



DEEP
LEARNING
INSTITUTE

DRIVE PX2 と DRIVEWORKS による 自動運転システム開発

室河 徹 / 吉井 健一

NVIDIA

DRIVEWORKS を使用した HANDS-ON LAB

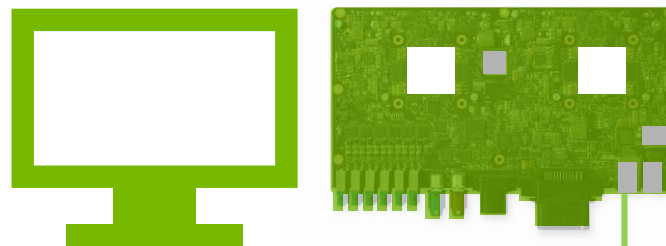
ラボ準備

ラボについて

開発用ホストマシン (Linux PC)



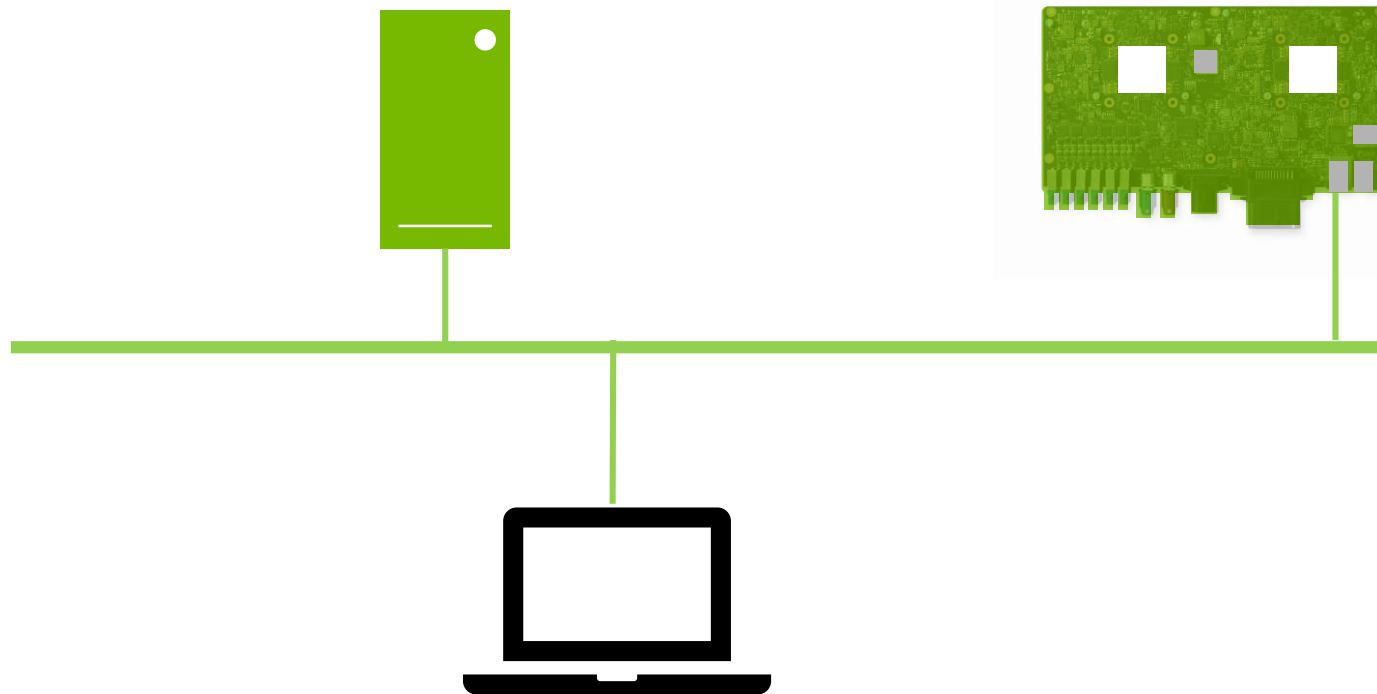
ターゲット (DRIVE PX2)



ラボについて

開発用ホストマシン (Linux PC)

ターゲット (DRIVE PX 2)



リモートターミナル

それでは始めましょう

ホストマシンへのログイン

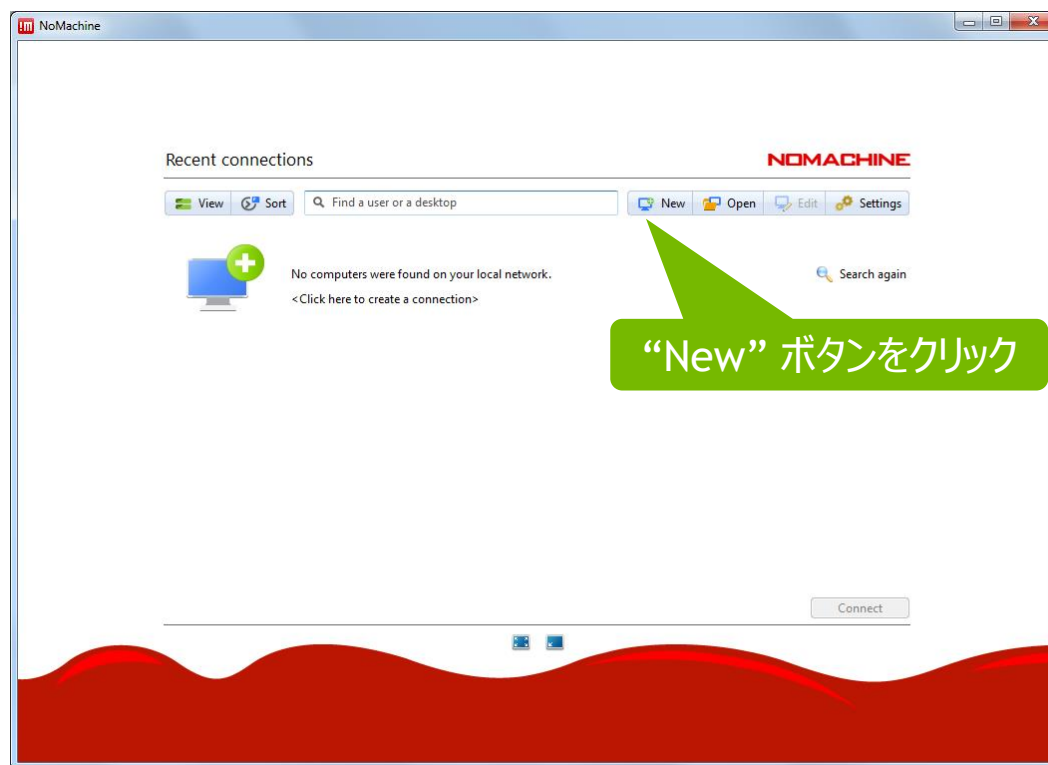
Quiklabs にログインして開発用**ホストマシン**への接続方法を確認

- NoMachine のダウンロード (未インストールの場合)
 - <https://www.nomachine.com/download>
- NoMachine のインスタンスを起動
- NoMachine より AWS サーバーへ接続
 - IP アドレスは qwiklabs 内に記されています。

これが開発用ホストマシンになります。

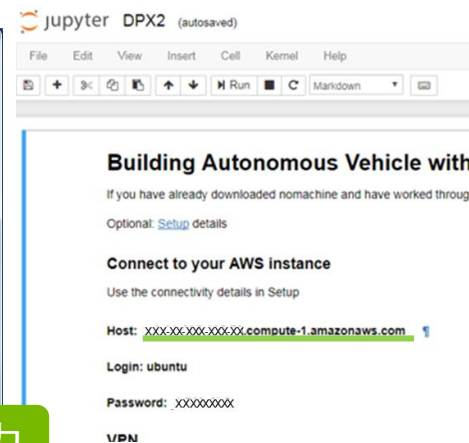
NOMACHINE 手順 1

開発ホストマシンへの接続手順



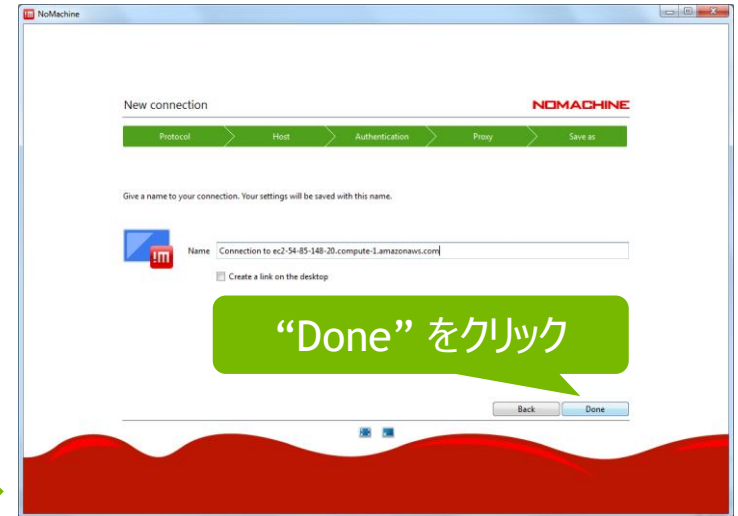
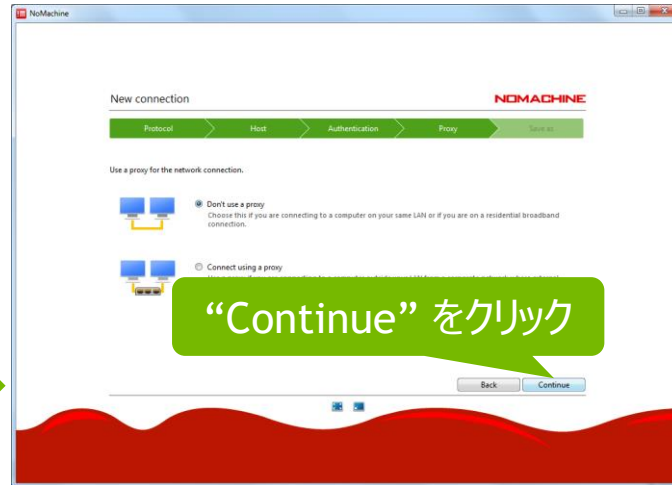
NOMACHINE 手順 2

開発ホストマシンへの接続手順



NOMACHINE 手順 3

開発ホストマシンへの接続手順



NOMACHINE 手順 4

開発ホストマシンへの接続手順

The image shows the NoMachine application window titled "NoMachine - Connection to ec2-54-85-148-20.compute-1.amazonaws.com". The interface displays the connection details and a login prompt. A green arrow points from the "Connect" button in the previous step to the login fields. The "Username" field contains "ubuntu" and the "Password" field contains a masked password. A green callout box points to the "OK" button at the bottom right.

