Real-time Video Data Transmission and Processing with YOLO based on AI + X Playground

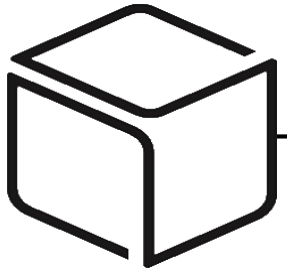
GIST college EECS
20165151 이창하

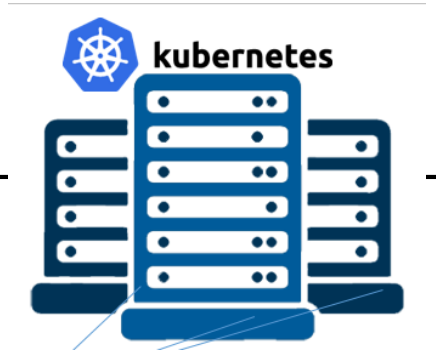
AI + X playground_ Yolo-Processed Video Sata Streaming

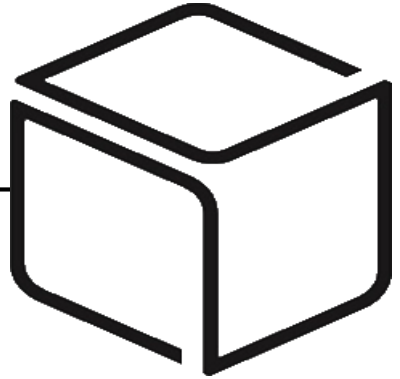
End

Edge

Sage2







1. Transmitting Video Data from Webcam

2. Tagging the Video Data with YOLO

3. Transmitting the Data to Sage2

4. Display the Tagged Video Data

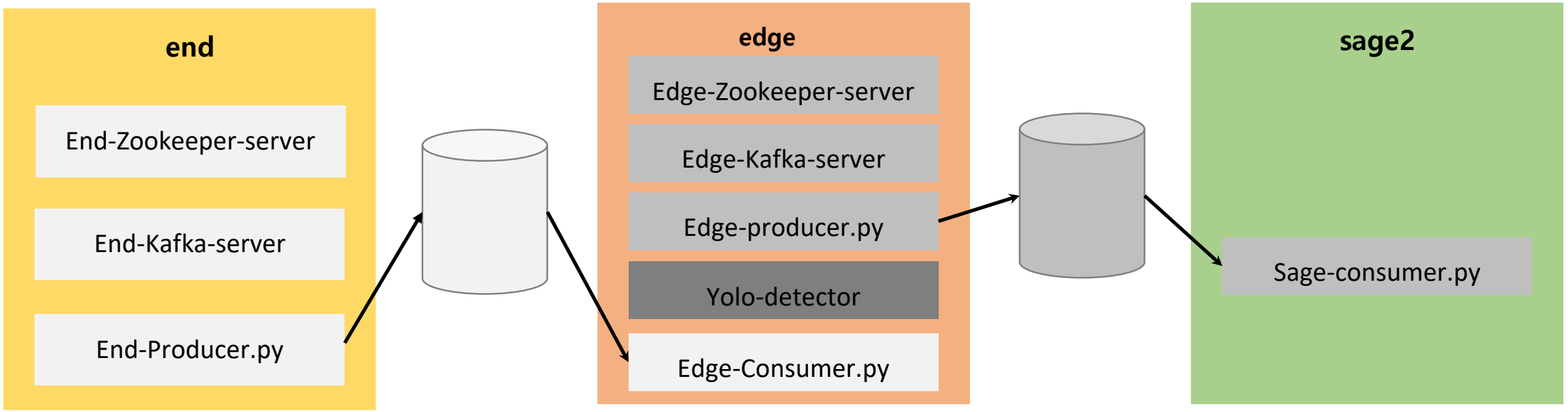
Video stream transmitted with Kafka

Kafka with python

: To transmit video data from box to box
(use python implemented kafka)

We'll use kafka to transmit beteween pairs of boxes.

