mnist.load_data()
>>> x_train, x_test = x_train / 255.0, x_test / 255.0
>>>
>>> model = tf.keras.models.Sequential([
...   tf.keras.layers.Flatten(input_shape=(28, 28)),
...   tf.keras.layers.Dense(512, activation=tf.nn.relu),
...   tf.keras.layers.Dropout(0.2),
...   tf.keras.layers.Dense(10, activation=tf.nn.softmax)
... ])
>>> model.compile(optimizer='adam',
...               loss='sparse_categorical_crossentropy',
...               metrics=['accuracy'])
>>> model.fit(x_train, y_train, epochs=5)
Epoch 1/5
2019-02-04 16:38:34.256067: I
tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports
instructions that this TensorFlow binary was not compiled to use: AVX2
AVX512F FMA
2019-02-04 16:38:34.648621: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0
with properties:
name: Tesla V100-SXM2-16GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0000:3d:00.0
totalMemory: 15.78GiB freeMemory: 15.37GiB
2019-02-04 16:38:34.648723: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu
devices: 0
2019-02-04 16:38:36.224484: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect
StreamExecutor with strength 1 edge matrix:
2019-02-04 16:38:36.224530: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2019-02-04 16:38:36.224539: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:   N
2019-02-04 16:38:36.224848: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow
device (/job:localhost/replica:0/task:0/device:GPU:0 with 14874
MB memory) -> physical GPU (device: 0, name: Tesla V100-SXM2-16GB, pci bus id:
0000:3d:00.0, compute capability: 7.0)
60000/60000 [==============================] - 8s 127us/step - loss: 0.2215 -
acc: 0.9341
Epoch 2/5
60000/60000 [==============================] - 3s 49us/step - loss: 0.0975 - acc:
0.9691
Epoch 3/5
60000/60000 [==============================] - 3s 50us/step - loss: 0.0702 - acc:
0.9783
Epoch 4/5
60000/60000 [==============================] - 3s 50us/step - loss: 0.0547 - acc:
0.9825
Epoch 5/5
60000/60000 [==============================] - 3s 50us/step - loss: 0.0434 - acc:
0.9860
<tensorflow.python.keras.callbacks.History object at 0x2b8c1553df60>
>>> model.evaluate(x_test, y_test)
10000/10000 [==============================] - 0s 27us/step
[0.07372516659436515, 0.9788]
>>>
```

コンテナ（Singularity）の利用(1/3)

- Singularity
  - ABCIではLinuxコンテナとしてSingularityが利用可能
  - 利用可能なバージョンは、moduleコマンドで確認可

実行例

```
$ module avail
---------------------- /apps/modules/modulefiles/runtimes -----------------
hadoop/2.9.1        singularity/2.6.1 spark/2.3.2
hadoop/2.9.2        spark/2.3.1       spark/2.4.0
```

  - Docker HUB から取得したDockerイメージを Singularityイメージに変換し利用が可能

実行例 （コマンド実行例は次項で説明）

- Python3, GPU環境で構築された TensorFLow のDockerイメージをSingularityで使用
  Dockerイメージ URL：https://hub.docker.com/r/tensorflow/tensorflow/
  tag ： -gpu-py3

コンテナ（Singularity）の利用 (2/3)

- TensorFlowのコンテナ取得と実行

Dockerイメージ からSingularityイメージの取得

- 使用イメージ (tag: -gpu-py3) https://hub.docker.com/r/tensorflow/tensorflow/

```
[username@es3 ~]$ cd ~/Sample
[username@es3 Sample]$ module load  singularity/2.6.1
[username@es3 Sample]$ singularity pull docker://tensorflow/tensorflow:latest-gpu-py3
Docker image path: index.docker.io/tensorflow/tensorflow:latest-gpu-py3
Cache folder set to /fs3/home/axa01001hf/.singularity/docker
[17/17] |==================================| 100.0%
                                    :
Done. Container is at: ./tensorflow-latest-gpu-py3.simg
[username@es3 Sample]$ ls
tensorflow-latest-gpu-py3.simg
```

イメージファイルの取得と作成

.simg ファイルが作成される

実行