Recent connections

Connection to ec2-54-85-148-20.compute-1.amazonaws.com

Connection to ec2-54-85-148-20.compute-1.amazonaws.com

“Connect” をクリック

Connection to ec2-54-85-148-20.compute-1.amazonaws.com

Please type your username and password to login.

Username: ubuntu

Password: *****

“ubuntu” と入力

Save this password in the registry

Qwiklabs に記載の Password を入力

“OK” をクリック

Back OK

jupyter DPX2 (autosaved)

File Edit View Insert Cell Kernel Help

Building Autonomous Vehicle with

If you have already downloaded nomachine and have worked through

Optional [Setup](#) details

Connect to your AWS instance

Use the connectivity details in Setup

Host: XXXXX:XXX:XXX:XXX.compute-1.amazonaws.com

Login: ubuntu

Password: XXXXXXXX

VDP

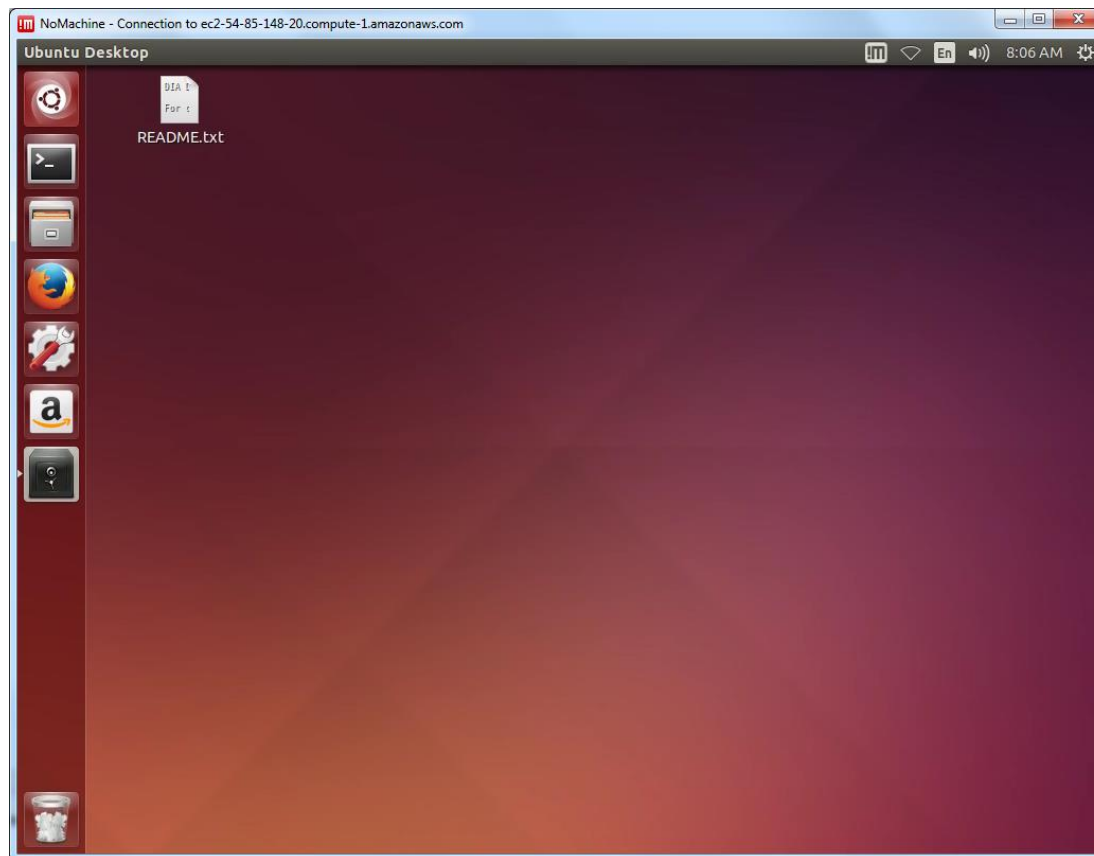
NOMACHINE 手順 5

開発ホストマシンへの接続手順



NOMACHINE

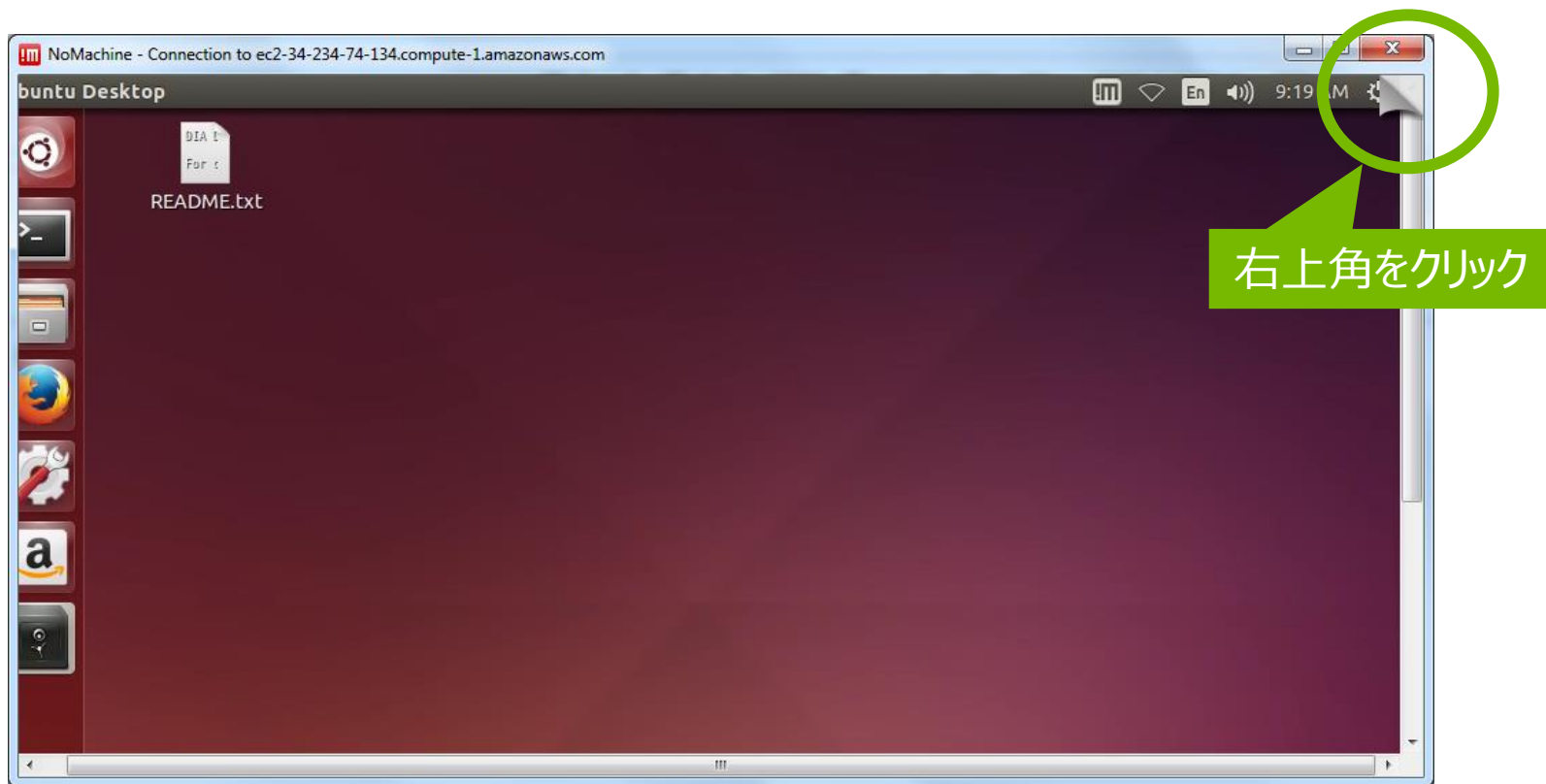
開発ホストマシンに接続完了



開発ホストマシンの
デスクトップ画面

NOMACHINE

スクリーンの調整が必要な場合



NOMACHINE

スクリーンの調整



それでは始めましょう

DRIVE PX2 へのログイン

机に置かれたトークンをつかって VPN 設定を行ってください。

ネットワーク上の DRIVE PX2 にログインします。

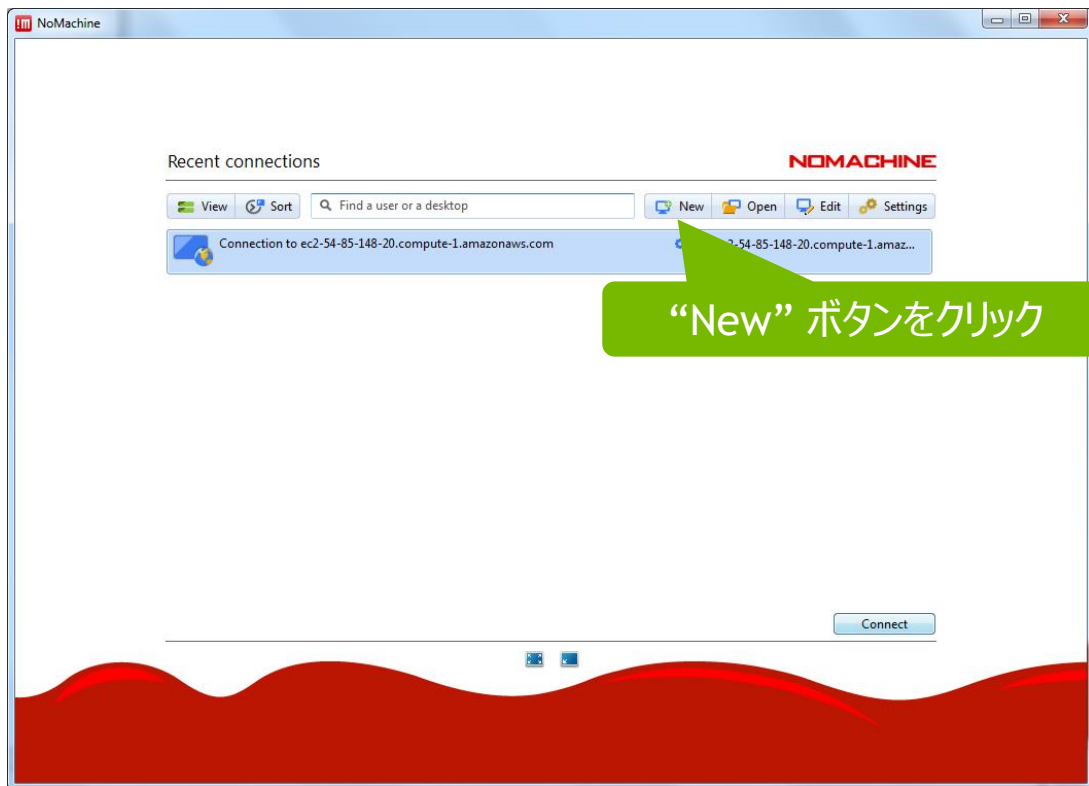
- NoMachine で新しいインスタンスを立ち上げます。
- 指定された IP アドレスをつかって DRIVE PX2 に接続します。

これが今日使用する DRIVE PX2 になります。

NOMACHINE 手順 6

DRIVE PX2 への接続手順

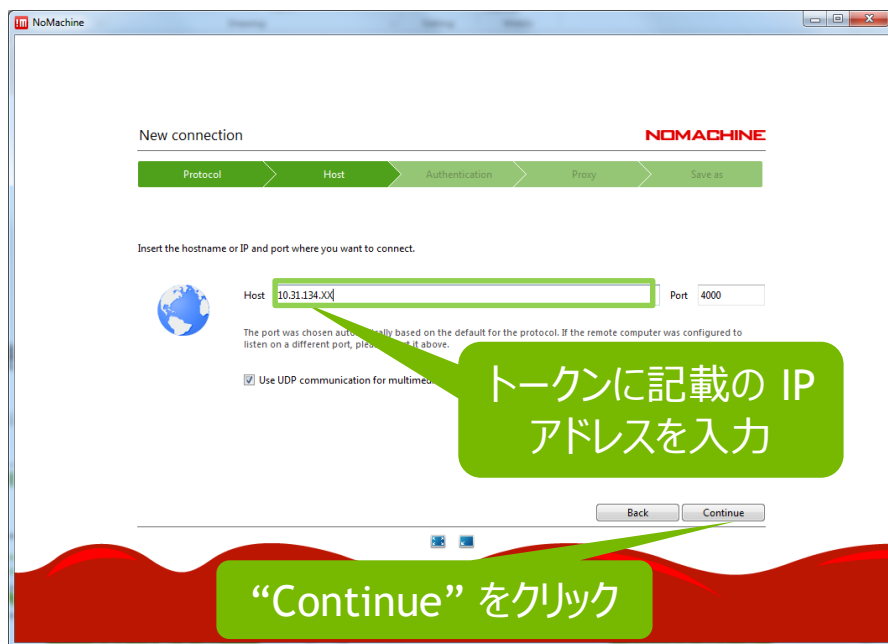
NoMachine をもう一つ立ち上げて
DRIVE PX2 用のインスタンスを作成する



NOMACHINE 手順 7

DRIVE PX2への接続手順

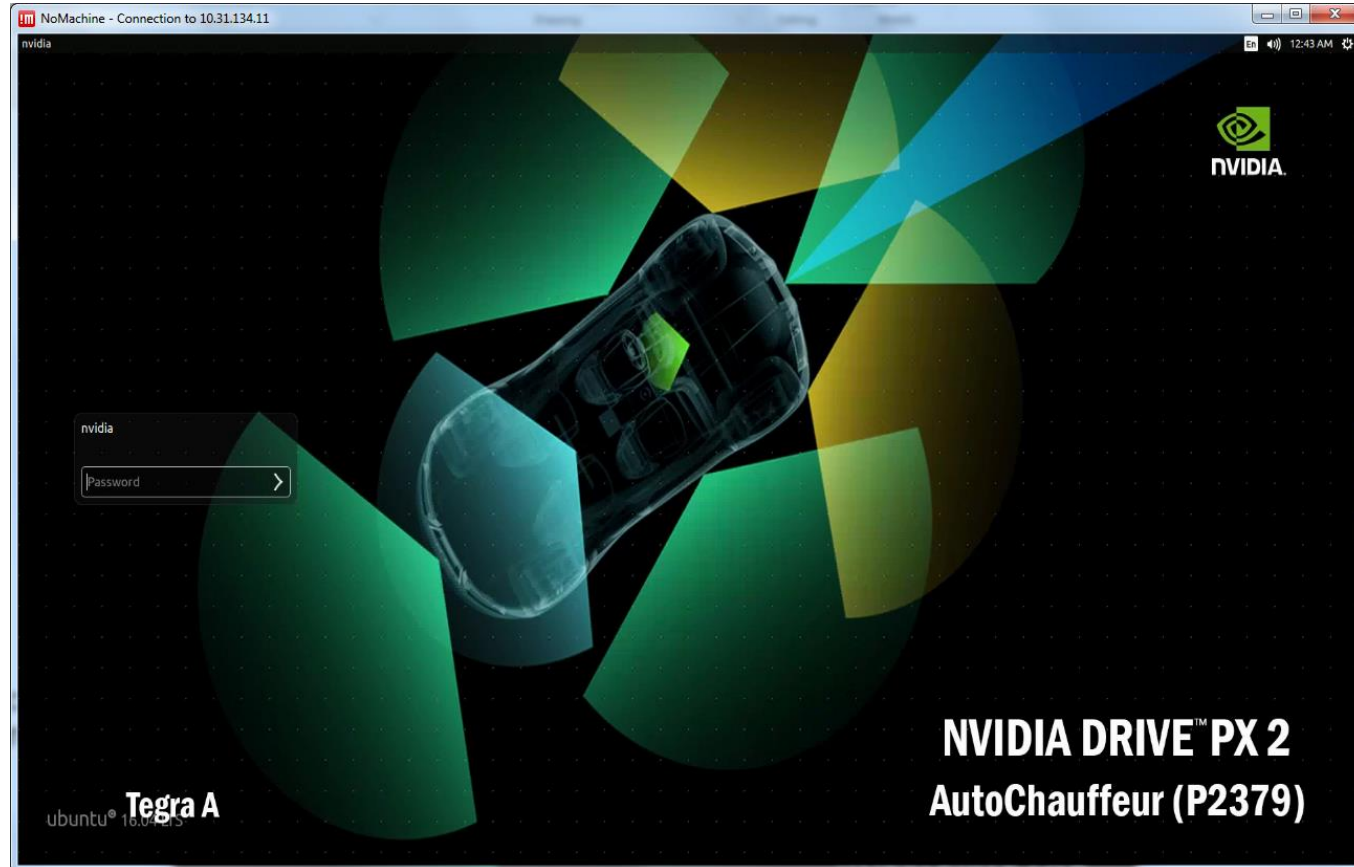
開発ホストマシンへの接続手順と同様のことを繰り返します。
DRIVE PX2 接続で異なるのは以下の部分です。



