

ABCI利用マニュアル (MAC編)

国立研究開発法人 産業技術総合研究所

情報・人間工学領域

招聘研究員 萩島功一

2019年2月7日

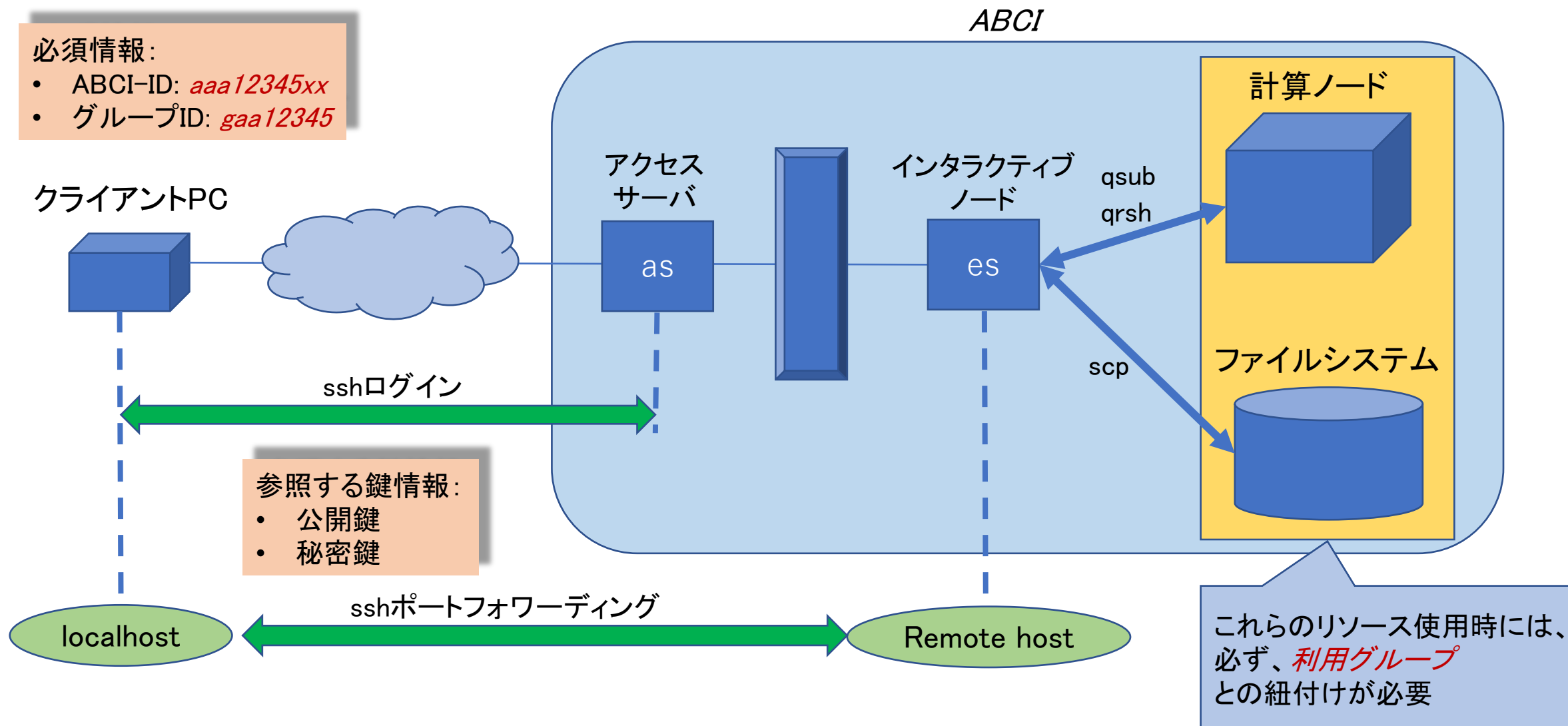
2019年3月25日（改定）

2019年3月27日（追加改定）

目 次

- プロトコル概念図
- 公開鍵・秘密鍵ペアの生成、公開鍵の登録
- ABCIへのログイン(ssh)
- ファイルのアップロード、ダウンロード(scp)
- インタラクティブジョブ(qrsh)
- バッチジョブ(qsub)
- 仮想環境で、TensorFlowをインストール
- qsubでの実行例
- Singularityを利用して、TensorFlowの環境を構築
- Jupyter Notebookの利用