```
[username@es3 ~]$ qrsh –l rt_G.small=1 –g groupname
[username@es3 ~]$ cd ~/Sample
[username@g0003 Sample]$ module load  singularity/2.6.1
[username@g0003 Sample]$ singularity shell –nv ./tensorflow-latest-gpu-py3.simg
Singularity: Invoking an interactive shell within container…

Singularity tensorflow-latest-gpu-py3.simg:~/Sample>
```

コンテナ（Singularity）の実行

プロンプトが変化

コンテナ（Singularity）の利用(3/3)

- TensorFlow の実行例　(前項のpythonと同じサンプル)

```
[usename@g0003 Sample]$ module load singularity/2.6.1
[usename@g0003 Sample]$ singularity shell --nv ./tensorflow-latest-gpu-py3.simg
Singularity: Invoking an interactive shell within container...
Singularity tensorflow-latest-gpu-py3.simg:~/Sample> python
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> mnist = tf.keras.datasets.mnist
>>> (x_train, y_train),(x_test, y_test) = mnist.load_data()
>>> x_train, x_test = x_train / 255.0, x_test / 255.0
>>>
>>> model = tf.keras.models.Sequential([
...   tf.keras.layers.Flatten(input_shape=(28, 28)),
...   tf.keras.layers.Dense(512, activation=tf.nn.relu),
...   tf.keras.layers.Dropout(0.2),
...   tf.keras.layers.Dense(10, activation=tf.nn.softmax)
... ])
>>> model.compile(optimizer='adam',
...               loss='sparse_categorical_crossentropy',
...              metrics=['accuracy'])
>>> model.fit(x_train, y_train, epochs=5)
Epoch 1/5
2019-02-04 11:07:42.855674: I tensorflow/core/platform/cpu_feature_guard.cc:141]
Your CPU supports instructions that this TensorFlow binary was not compiled to use:
AVX2 AVX512F FMA
2019-02-04 11:07:43.257930: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 0 with
properties:
name: Tesla V100-SXM2-16GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0000:3e:00.0
totalMemory: 15.78GiB freeMemory: 15.37GiB
2019-02-04 11:07:43.257978: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu
devices: 0
2019-02-04 11:07:43.747168: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect
StreamExecutor with strength 1 edge matrix:
2019-02-04 11:07:43.747217: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0
2019-02-04 11:07:43.747227: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:   N
2019-02-04 11:07:43.747473: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow
device (/job:localhost/replica:0/task:0/device:GPU:0 with 14874 MB memory) ->
physical GPU (device: 0, name: Tesla V100-SXM2-16GB, pci bus id: 0000:3e:00.0,
compute capability: 7.0)
60000/60000 [==============================] - 5s 77us/step - loss: 0.2212 - acc:
0.9346
Epoch 2/5
60000/60000 [==============================] - 3s 52us/step - loss: 0.0957 - acc:
0.9712
Epoch 3/5
60000/60000 [==============================] - 3s 52us/step - loss: 0.0690 - acc:
0.9790
Epoch 4/5
60000/60000 [==============================] - 3s 52us/step - loss: 0.0524 - acc:
0.9828
Epoch 5/5
60000/60000 [==============================] - 3s 52us/step - loss: 0.0424 - acc:
0.9864
<tensorflow.python.keras.callbacks.History object at 0x2b9f6feb7f28>
>>> model.evaluate(x_test, y_test)
10000/10000 [==============================] - 0s 30us/step
[0.06862098166315118, 0.9798]
```

Jupyter Notebook の利用(1/5)

- Jupyter Notebook
  - ブラウザで利用するインタラクティブな計算環境
  - ABCIではSSHトンネルの応用で起動が可能
    - インタラクティブノードから計算ノードにSSHトンネルを設定
  - 今回は、前項の TensorFlow のPython環境に導入
    - TensorFlowをJupyter Notebookから使用

Jupyter Notebookの使用 (2/5)

- インストールとサービス起動

インストール（前項のTensorFlow環境にインストール）