NOMACHINE

DRIVE PX2 に接続完了

Login: **nvidia**
Password: **nvidia**



DIRVE PX2 の
デスクトップ画面

ラボ実習

DRIVEWORKS ツアー

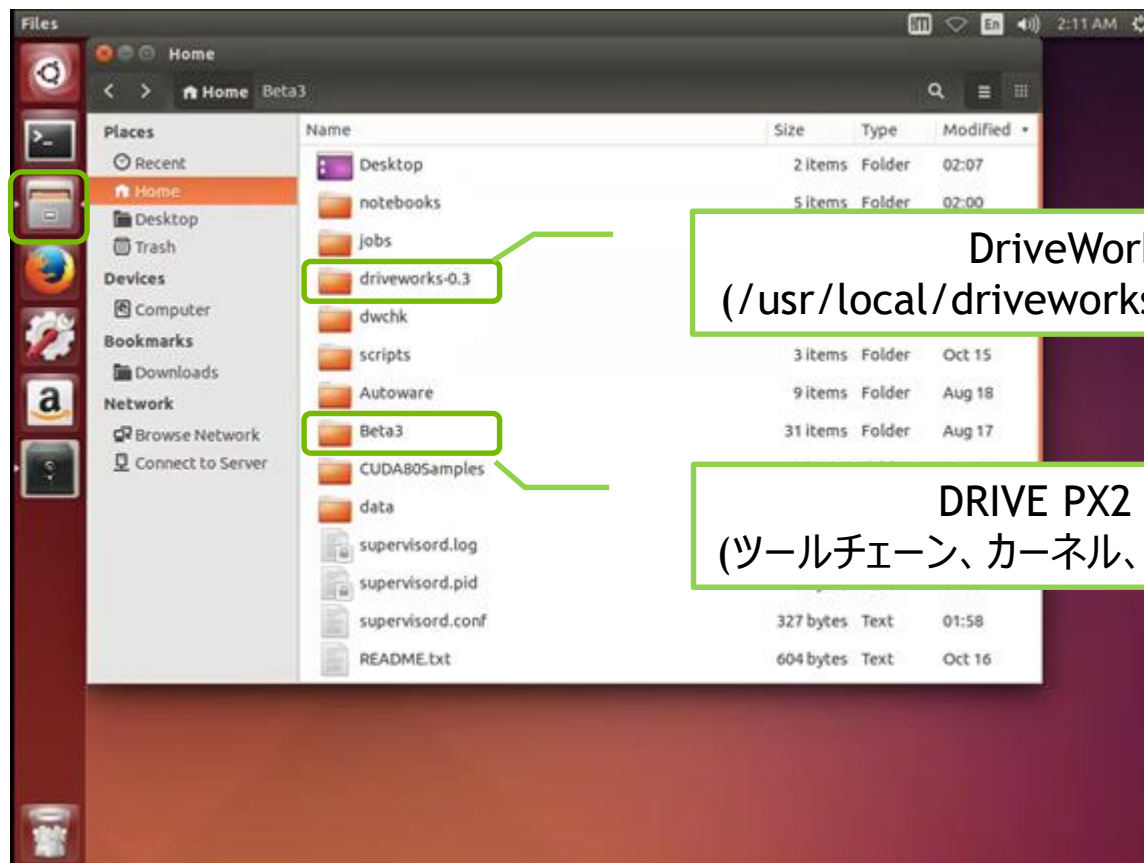
DRIVEWORKS ツアー（ホスト）

開発用ホストマシン

NoMachine でホストマシンを表示させてください。

DRIVEWORKS ツアー (ホスト)

Home (/home/ubuntu)



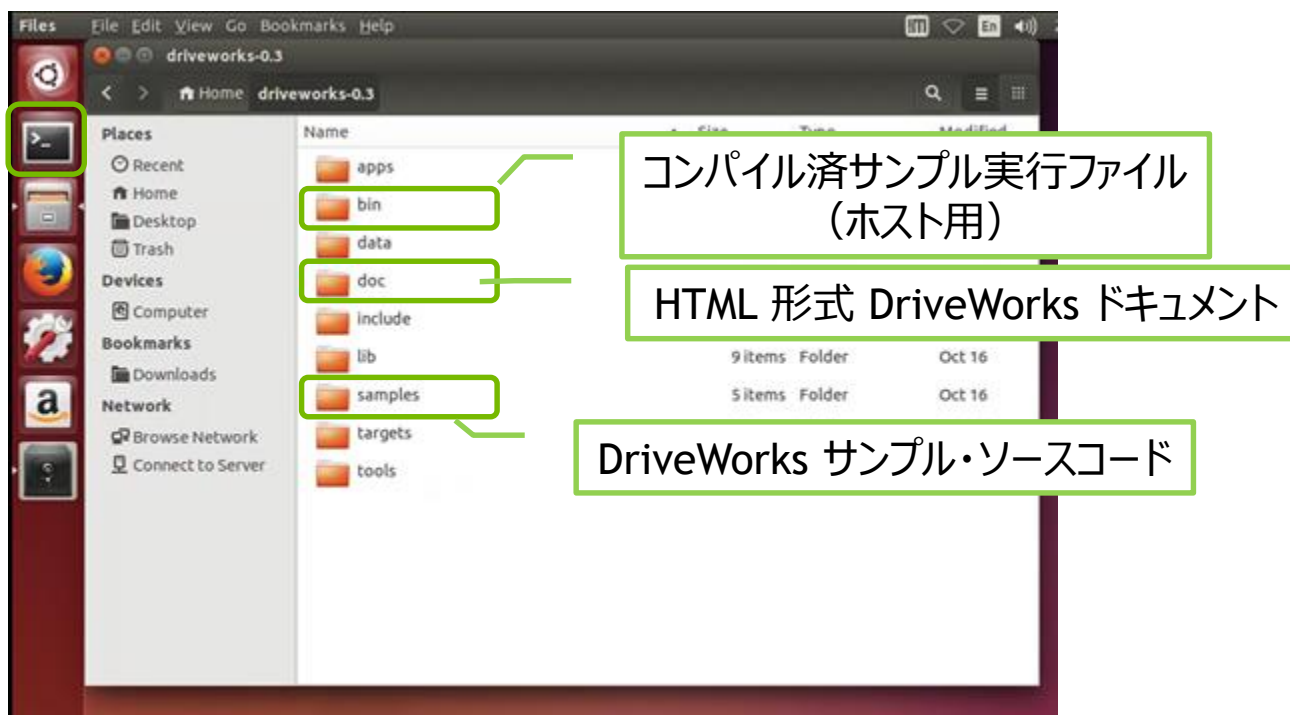
DriveWorks ディレクトリ
(/usr/local/driveworks-0.3 からコピーされたもの)

* 本日はこのディレクトリの中のファイルを使用します。

DRIVE PX2 用 SDK ディレクトリ
(ツールチェーン、カーネル、ファイルシステム、ツール、etc.)

DRIVEWORKS ツアー (ホスト)

DriveWorks ディレクトリ (~ /driveworks-0.3)



DRIVEWORKS ツアー (ホスト)

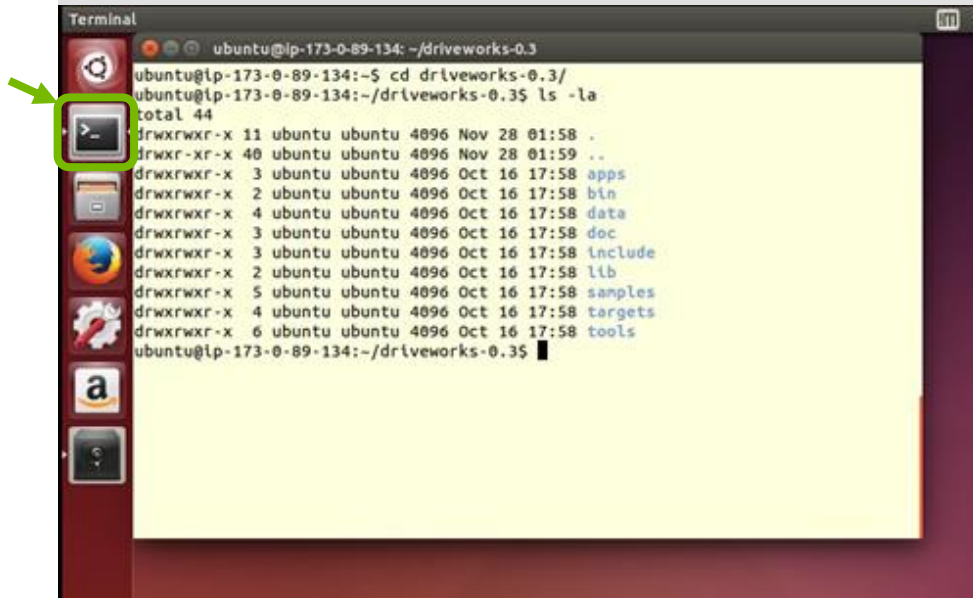
ターミナルで確認

DriveWorks ディレクトリに移動

```
$ cd driveworks-0.3
```

ディレクトリの中身を確認

```
$ ls -la
```



DRIVEWORKS ツアー (ホスト)

サンプルコードの確認

サンプルコードのディレクトリを確認

```
$ ls samples/src -la
```

```
ubuntu@ip-173-0-83-62: ~/driveworks-0.3
ubuntu@ip-173-0-83-62:~/driveworks-0.3$ ls samples/src -la
total 88
drwxrwxr-x 22 ubuntu ubuntu 4096 Oct 16 17:58 .
drwxrwxr-x  5 ubuntu ubuntu 4096 Oct 16 17:58 ..
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 colorcorrection
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 common
drwxrwxr-x  7 ubuntu ubuntu 4096 Oct 16 17:58 dnn
drwxrwxr-x  6 ubuntu ubuntu 4096 Oct 16 17:58 drivenet
drwxrwxr-x  3 ubuntu ubuntu 4096 Oct 16 17:58 egomotion
drwxrwxr-x  3 ubuntu ubuntu 4096 Oct 16 17:58 features
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 freespace
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 hello_world
drwxrwxr-x  4 ubuntu ubuntu 4096 Oct 16 17:58 image
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 ipc
drwxrwxr-x  4 ubuntu ubuntu 4096 Oct 16 17:58 laneDetection
drwxrwxr-x  3 ubuntu ubuntu 4096 Oct 16 17:58 mapping
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 maps
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 raw
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 rectifier
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 renderer
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 rigconfiguration
drwxrwxr-x 17 ubuntu ubuntu 4096 Oct 16 17:58 sensors
drwxrwxr-x  2 ubuntu ubuntu 4096 Oct 16 17:58 sfm
drwxrwxr-x  5 ubuntu ubuntu 4096 Oct 16 17:58 stereo
ubuntu@ip-173-0-83-62:~/driveworks-0.3$
```