プロトコル概念図



公開鍵・秘密鍵ペアの生成

1) ターミナルを開き、公開鍵・秘密鍵のペアを生成する(ssh-rsaの例)。



```
yourpc$ install -m 0700 -d ~/.ssh
yourpc$ cd ~/.ssh
yourpc$ 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): <-- パスフレーズを入力(任意)
Enter same passphrase again: <-- もう一度、パスフレーズを入力(任意)
Your identification has been saved in id_rsa.
Your public key has been saved in id_rsa.pub.
yourpc$ ls ~/.ssh
id_rsa      id_rsa.pub
```

- * id_rsa (秘密鍵)・・・クライアントPCに配置
- * id_rsa.pub (公開鍵)・・・ABCIに登録

2) ターミナルから公開鍵を参照する(ssh-rsaの例)。

```
yourpc$ cat ~/.ssh/id_rsa.pub
ssh-rsa
abcdefghijklmnopqrstuvwxyz1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
QRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890abcdefghijklmnopqrstuvwxyzABCDEFGH
IJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890abcdefghijklmnopqrstuvwxyz
yzABCDEFGHJKLMNOPQRSTUVWXYZgxVQ7WEhTBbCel xxx@dhcp012345.a01.yyy.co.jp
yourpc$
```

公開鍵の登録 (1/3)

操作説明	画面
<p>利用者ポータル（画像認証）にアクセス</p> <ul style="list-style-type: none">• 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スに入力 <p>利用者ポータル https://portal.abci.ai/user/</p>	 <p>利用者ログイン</p> <p>言語 / Language <input checked="" type="radio"/> 日本語 / Japanese <input type="radio"/> 英語 / English</p> <p>ABCIアカウント名 <input type="text"/></p> <p>画像認証 <input type="text" value="657551"/> <input type="button" value="リロード"/></p> <p>(数字6文字)</p> <p>この場合は「657551」を入力</p>
<p>* 2回目からのアクセス時には、本パスワードを入力</p> <p>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</p> <ul style="list-style-type: none">• ABCIアカウント名を入力• 仮パスワードを入力 <div data-bbox="420 992 1095 1342"><p>xxxxx 様</p><p>通知メール文例</p><p>このメールはABCI利用ポータルに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。</p><p>https://portal.abci.ai/user/login.php?action=input_passw&login_url=XXXXXXXXXXXXXXXXXXXX</p><p>※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。</p></div>	 <p>ABCI利用ポータル</p> <p>ABCIアカウント名 <input type="text"/></p> <p>パスワード <input type="password"/></p> <p><input type="button" value="ログイン"/></p> <p>接続先URL</p> <p>期限に注意</p> <p>仮パスワード (2回目以降は本パスワード)</p>

公開鍵の登録 (2/3)

(2回目以降のアクセスでは出てこない)