```
[username@es3 ~]$ module load python/3.6/3.6.5
[username@es3 ~]$ source ~/Sample/v_tf_gpu/bin/activate
(v_tf_gpu) es3 $ pip install --upgrade pip
(v_tf_gpu) es3 $ pip install jupyter
(v_tf_gpu) es3 $ deactivate
```

← 前項「Pythonの利用」環境を使用

On-demand サービスでの Jupyter Notebook の起動

- 計算ノードでブラウザを開かず(--no-browser)、IPアドレスに計算ノード(--ip=`hostname`)を指定し起動。
- 計算ノード名「g0004」とトークン「token=e7f0ba….」は、後でコピペして利用する。
- トークン直前のポート番号は、デフォルト8888。変更される場合があるので注意すること。

```
[username@es3 ~]$ qrsh –l rt_F=1 –g グループ名 –l h_rt=01:00:00
[username@g0004 ~]$ module load python/3.6/3.6.5
[username@g0004 ~]$ module load cuda/9.0/9.0.176.4 cudnn/7.4/7.4.2
[username@g0004 ~]$ source ~/Sample/v_tf_gpu/bin/activate
(v_tf_gpu) g0004 $ jupyter notebook --no-browser --ip=`hostname` >> jupyter.log 2>&1 &
(v_tf_gpu) g0004 $ jupyter notebook list
http://g0004.abci.local:8888/?token=e7f0ba979d4ffd9eeb7e6debf5a326f853fc289583f92dc5::
/fs3/home/username
```

Jupyter Notebookの使用 (3/5)

- 利用者環境での操作

SSHトンネル設定

(Windows PuTTY)

- 計算ノードのターミナルで[タイトルバ]ーを右クリック＞[Change settings…] > [Connection]>[SSH]>[Tunnels]を選択
- ログイン時のSSHトンネル作成手順を参照に以下設定値のトンネルを作成する.

  Source port ：18888

  Destination ： 計算ノードのホスト名:ポート番号

  例) g0004:8888

(Mac/Linux)