End-to-Edge: webcam video stream is transmitted to topic,
and edge-side consumer catches the messages from the topic
Edge-to-Sage2: edge executes object-detection frames
of the messages one-by-one, and again these images are transmitted
to topic. Then Sage2-side consumer catch them



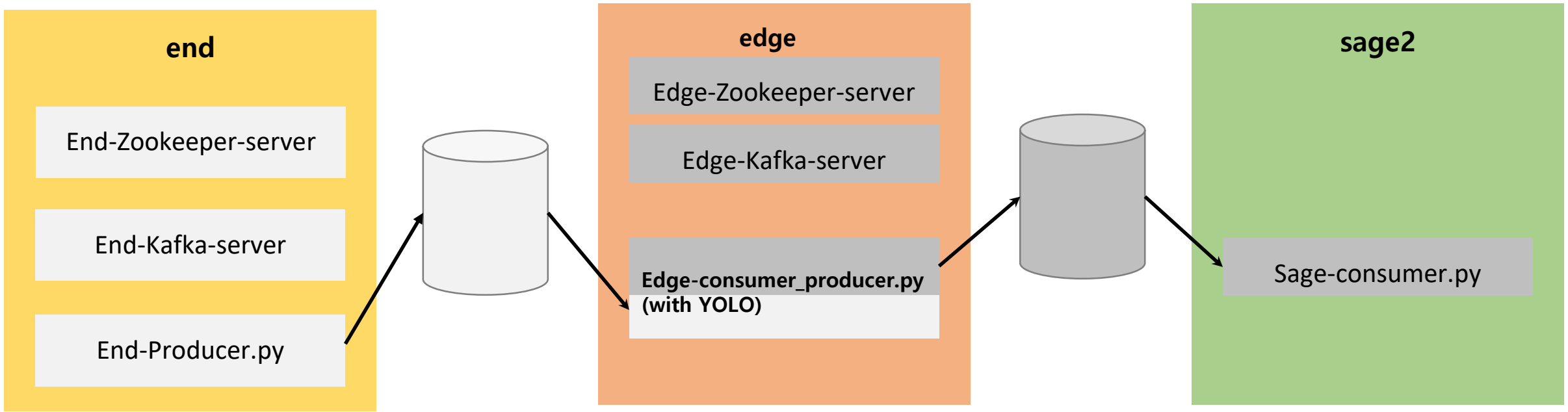
Video stream transmitted with Kafka

OpenCV-python

: To use YOLO model, i chose python-language. Because, It is more easy to use and I didn't want to change kafka implementation

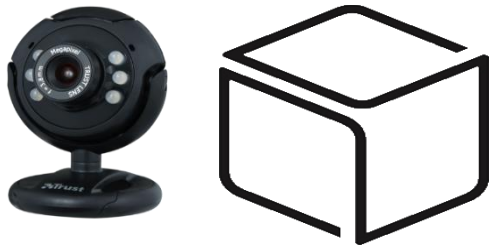
To avoid overhead from video transmisson,

I intergrated 'Edge_consumer', 'yolo_detector' and 'Edge_producer' into 'Edge-consumer_producer.py'



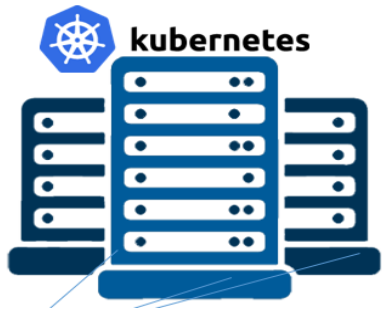
1) Prerequisite [0/15]

End



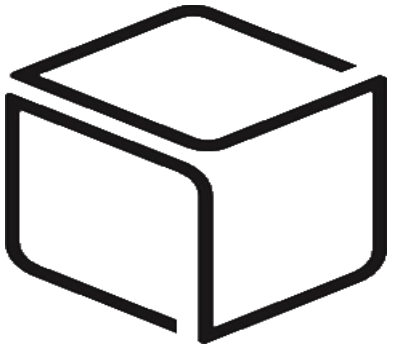
- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0