操作説明	画面						
利用規定への同意（初回ログイン時のみ） <ul style="list-style-type: none">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	<div><h3>利用規定への同意</h3><p>下記を参照し、同意をお願いします。同意いただけない場合ABCIをご利用できませんので、ご理解をお願いいたします。</p><p>産総研外の利用者の方 産総研内の利用者の方</p><p><input checked="" type="radio"/> 同意する <input type="radio"/> 同意しない</p><p>以下のセキュリティ上の遵守項目を読み、同意いただけたら「全てを同意して次へ進む」で次に進んでください。</p><div><p>利用責任者、利用管理者、利用者（以後、利用者等という）は、研究所から提供されるABCI利用に関するアカウント及びアカウントのパスワードを研究所の承諾なく第三者に開示してはならず、かつ、第三者に推測されないように適切に設定し、管理しなければなりません。</p><p>利用者等は、利用者等のデータ等がいかなる法令にも違反していないことを表明及び保証し、利用者等のデータ等の開発、内容、運用、維持及び利用につき、責任を負います。（利用者等のデータ等のセキュリティ及びバックアップ）</p><p>利用者等は、ABCIを適正に利用し、利用者等のデータ等について、セキュリティを確保し保護すること、及び定期的に保存することを含め、適切なセキュリティ及び保護を行うことを誓約します。（安全保障輸出管理関係法令の遵守）</p><p>利用者等は、次の各号に該当する行為を行ってはなりません。</p><ol style="list-style-type: none">一 約款・規約及び回答書に記載されている事項に違反する行為二 申請書に記載した利用目的以外にABCIを利用する行為三 研究所若しくは第三者の著作権・商標権等の知的財産権を侵害する行為又はそのおそれがある行為四 研究所若しくは第三者の財産、プライバシー若しくは肖像権を侵害する行為又はそのおそれがある行為五 ABCIポイントを含めた研究所の電子情報を改ざん又は消去する行為</div></div>						
パスワード変更（初回ログイン時のみ） <ul style="list-style-type: none">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。 <p>※ 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。</p>	<div><h3>パスワード変更</h3><p>以下のフォームにパスワードをご入力の上、「変更」ボタンをクリックしてください。</p><table><tr><td>現在のパスワード</td><td><input type="password"/></td></tr><tr><td>新しいパスワード</td><td><input type="password"/></td></tr><tr><td>新しいパスワード（確認）</td><td><input type="password"/></td></tr></table><p><input type="button" value="戻る"/> <input type="button" value="変更"/></p><div><h4>パスワード規約</h4><ul style="list-style-type: none">15以上の文字をランダムに並べた文字列を指定してください。例えばLinux の辞書に登録されている単語は使用できません。文字をランダムに選ぶ方法として、パスワード作成用のソフトウェアを用いるなどして、自動的に生成することを推奨します。現在のパスワードと異なる文字列を指定してください。英小文字、英大文字、数字、特殊文字の4種類全てを含む文字列を指定してください。使用可能な特殊文字は以下の通りです。 空白、!、"、#、\$、%、&、'、(、)、*、+、,、-、.、/、:、;、<、=、>、?、@、[、\、]、^、_、`、{、 、}、~全角文字は使用できません。</div></div>	現在のパスワード	<input type="password"/>	新しいパスワード	<input type="password"/>	新しいパスワード（確認）	<input type="password"/>
現在のパスワード	<input type="password"/>						
新しいパスワード	<input type="password"/>						
新しいパスワード（確認）	<input type="password"/>						

ログイン(ssh)

1) ターミナルからアクセスサーバ(*as.abci.ai*)にログインします。

```
yourpc$ ssh -L 10022:es:22 -l aaa12345xx 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. → 初回ログイン時のみ表示される。
Are you sure you want to continue connecting (yes/no)? ← **yesを入力**
Warning: Permanently added 'XX.XX.XX.XX' (RSA) to the list of known hosts.
Enter passphrase for key '/home/username/.ssh/id_rsa': ← **パスフレーズ入力**
Welcome to ABCI access server. Please press any key if you disconnect this session.

Warning

上記状態で何らかのキーを入力するとSSH接続が切断されてしまいますので注意してください。

2) 別のターミナルで、インタラクティブノード(es)にポートフォワーディングします。

```
yourpc$ ssh -p 10022 -l aaa12345xx localhost
```

The authenticity of host 'localhost (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. → 初回ログイン時のみ表示される。
Are you sure you want to continue connecting (yes/no)? yes ← **yesを入力**
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Enter passphrase for key '/home/username/.ssh/id_rsa': ← **パスフレーズ入力**
[username@es1 ~]\$