```
[username@es3 ~]$ssh -L 18888:g0004:8888 -l usename -i ~/.ssh/id_rsa_hpc-p 10022 localhost
```

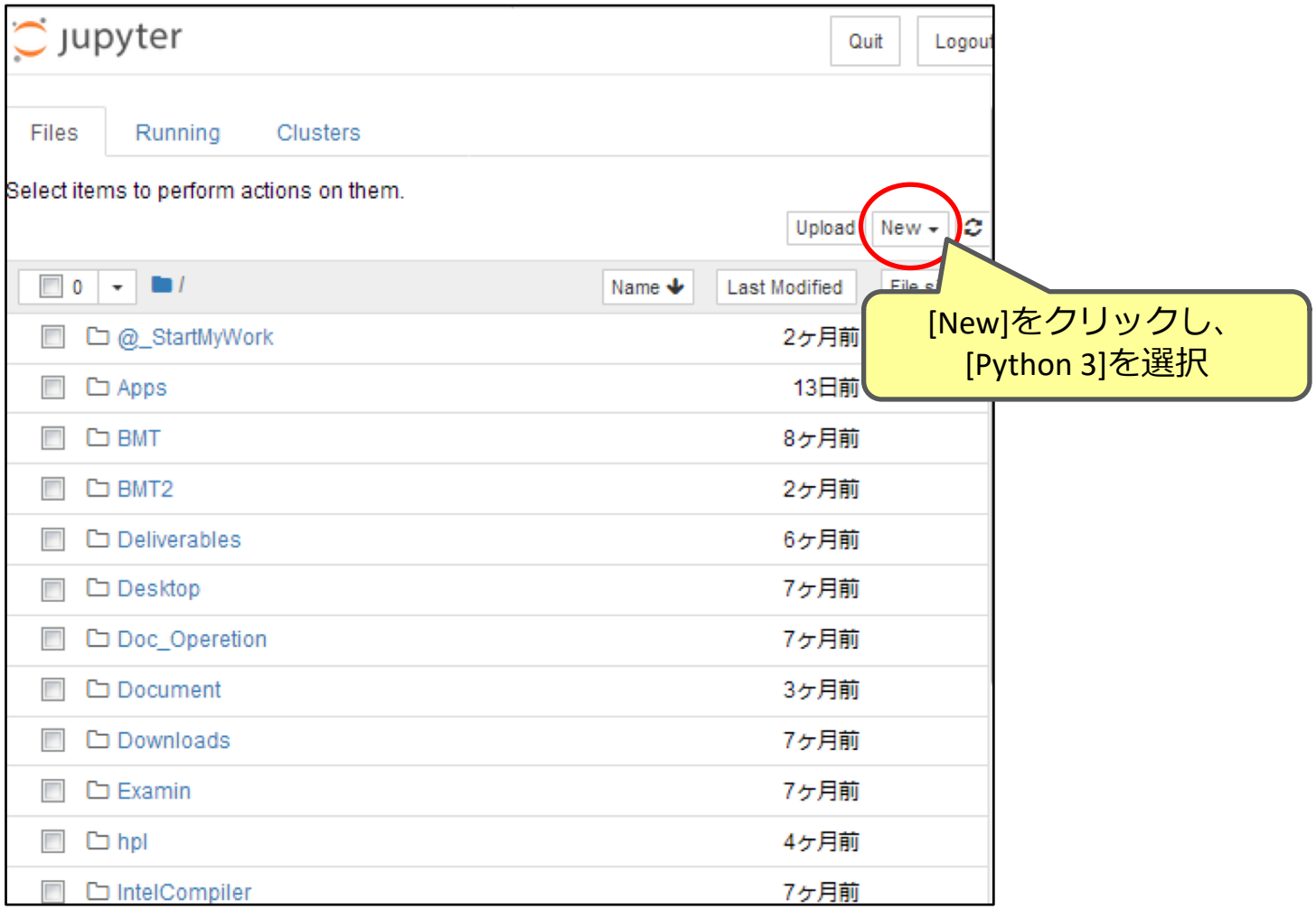
Webブラウザから接続

- ブラウザに以下のURLを指定する。
- 計算ノードで Jupyter Notebook起動時のトークン「token=e7f0b…」を貼り付ける。