DRIVEWORKS ツアー (DRIVE PX2)

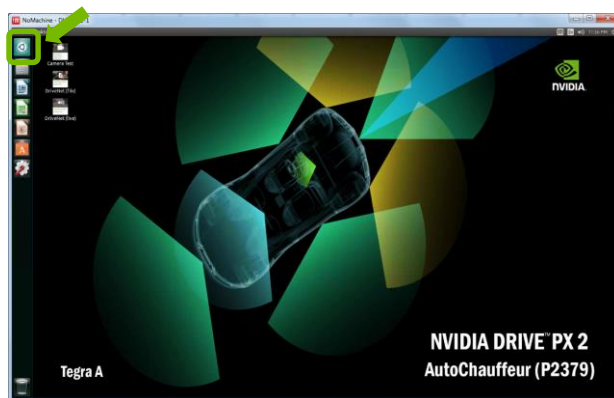
DRIVE PX2

NoMachine で DRIVE PX2 を表示させてください。

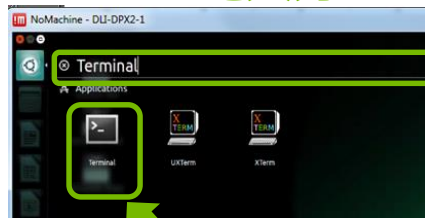
DRIVEWORKS ツアー (DRIVE PX2)

ターミナルの起動

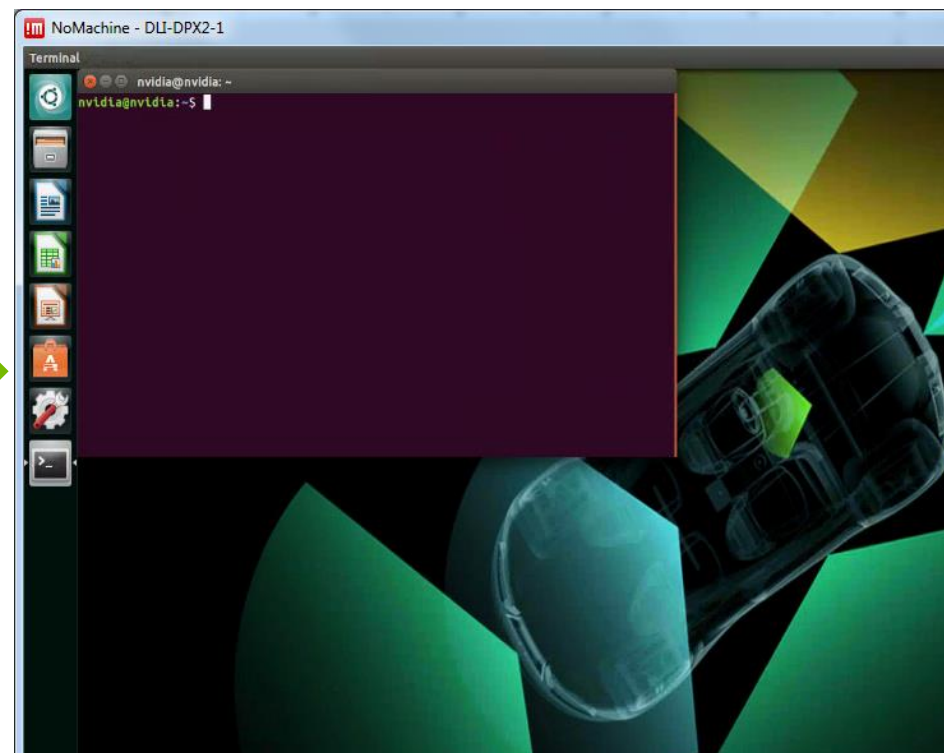
DASH アイコンをクリック



Terminal と入力



クリック



DRIVEWORKS ツアー (DRIVE PX2)

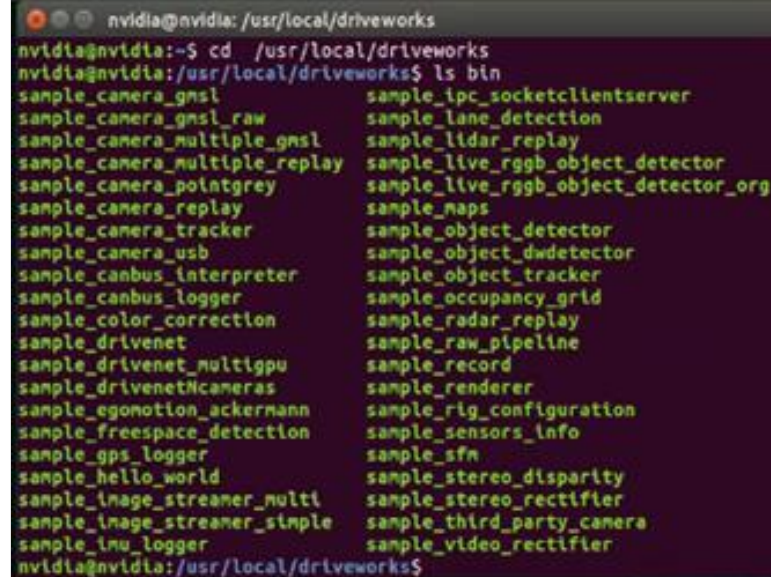
サンプルコードの確認

DriveWork ディレクトリに移動

```
$ cd /usr/local/driveworks
```

bin ディレクトリにあるサンプルコードを確認

```
$ ls bin
```



```
nvidia@nvidia: /usr/local/driveworks
nvidia@nvidia:~$ cd /usr/local/driveworks
nvidia@nvidia:/usr/local/driveworks$ ls bin
sample_camera_gnsl          sample_ipc_socketclientserver
sample_camera_gnsl_raw      sample_lane_detection
sample_camera_multiple_gnsl sample_lidar_replay
sample_camera_multiple_replay sample_live_rggb_object_detector
sample_camera_pointgrey     sample_live_rggb_object_detector_org
sample_camera_replay        sample_maps
sample_camera_tracker       sample_object_detector
sample_camera_usb           sample_object_dwdetector
sample_canbus_interpreter   sample_object_tracker
sample_canbus_logger        sample_occupancy_grid
sample_color_correction     sample_radar_replay
sample_drivenet             sample_raw_pipeline
sample_drivenet_multigpu    sample_record
sample_drivenet11cameras    sample_renderer
sample_egonotion_ackermann  sample_rig_configuration
sample_freespace_detection  sample_sensors_info
sample_gps_logger           sample_sfn
sample_hello_world          sample_stereo_disparity
sample_image_streamer_multi sample_stereo_rectifier
sample_image_streamer_single sample_third_party_camera
sample_inu_logger           sample_video_rectifier
nvidia@nvidia:/usr/local/driveworks$
```

DRIVEWORKS のサンプルを実行してみましょう

DRIVEWORKS サンプル実行 (DRIVE PX2)

DRIVE PX2

NoMachine で DRIVE PX2 を表示させてください。

サンプルへのパラメータの確認 (DRIVE PX2)

help オプションの指定

DRIVE PX2 上の実行ファイルがあるディレクトリに移動してください。

```
$ cd /usr/local/driveworks/bin/
```

いずれかのサンプルを「--help」オプションつきで実行します。
例)

```
$ ./sample_camera_gmsl --help
```

パラメータ例)

- camera-type: カメラタイプの指定
- csi-port: カメラを接続しているポートの指定
- write-file: 動画を出力するファイル名の指定

```
nvidia@nvidia: /usr/local/driveworks/bin$ ls
sample_camera_gmsl          sample_ipc_socketclientserver
sample_camera_gmsl_raw      sample_lane_detection
sample_camera_multiple_gmsl  sample_lidar_replay
sample_camera_multiple_replay sample_live_rggb_object_detector
sample_camera_pointgrey     sample_live_rggb_object_detector_org
sample_camera_replay         sample_maps
sample_camera_tracker        sample_object_detector
sample_camera_usb            sample_object_dwdetector
sample_canbus_interpreter    sample_object_tracker
sample_canbus_logger          sample_occupancy_grid
sample_color_correction       sample_radar_replay
sample_drivenet              sample_raw_pipeline
sample_drivenet_multigpu      sample_record
sample_drivenetNcameras       sample_renderer
sample_egomotion_ackermann    sample_rig_configuration
sample_freespace_detection    sample_sensors_info
sample_gps_logger             sample_sfn
sample_hello_world            sample_stereo_disparity
sample_image_streamer_multi   sample_stereo_rectifier
sample_image_streamer_simple  sample_third_party_camera
sample_imu_logger             sample_video_rectifier
nvidia@nvidia: /usr/local/driveworks/bin$ ./sample_camera_gmsl --help
Options:
--camera-type=ar0231-rcbb
--csi-port=ab
--fifo-size=3
--offscreen=0
--serialize-bitrate=8000000
--serialize-framerate=30
--serialize-type=h264
--slave=0
--write-file=
nvidia@nvidia: /usr/local/driveworks/bin$
```


DRIVENET サンプルの実行 (DRIVE PX2)