```
ssh -L 10022:es:22 as.abci.ai  
ssh -p 10022 localhost
```

localhostのポート番号:remote host:remote hostのポート番号
localhostのポート番号

ファイルのアップロード・ダウンロード(ssh)

1) ファイルのアップロード: インタラクティブノード(es)にログインし、別のターミナルからscpを実行。

```
yourpc$ scp -P 10022 local-file aaa12345xx@localhost:remote-file  
# local-file: PCからアップしたいファイル、remote-file: リモートファイル名  
Enter passphrase for key: ++++++++ ← パスフレーズを入力  
local-file                                100%      file-size  transfer-speed      transfer-time
```

2) ファイルのダウンロード: インタラクティブノード(es)にログインし、別のターミナルからscpを実行。

```
yourpc$ scp -P 10022 aaa12345xx@localhost:sample ./  
# sample: PCへダウンロードしたいファイル  
Enter passphrase for key: ++++++++ ← パスフレーズを入力  
sample                                100%      file-size  transfer-speed      transfer-time
```

インタラクティブジョブ(qrsh)

1) インタラクティブノード(es)にログインして実行。

```
[username@es1 ~]$ qrsh -g gaa12345 -l rt_F=1 -l h_rt=01:00:00  
# gaa12345: グループ名, rt_F=1: 計算資源タイプ(フルノードを1個), h_rt=01:00:00(最大1時間確保)
```

2) インタラクティブジョブの状況を参照。

```
[username@es1 ~]$ qstat
```

job-ID	prior	name	user	state	submit/start at	queue	jclass	slots	ja-task-ID
151646	0.28027	<i>QRLOGIN</i>	<i>aaa12345xx</i>	r	01/21/2019 09:39:43	gpu@g0371		10	

バッチジョブ(qsub)

1) インタラクティブノード(es)にログインして実行。

```
[username@es1 ~]$ qsub -g gaa12345 -l rt_C.small=1 sample.sh  
# gaa12345: グループ名, rt_C.small=1 : 計算資源タイプ (CPU x 5コア), sample.sh: ジョブスクリプト  
Your job 151645 ("sample.sh") has been submitted
```

2) バッチジョブの状況を参照。

```
[username@es1 ~]$ qstat
```

job-ID	prior	name	user	state	submit/start at	queue	jclass	slots	ja-task-ID
151646	0.25586	<i>sample.sh</i>	<i>aaa12345xx</i>	r	01/20/2019 15:16:53	gpu@g0002		10	

3) バッチジョブの出力。

```
[username@es1 ~]$ ls -l
```

-rw-r--r--	1	<i>aaa12345xx gaa12345</i>	172	1月 20 15:17	sample.sh.e151646	<エラー出力ファイル>
-rw-r--r--	1	<i>aaa12345xx gaa12345</i>	0	1月 20 13:51	sample.sh.o151235	<正常出力ファイル>

Pythonで、TensorFlow(GPU)の環境を構築

1) 計算ノード(G.samll)にログインし、TensorFlow-gpuをインストール(一度、実施すればいい)。

CUDA9.0ではtensorflow-latestをサポートしていないので、tensorflow_gpu:1.12.0をインストール

```
[username@es3 ~]$ qssh -l rt_G.small -g gaa50091
[username@g0001 ~]$ module load python/3.6/3.6.5
[username@g0001 ~]$ python3 -m venv ~/Sample/v_tf_gpu
[username@g0001 ~]$ source ~/Sample/v_tf_gpu/bin/activate
(v_tf_gpu) [username@g0001 ~]$ which python
~/Sample/v_tf_gpu/bin/python
(v_tf_gpu) [username@g0001 ~]$ module load cuda/9.0/9.0.176.4 cudnn/7.4/7.4.2
(v_tf_gpu) [username@g0001 ~]$ pip3 install --ignore-installed --upgrade
https://storage.googleapis.com/tensorflow/linux/gpu/tensorflow_gpu-1.12.0-cp36-cp36m-linux_x86_64.whl
Collecting tensorflow-gpu==1.12.0 from https://storage.googleapis.com/tensorflow/linux/gpu/
...
Successfully installed absl-py-0.7.1 astor-0.7.1 gast-0.2.2 grpcio-1.19.0 h5py-2.9.0 keras-applications-1.0.7 keras-preprocessing-1.0.9
markdown-3.0.1 numpy-1.16.2 protobuf-3.7.0 setuptools-40.8.0 six-1.12.0 tensorboard-1.13.1 tensorflow-gpu-1.12.0 termcolor-1.1.0
werkzeug-0.15.1 wheel-0.33.1
(v_tf_gpu) [username@g0001 ~]$ deactivate
```

2) PythonでTensorFlowを実行。