```
http://localhost:18888/?token=e7f0ba979d4ffd9eeb7e6debf5a326f853fc289583f92dc5
```

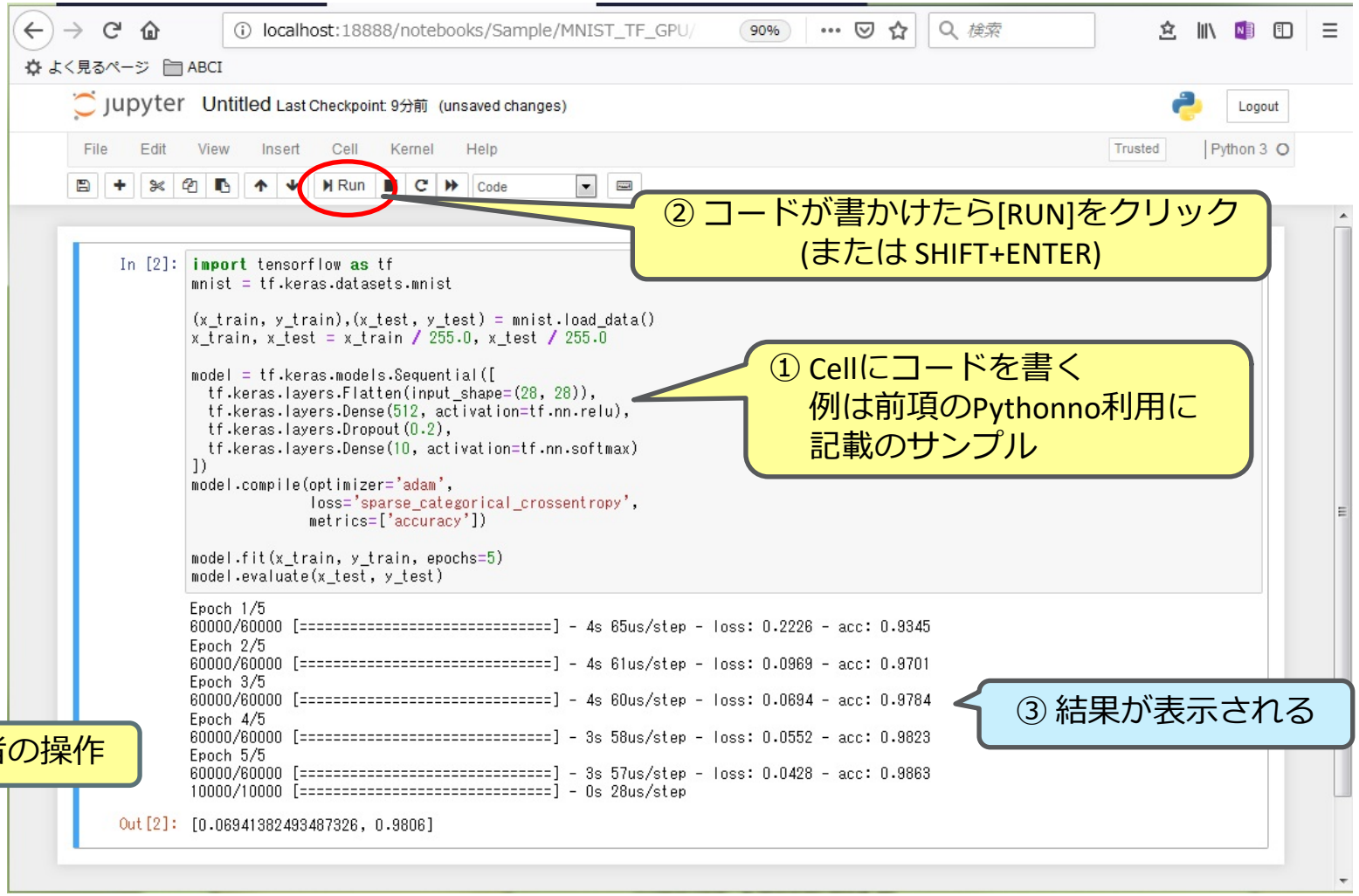
Jupyter Notebookの使用 (4/5)

- 実行例



Jupyter Notebookの使用 (5/5)

- TensorFlowの実行例 (前項Pythonの利用のサンプル)




```
In [2]: import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)

Epoch 1/5
60000/60000 [==============================] - 4s 65us/step - loss: 0.2226 - acc: 0.9345
Epoch 2/5
60000/60000 [==============================] - 4s 61us/step - loss: 0.0969 - acc: 0.9701
Epoch 3/5
60000/60000 [==============================] - 4s 60us/step - loss: 0.0694 - acc: 0.9784
Epoch 4/5
60000/60000 [==============================] - 3s 58us/step - loss: 0.0552 - acc: 0.9823
Epoch 5/5
60000/60000 [==============================] - 3s 57us/step - loss: 0.0428 - acc: 0.9863
10000/10000 [==============================] - 0s 28us/step

Out[2]: [0.06941382493487326, 0.9806]
```

Mac/Linuxによる操作

- SSH鍵の作成
- アクセスサーバへのログイン
- インタラクティブノードへのログイン
- ファイルのアップロード／ダウンロード

SSH鍵の作成（Mac/Linux）

- ssh-keygenコマンドによるSSH鍵生成

コマンド