マルチクラスの物体検出

コンパイル済みのマルチクラス物体検出サンプルを実行します。

DRIVE PX2 上の実行ファイルがあるディレクトリに移動してください。

```
$ cd /usr/local/driveworks/bin/
```

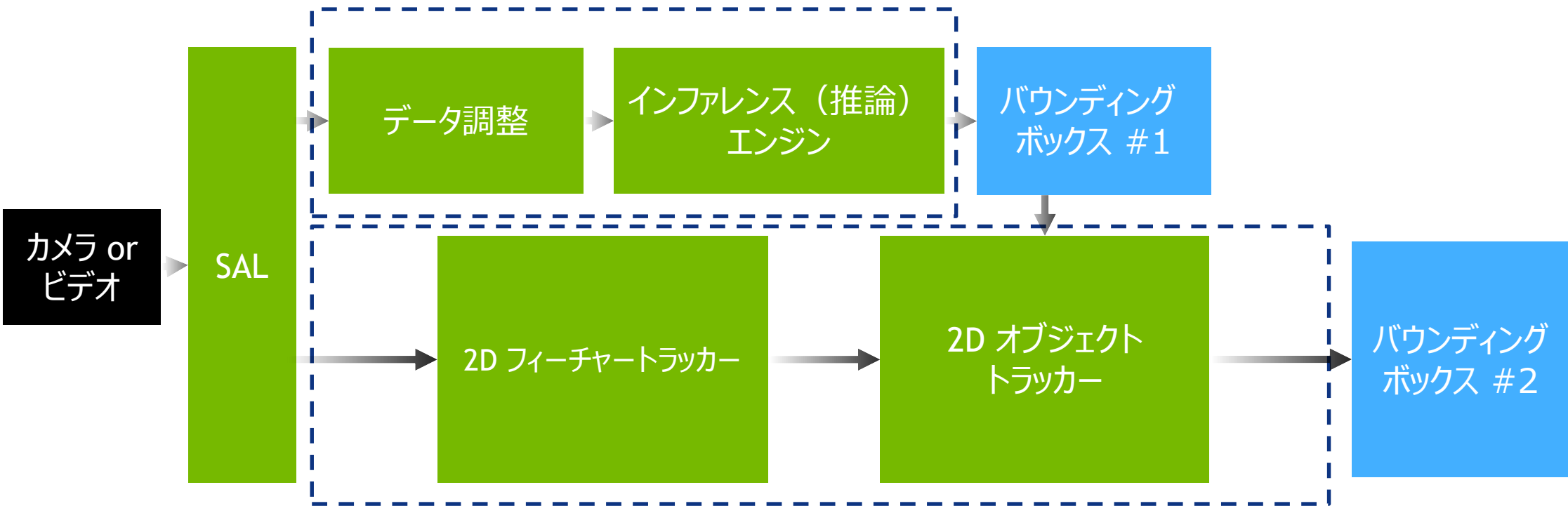
DriveNet のサンプルを実行します。

```
$ ./sample_drivenet
```



DRIVENET

処理フロー



LANENET サンプルの実行 (DRIVE PX2)

レーンの検出

コンパイル済みのレーン検出サンプルを実行します。

DRIVE PX2 上の実行ファイルがあるディレクトリに移動してください。

```
$ cd /usr/local/driveworks/bin/
```

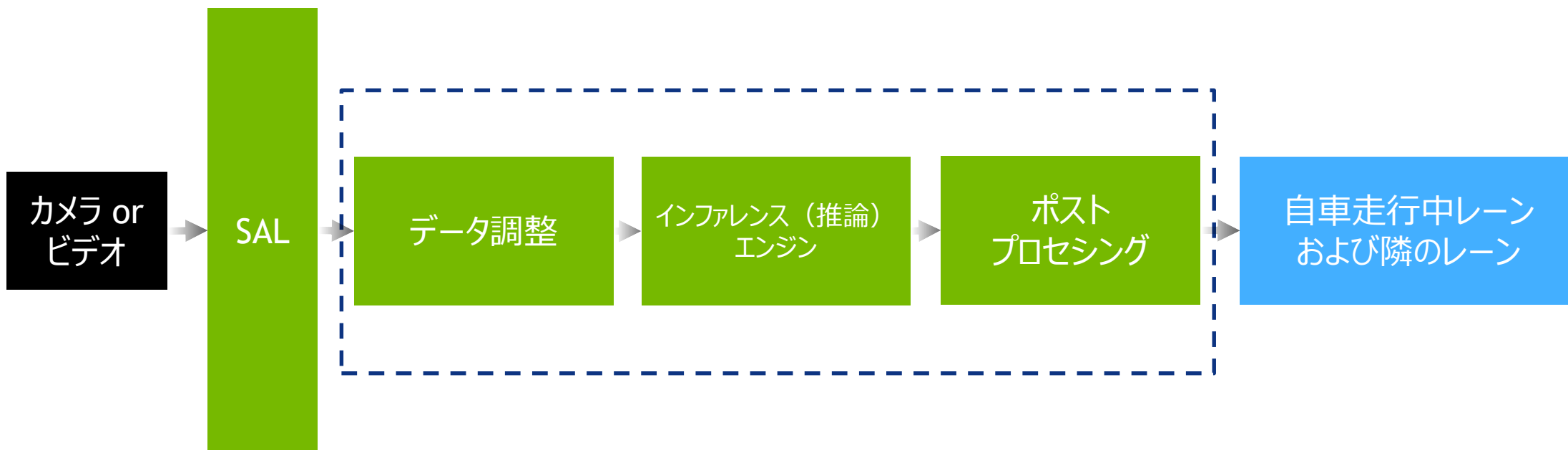
レーン検出のサンプルを実行します。

```
$ ./sample_lane_detection
```