```
[username@g0001 ~]$ python3 -m venv ~/Sample/v_tf_gpu
[username@g0001 ~]$ source ~/Sample/v_tf_gpu/bin/activate
(v_tf_gpu) [username@g0001 ~]$ ~> 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
>>> import tensorflow as tf
>>>
```

TensorFlow (GPU)が利用できるようになる

qsubでの実行例

1) スクリプト例(run_mnist.sh)

```
#!/bin/bash
# Environment Modules の初期化
source /etc/profile.d/modules.sh
# モジュールのロード
module load python/3.6/3.6.5
module load cuda/9.0/9.0.176.4
module load cudnn/7.4/7.4.2
# venv 環境の有効化
source ~/Sample/v_tf_gpu/bin/activate
# 計算を実行(スクリプト)
cd ~/Sample/v_tf_gpu
python sample_mnist.py
```

4) インタラクティブノードでのコマンド

```
# スクリプト、プログラムに実行権を付与
[username@es3 v_tf_gpu]$ chmod u+x run_mnist.sh
[username @es3 v_tf_gpu]$ chmod u+x sample_mnist.py
[username @es3 v_tf_gpu]$ ls -l
-rwxr----- 1 aaa12345xx aaa12345xx 417 3月 27 11:13 run_mnist.sh
-rwxr----- 1 aaa12345xx aaa12345xx 720 3月 27 10:05 sample_mnist.py
# -rwxr- となっていることを確認

[username @es3 v_tf_gpu]$ qsub -g gaa12345 -l rt_G.small=1 run_mnist.sh
```

2) プログラム例(sample_mnist.py)

```
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

def create_model():
    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)
    ])
    return model
model = create_model()
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=3)
model.save("./mnist_model.hdf5")
print(model.evaluate(x_test, y_test))
print("finished")
```

3) スクリプト、プログラムをscpでアップロード

```
yourpc$ scp -P 10022 run_mnist.sh aaa12345xx@localhost:run_mnist.sh
yourpc$ scp -P 10022 sample_mnist.py aaa12345xx@localhost:sample_mnist.py

# ~/Sample/v_tf_gpuフォルダへ移動
```

Singularityを利用して、TensorFlow(GPU)の環境を構築

1) インタラクティブノード(es)にログインし、TensorFlowのDockerイメージを取得(一度、実施すればいい)。

以下のイメージを使用 (tag: -gpu-py3)

<https://hub.docker.com/r/tensorflow/tensorflow/>

```
[username@es3 ~]$ module load singularity/2.6.1
[username@es3 ~]$ singularity pull docker://tensorflow/tensorflow:1.12.0-gpu-py3
Docker image path: index.docker.io/tensorflow/tensorflow:1.12.0-gpu-py3
Cache folder set to /fs3/home/axa01001hf/.singularity/docker
[17/17] |=====| 100.0%
:
Done. Container is at: ./tensorflow-1.12.0-gpu-py3.simg
[username@es3 ~]$ ls
tensorflow-1.12.0-gpu-py3.simg
```

2) インタラクティブジョブ(計算ノード)で、Singularityを実行し、TensorFlowの環境を構築。

```
[username@es3 ~]$ qsub -l rt_F=1 -l h_rt=01:00:00 -g gaa12345
[username@g0003 ~]$ module load singularity/2.6.1
[username@g0003 ~]$ singularity shell --nv ./tensorflow-1.12.0-gpu-py3.simg
Singularity: Invoking an interactive shell within container...