```
$ ssh-keygen [-t type of key] [-b bits] [-C comment] [-f krl_file]
```

オプション	説明
-t	暗号化の方式を指定。ABCIでは、RSA, ECDSAおよびEd25519をサポート
-b	キーのbit数。ABCIでは、RSAは256bit以上, ECDSAは256/384/521bit, Ed25519の場合は256b itをサポート
-C	コメント。ABCIではメールアドレスを指定。
-f	キーファイルの出力パスとファイル名

実行例

```
[username@es3 Sample]$ ssh-keygen -t rsa -b 4096 -C "xxx@yyy.co.jp" -f ~/.ssh/id_rsa
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ./ssh/id_rsa.
Your public key has been saved in ./.ssh/rsa.pub.
 :
[username@es3 Sample]$
```

メールアドレスを入力

パスワード入力と確認

秘密鍵の出力パス

公開鍵の出力パス

アクセスサーバへのログイン (Mac/Linux)

- sshコマンドによる接続

コマンド

```
$ ssh -L socket:host:hostport -l login_name] hostname
```

オプション	説明
-L	SSHトンネルするローカルのポート番号をsocketに指定。変更可能。 hostにアクセスサーバのホスト名、as.abci.aiを指定。変更不可。 hostport には、SSH接続するポート番号22を指定。変更不可。
-l	ログイン名。ABCIアカウントを指定

実行例

```
[user@yourpc ~]$ssh -L 10022:es:22 -l username as.abci.ai
The authenticity of host 'as.abci.ai (0.0.0.1)' can't be established.
 RSA key fingerprint is XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX.
Are you sure you want to continue connecting (yes/no)? yes
 Warning: Permanently added 'XX.XX.XX.XX' (RSA) to the list of known hosts.
 Enter passphrase for key '/home/username/.ssh/id_rsa':
Welcome to ABCI access server.
Please press any key if you disconnect this session.
```

初回時表示

← yes を入力

← パスフレーズ入力

← ログイン成功時メッセージ

※ このターミナルはキー操作をすると後のSSH接続が切断されるため注意

インタラクティブノードへのログイン (Mac/Linux)

- sshコマンドによる接続
  - アクセスサーバ接続とは異なるターミナルを開いて実施

コマンド

```
$ ssh -p port -l login_name hostname
```

オプション	説明
-p	SSHトンネルに設定したポート番号。
-l	ログイン名。ABCIアカウントを指定。

実行例

```
[user@yourpc ~]$ssh -p 10022 -l username localhost
The authenticity of host 'as.abci.ai (127.0.0.1)' can't be established.
 RSA key fingerprint is XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX.
Are you sure you want to continue connecting (yes/no)? yes
 Warning: Permanently added 'XX.XX.XX.XX' (RSA) to the list of known hosts.
 Enter passphrase for key '/home/username/.ssh/id_rsa':
[username@es3 ~]$
```

初回時表示

← yes を入力

← パスフレーズ入力

← ログイン成功。この場合は es3 にログイン

ファイルの転送(Mac/Linux)

- scpコマンドによるファイルの転送

コマンド（アップロード）

```
$ scp -P port localfile username@localhost:/remote-file
```

コマンド (ダウンロード)

```
$ scp -P port username@localhost:/remote-file localfile
```

オプション	説明
-P	SSHトンネルに設定したポート番号。
localfile	利用者PC上のディレクトリ／ファイルパス
remote-dir-file	リモートのディレクトリ／ファイルパス

実行例（アップロード）

```
[user@yourpc ~]$scp -P 10022 localfile usename@localhost:/remote-dir
Enter passphrase for key: ++++++++        ← パスフレーズ

local-file    100% |**********************|  file-size  transfer-time
[username@es3 ~]$
```

マニュアル

より詳細な情報については、以下を参照下さい。

- ABCIユーザサポート
  https://abci.ai/ja/how_to_use/user_support.html
- ABCI利用に関するFAQ
  https://abci.ai/ja/how_to_use/yakkan.html
- 利用の手引き
  https://portal.abci.ai/docs/ja/
- 利用の手引き（ポータル）
  https://portal.abci.ai/docs/portal/ja/