LANENET

処理フロー



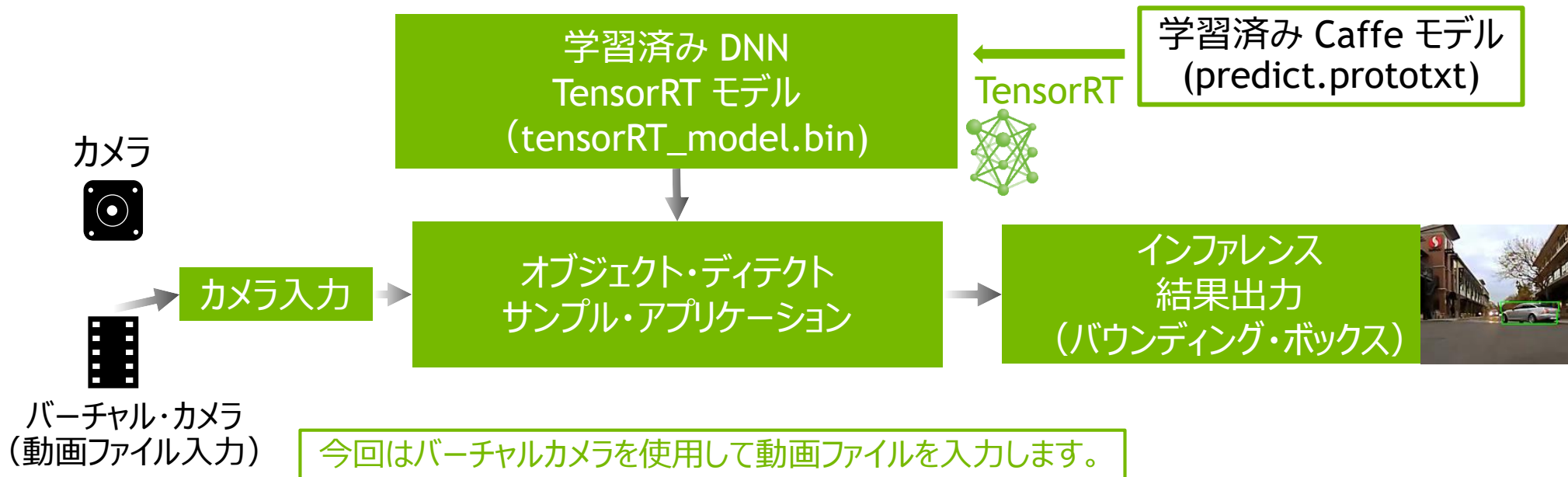
DRIVEWORKS API ハンズオン・ラボ

DRIVEWORKS API ハンズオン・ラボ

概要

学習済みのディープ・ニューラル・ネット（DNN）を使用

カメラ入力から車両認識しバウンディング・ボックスを表示させるサンプルを完成させます。



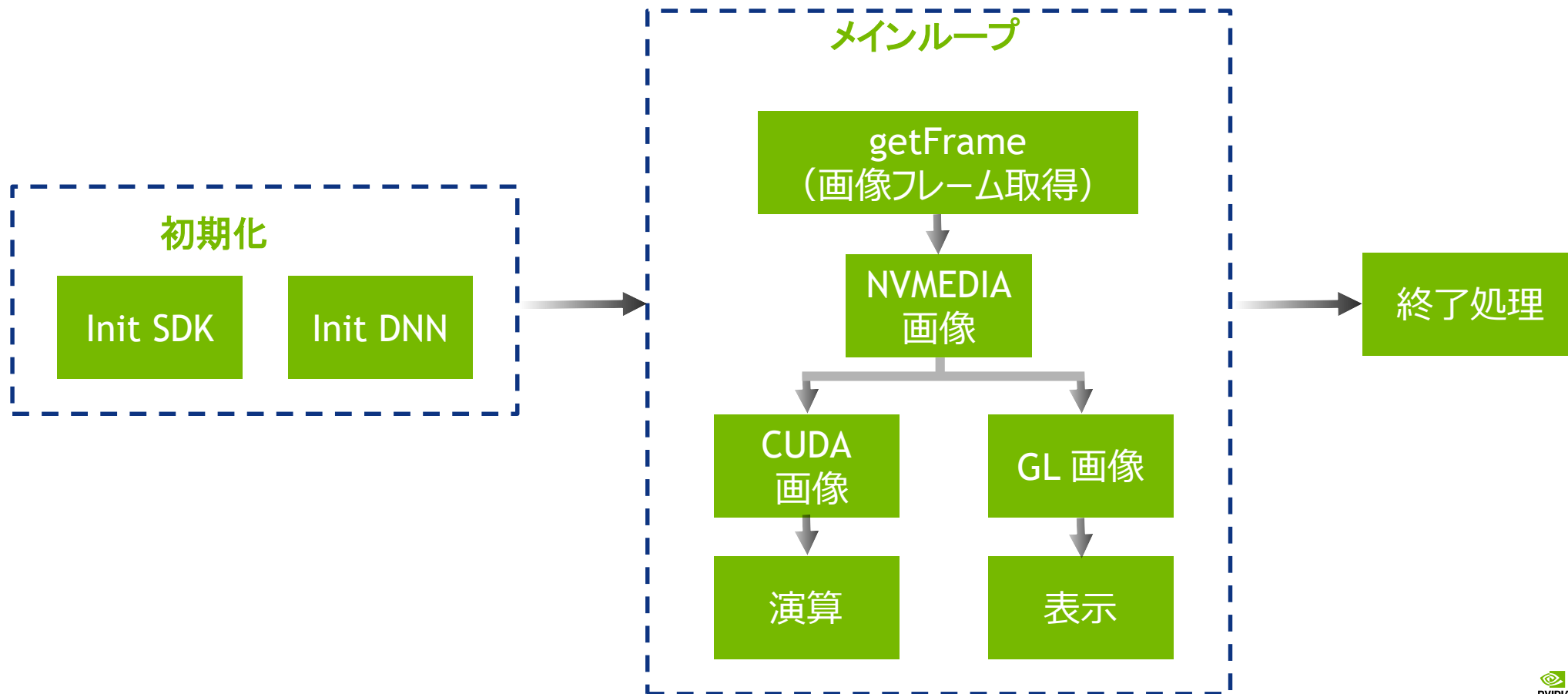
DRIVEWORKS API ハンズオン・ラボ

開発用ホストマシン

NoMachine でホストマシンを表示させてください。

DRIVEWORKS API ハンズオン・ラボ

データフロー概略



DRIVEWORKS API ハンズオン・ラボ

バーチャルカメラ入力によるオブジェクト DNN トラッカー

```
$ cd ~/driveworks-0.3/  
$ gedit ~/driveworks-0.3/samples/src/dnn/CMakeLists.txt
```

以下の様に「CMakeLists.txt」ファイルの8行目のコメントアウトを無効にしてください。

```
#TODO 1: Undo the following section to enable cross compiling of project  
add_subdirectory(sample_live_rggb_object_detector)
```

DRIVEWORKS API ハンズオン・ラボ

バーチャルカメラ入力によるオブジェクト DNN トラッカー

アプリケーションコードを変更するため、サンプルの「main.cpp」ファイルをエディタで開きます。

```
$ gedit ~/driveworks-  
0.3/samples/src/dnn/sample_live_rggb_object_detector/main.cpp
```

DRIVEWORKS API ハンズオン・ラボ

バーチャルカメラ入力によるオブジェクト DNN トラッカー

センサーの初期化（214行目）

(gedit: [Edit]-[Preferences]-[Display line numbers] をチェックで行数が左端に表示されます。)

```
// create sensor abstraction layer
dwSAL_initialize(&gSal, gSdk);

// create GMSL Camera interface
uint32_t cameraSiblings    = 0U;
float32_t cameraFramerate = 0.0f;
dwImageType imageType;
// TODO 2: Copy these input parameters to createVideoReplay function
// gCameraSensor, gCameraWidth, gCameraHeight, cameraSiblings, cameraFramerate,
imageType, gSal
createVideoReplay(/* TODO 2 */, gArguments.has("video") ?
gArguments.get("video") : "");
```

DRIVEWORKS API ハンズオン・ラボ

バーチャルカメラ入力によるオブジェクト DNN トラッカー

デバイスにより NVMedia もしくは CUDA センサを生成 (223行目)

```
//TODO 3: Create NVMediaSensor for PX2 and a Cuda sensor for Host
//gSdk, 0, gCameraSensor, gCameraWidth, gCameraHeight
#ifdef VIBRANTE
    gSensorIO.reset(new SensorIONvmedia(/* TODO 3 */));
#else
    gSensorIO.reset(new SensorIOCuda(/* TODO 3 */));
#endif
```

DRIVE PX2 の場合

ホストの場合

DRIVEWORKS API ハンズオン・ラボ

バーチャルカメラ入力によるオブジェクト DNN トラッカー

SensorIONvmedia を見てみましょう。(dnn_common/SensorIONvmedia.cpp: 85行目)

```
// image API translator
displayImageProperties.type = DW_IMAGE_NVMEDIA;
result = dwImageStreamer_initialize(&m_nvm2gl, &displayImageProperties,
DW_IMAGE_GL, context);
if (result != DW_SUCCESS) {
    throw std::runtime_error(std::string("Cannot init image streamer nvm-
gl: ") + dwGetStatusName(result));
}

cameraImageProperties.type = DW_IMAGE_NVMEDIA;
result = dwImageStreamer_initialize(&m_nvm2cudaYuv, &cameraImageProperties,
DW_IMAGE_CUDA, context);
if (result != DW_SUCCESS) {
    throw std::runtime_error(std::string("Cannot init image streamer nvm-
cuda: ") + dwGetStatusName(result));
}
```

DRIVEWORKS API ハンズオン・ラボ

バーチャルカメラ入力によるオブジェクト DNN トラッカー

DNNの初期化 (234行目)

```
//TODO 4: Copy the global sdk initialization parameter to initDNN function  
//gSdk  
gDnnInference.reset(new DNNInference(/*TODO 4*/));
```

DRIVEWORKS API ハンズオン・ラボ

バーチャルカメラ入力によるオブジェクト DNN トラッカー

終了処理 (Clean up) (268行目)

```
//TODO 5: Stop the camera sensor  
//gCameraSensor  
dwSensor_stop(/* TODO 5 */);  
  
//TODO 6: Release the camera sensor  
//&gCameraSensor  
dwSAL_releaseSensor(/* TODO 6 */);
```

DRIVEWORKS API ハンズオン・ラボ

ホストから DRIVE PX2 に対するクロス・コンパイル

ビルド用のディレクトリ（build）を作成し、その中で CMake を使用して Makefile を生成します。
この Makefile を使用してクロス・コンパイルを行います。

開発ホストマシンのデスクトップ上にある README.txt にクロス・コンパイルをするための一連のコマンドが記載されています。

それに従ってコンパイルすることで DRIVE PX2 用実行ファイルが生成されます。

DRIVEWORKS API ハンズオン・ラボ

README.txt 内で指示されているクロス・コンパイル方法

コード生成用の「build」ディレクトリを作ります。 (すでに「build」ディレクトリがある場合はまず `rm -r build` で消去します。)

```
$ cd ~/driveworks-0.3
$ mkdir build
$ cd build
```

CMake を使用して Makefile を生成します。

```
$ cmake -DCMAKE_BUILD_TYPE=Release ¥
  -DCMAKE_TOOLCHAIN_FILE=/home/ubuntu/driveworks-0.3/samples/cmake/Toolchain-V4L.cmake ¥
  -DVIBRANTE_PDK:STRING=/home/ubuntu/Beta3/VibranteSDK/vibrante-tl86ref-linux ¥
~/driveworks-0.3/samples
```

生成された Makefile を使用して DRIVE PX2 用実行ファイルをコンパイルします。

```
$ make
$ make install
```

クロス・コンパイルに成功し実行用のバイナリ・ファイルが生成されました。実行ファイルに適切なパーミッションを設定します。

```
$ chmod -R 777 install/bin
```

DRIVEWORKS API ハンズオン・ラボ

DRIVE PX2

NoMachine で DRIVE PX2 を表示させてください。

DRIVEWORKS API ハンズオン・ラボ

実行ファイル転送：ホスト → DRIVE PX2

ホストマシン上の実行ファイルをDRIVE PX2 にコピーします。

下記の <AMAZON EC2 INSTANCE NAME> の部分を開発用ホストマシン名に置き換えて、scp コマンドで DRIVE PX2 に実行をコピーします。

まずは HOME ディレクトリ (~/) にコピーします。
パスワードを聞かれたら qwiklabs のパスワードを入します。

```
$ scp ubuntu@<AMAZON EC2 INSTANCE NAME>:/home/ubuntu/driveworks-0.3/build/install/bin/sample_live_rggb_object_detector ~/
```

HOME ディレクトリから DriveWorks の bin ディレクトリにコピーします。
パスワードを聞かれたら「nvidia」を入力します。

```
$ sudo mv ~/sample_live_rggb_object_detector /usr/local/driveworks/bin
```

DRIVEWORKS API ハンズオン・ラボ

生成されたサンプルコードの実行

クロス・コンパイルされたサンプルを実行させましょう。

DRIVE PX2 用の NoMachine を表示させ、以下を実行してください。

```
$ cd /usr/local/driveworks/bin  
$ ./sample_live_rggb_object_detector
```