Singularity tensorflow-1.12.0-gpu-py3.simg:~> python
>>> import tensorflow as tf
>>>

# TensorFlow (GPU)が利用できるようになる
```

TensorFlowの実行例

```
hagishima — aaa10005yk@g0195:~ — ssh -p 10022 -l aaa10005yk localhost — 193x70
[aaa10005yk@es3 ~]$ qssh -l rt_F=1 -l h_rt=01:00:00 -g gaa50069
[aaa10005yk@g0195 ~]$ module load singularity/2.6.1
[aaa10005yk@g0195 ~]$ singularity shell --nv ./tensorflow-latest-gpu-py3.simg
Singularity: Invoking an interactive shell within container...

Singularity tensorflow-latest-gpu-py3.simg:~> 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
2019-01-31 05:41:33.601800: 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
2019-01-31 05:41:34.114172: 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
2019-01-31 05:41:34.450015: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 1 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
2019-01-31 05:41:34.789030: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 2 with properties:
name: Tesla V100-SXM2-16GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0000:b1:00.0
totalMemory: 15.78GiB freeMemory: 15.37GiB
2019-01-31 05:41:35.138469: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1432] Found device 3 with properties:
name: Tesla V100-SXM2-16GB major: 7 minor: 0 memoryClockRate(GHz): 1.53
pciBusID: 0000:b2:00.0
totalMemory: 15.78GiB freeMemory: 15.37GiB
2019-01-31 05:41:35.138560: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1511] Adding visible gpu devices: 0, 1, 2, 3
2019-01-31 05:41:38.702250: I tensorflow/core/common_runtime/gpu/gpu_device.cc:982] Device interconnect StreamExecutor with strength 1 edge matrix:
2019-01-31 05:41:38.702304: I tensorflow/core/common_runtime/gpu/gpu_device.cc:988]      0 1 2 3
2019-01-31 05:41:38.702314: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 0:  N Y Y Y
2019-01-31 05:41:38.702321: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 1:  Y N Y Y
2019-01-31 05:41:38.702328: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 2:  Y Y N Y
2019-01-31 05:41:38.702334: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1001] 3:  Y Y Y N
2019-01-31 05:41:38.703068: 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 GP
U (device: 0, name: Tesla V100-SXM2-16GB, pci bus id: 0000:3d:00.0, compute capability: 7.0)
2019-01-31 05:41:38.704461: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:1 with 14874 MB memory) -> physical GP
U (device: 1, name: Tesla V100-SXM2-16GB, pci bus id: 0000:3e:00.0, compute capability: 7.0)
2019-01-31 05:41:38.704741: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:2 with 14874 MB memory) -> physical GP
U (device: 2, name: Tesla V100-SXM2-16GB, pci bus id: 0000:b1:00.0, compute capability: 7.0)
2019-01-31 05:41:38.705017: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1115] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:3 with 14874 MB memory) -> physical GP
U (device: 3, name: Tesla V100-SXM2-16GB, pci bus id: 0000:b2:00.0, compute capability: 7.0)
60000/60000 [=====] - 10s 171us/step - loss: 0.2208 - acc: 0.9353
Epoch 2/5
60000/60000 [=====] - 4s 65us/step - loss: 0.0966 - acc: 0.9708
Epoch 3/5
60000/60000 [=====] - 4s 65us/step - loss: 0.0679 - acc: 0.9785
Epoch 4/5
60000/60000 [=====] - 4s 66us/step - loss: 0.0529 - acc: 0.9826
Epoch 5/5
60000/60000 [=====] - 4s 66us/step - loss: 0.0418 - acc: 0.9863
<tensorflow.python.keras.callbacks.History object at 0x2b05720acef0>
>>> model.evaluate(x_test, y_test)
10000/10000 [=====] - 0s 33us/step
[0.06713192030405335, 0.9799]
>>> □
```

参照 : [_index.ipynb \(Google\)](#)
<https://www.tensorflow.org/tutorials/?hl=ja>

Jupyter Notebookの利用

1) Jupyter Notebookのインストール(一度、実施すればいい)。

```
[username@es3 ~]$ module load python/3.6/3.6.5
[username@es3 ~]$ python3 -m venv ~/lib/pyenv/jupyter_test
[username@es3 ~]$ source ~/lib/pyenv/jupyter_test/bin/activate
(jupyter_test) es4 $ pip install --upgrade pip
(jupyter_test) es4 $ pip install jupyter
(jupyter_test) es4 $ deactivate
```

2) インタラクティブジョブ(qrsh)で、Jupyter Notebookを起動する。

```
[username@es3 ~]$ qrsh -g gaa12345 -l rt_F=1 -l h_rt=01:00:00 <gaa12345: グループ名>
[aaa12345xx @g0019 ~]$ module load python/3.6/3.6.5
# aaa12345xx: username, g0019: 割当てられた計算ノードリソース
[aaa12345xx @g0019 ~]$ source ~/lib/pyenv/jupyter_test/bin/activate
(jupyter_test) [aaa12345xx @g0019 ~]$ jupyter notebook --no-browser --ip=`hostname` >> jupyter.log 2>&1 &
(jupyter_test) [aaa12345xx @g0019 ~]$ jupyter notebook list
Currently running servers:
http://g0004.abci.local:8888/?token=e7f0ba979d4ffd9eeb7e6debf5a326f853fc289583f92dc5 :: /fs3/home/aaa12345xx
```

3) 別ターミナルで。

```
yourpc $ ssh -L 10022:es:22 -l aaa12345xx as.abci.ai
```

4) さらに、別のターミナルで。

```
yourpc $ ssh -L 18888:g0004:8888 -l aaa12345xx (-i ~/.ssh/id_rsa) -p 10022 localhost
# -i: 秘密鍵オプションは省略可
```

5) ブラウザでアクセス(トークンは、2)をコピー)。

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


Jupyter Notebook画面例

The screenshot displays the Jupyter Notebook web interface in a browser. The address bar shows `localhost:18888/tree#notebooks`. The interface includes a top navigation bar with the Jupyter logo, a "Quit" button, and a "Logout" button. Below this is a tabbed interface with "Files", "Running", and "Clusters" tabs. The "Files" tab is active, showing a file browser view. At the top of the file browser, there are buttons for "Upload", "New", and a refresh icon. Below these buttons is a table of files and folders. The table has columns for "Name", "Last Modified", and "File size". The files listed include folders like "lib", "Untitled Folder", "v_tf", and "work", and files like "Untitled.ipynb", "Untitled1.ipynb", "Untitled2.ipynb", "Untitled3.ipynb" (which is marked as "Running"), "ABCマニュアル.pptx", "jupyter.log", "sample", "sample.sh", and several "sample.sh" files with unique IDs. At the bottom of the interface, there is a breadcrumb navigation bar showing the path "sample.sh" > "sample" > "remote-dir". A button labeled "すべてを表示" (Show all) is located at the bottom right of the breadcrumb bar.

Name	Last Modified	File size
0		
lib	3日前	
Untitled Folder	5分前	
v_tf	5日前	
work	2ヶ月前	
~	2ヶ月前	
Untitled.ipynb	2ヶ月前	7.92 kB
Untitled1.ipynb	2ヶ月前	1.37 kB
Untitled2.ipynb	1日前	845 B
Untitled3.ipynb	Running 4分前	845 B
ABCマニュアル.pptx	3分前	66.3 kB
jupyter.log	2分前	20 kB
sample	21時間前	66 B
sample.sh	19時間前	37 B
sample.sh.e151646	19時間前	172 B
sample.sh.o151235	21時間前	0 B
sample.sh.o151244	20時間前	0 B

ABCI参考サイト

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/>