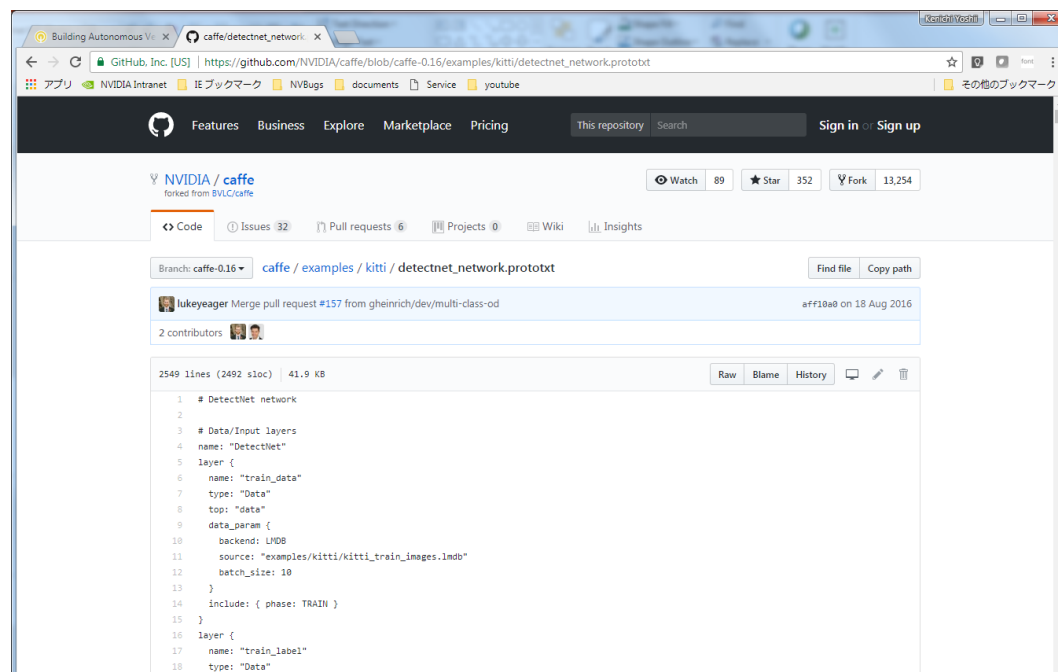

本日使用した DNN モデル (DETECTNET)

DETECTNET

DetectNet のモデル

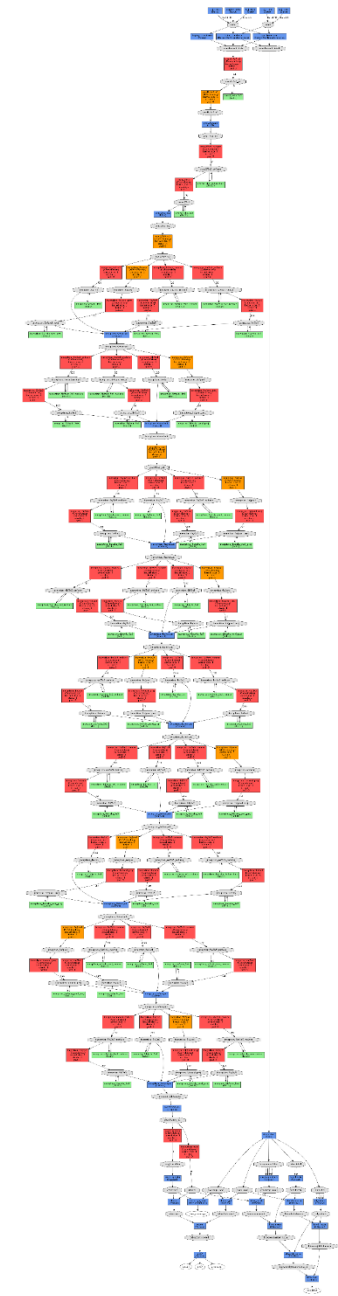
DetectNet の Caffe モデル

- [https://github.com/NVIDIA/caffe/tree/caffe-0.16/examples/kitti](https://github.com/NVIDIA/caffe/tree/caffe-0.16/examples/kitti/detectnet_network.prototxt)



The screenshot shows the GitHub repository for NVIDIA/caffe. The file path is `caffe / examples / kitti / detectnet_network.prototxt`. The file is 2549 lines long, 41.9 KB, and was last committed by `lukeyeager` on August 18, 2016. The code defines a DetectNet network with input layers for training data and labels.

```
1 # DetectNet network
2
3 # Data/Input layers
4 name: "DetectNet"
5 layer {
6   name: "train_data"
7   type: "Data"
8   top: "data"
9   data_param {
10     backend: LVDB
11     source: "examples/kitti/kitti_train_images.lmdb"
12     batch_size: 10
13   }
14   include: { phase: TRAIN }
15 }
16 layer {
17   name: "train_label"
18   type: "Data"
```

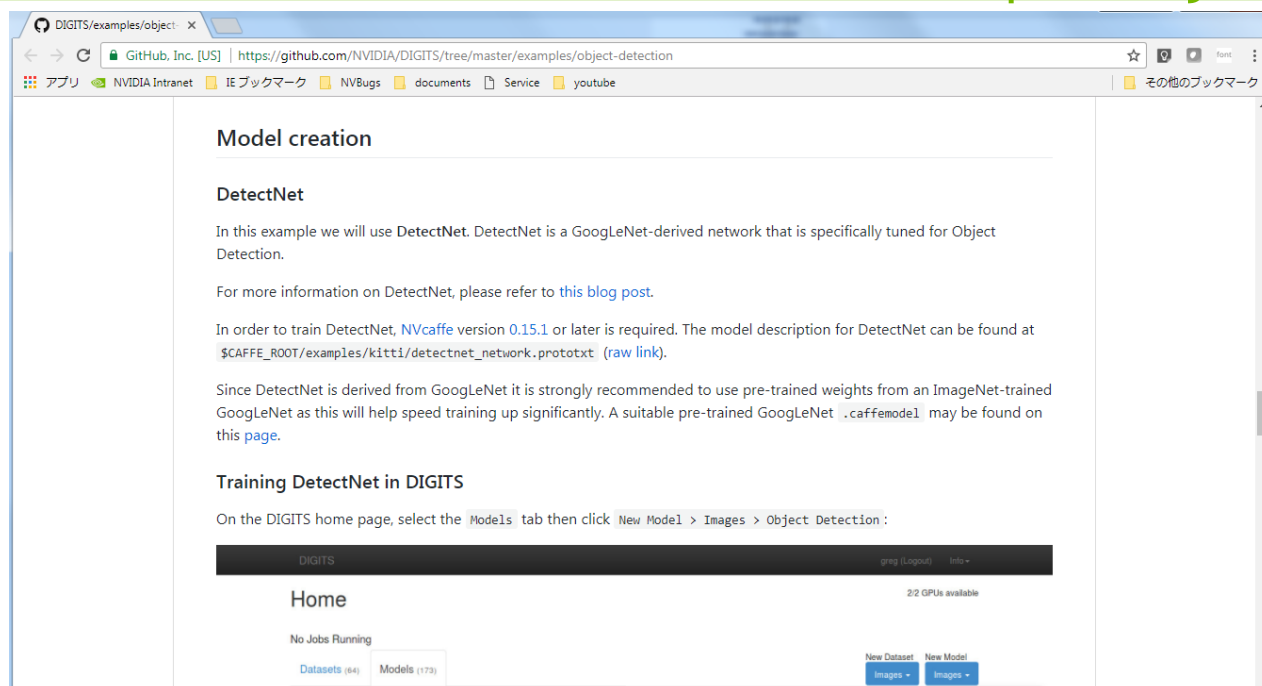


DETECTNET

DetectNet の学習方法

DIGITS を使用した DetectNet の学習のチュートリアル


- <https://github.com/NVIDIA/DIGITS/tree/master/examples/object-detection>



DETECTNET

TensorRT による最適化

DIGITS で学習済みの DetectNet を TensorRT で最適化後、DriveWorks の DNN サンプルで 사용할 수 있습니다. (本日のこのラボでは取扱いません。)



DriveWorks SDK Reference

0.3.422 Release

WelcomeDriveWorks APIDriveWorks SamplesDriveWorks ToolsMore

Search

▼ DriveWorks SDK Reference

Development Guide

▶ DriveWorks API

▶ DriveWorks Samples

▼ DriveWorks Tools

GUI Calibration Tool

CLI Calibration Tool

Prerecord Checker

Recording Library

▶ GUI Recording Tool

CLI Recording Tool

Replayer Tool

Sensor Recording Indexer

TensorRT Optimization Tool

Map Data Tool

▶ More

DriveWorks TensorRT Optimizer Tool

This tool enables optimization of a given Caffe model using TensorRT. For more information, see the *NVIDIA DriveWorks Release Notes*.

`./tensorrt_optimization`

Command Line Options

The following lists the required and optional command line arguments.

Required Arguments

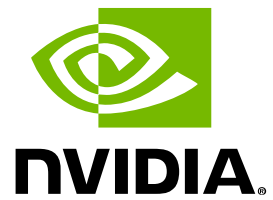
- `--prototxt`: Deploy file that describes the Caffe network (e.g., `--prototxt=deploy.prototxt`)
- `--caffemodel`: Caffe model file that contain weights (e.g., `--caffemodel=weights.caffemodel`)
- `--outputBlobs`: Names of output blobs combined with a comma (e.g., `--outputBlobs=bboxes,coverage`)

Optional Arguments

- `--iterations`: Number of iterations to run to measure speed (e.g., `--iterations=100`. Default = 10)
- `--batchSize`: Batchsize of the model to be generated (e.g., `--batchSize=2`. Default: 1)



DEEP
LEARNING
INSTITUTE



DEEP
LEARNING
INSTITUTE

www.nvidia.com/dli