Edge



- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn

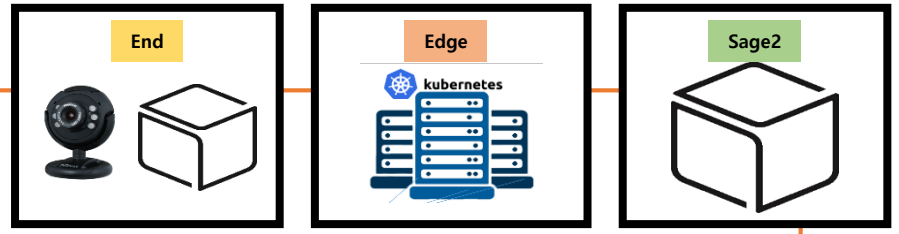
Sage2



- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0

1) Prerequisite [1/15]

- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0



Installation

$sudo apt-get update
$sudo apt-get install python3

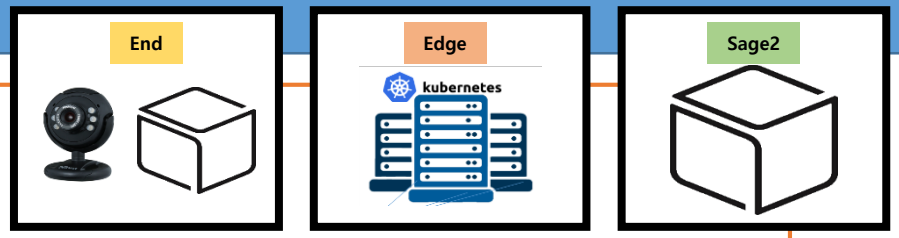
Verify the installation

$python3 –version

```
(venv-python3.6) netcs@netcs-NUC8i7HVK:~/privateWorkPlace$ python3 --version
Python 3.6.9
(venv-python3.6) netcs@netcs-NUC8i7HVK:~/privateWorkPlace$
```

1) Prerequisite [2/15]

- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0



Installation

$sudo apt-get update
$sudo apt-get install python3-pip

Verify the installation

$pip3 --version

```
(venv-python3.6) netcs@netcs-NUC8i7HVK:~/privateWorkPlace$ pip3 --version
pip 20.0.2 from /home/netcs/venv-python3.6/lib/python3.6/site-packages/pip (python 3.6)
```

Installation

$sudo pip3 install numpy

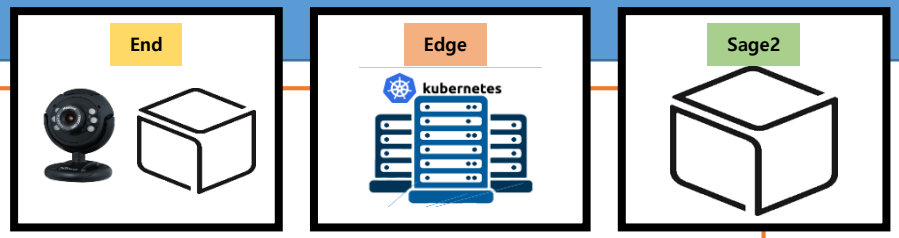
Verify the installation

$python3
>>import numpy as np
>>np.random.rand(4,4)

```
array([[ 0.58241142,  0.96068432,  0.64404447,  0.13767207],
       [ 0.14788868,  0.65965442,  0.86432515,  0.60396838],
       [ 0.15875242,  0.89423004,  0.30254474,  0.40171501],
       [ 0.78022208,  0.89349476,  0.83617177,  0.23359508]])
```

1) Prerequisite [3/15]

- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0



Installation

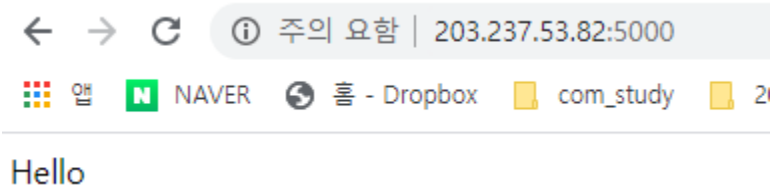
$sudo pip3 install flask

Verify the installation

```
$python3
>>from flask import Flask
>>app = Flask(__name__)
>>@app.route('/')
...def hello_world():
...                 return "hello"
...
>>app.run(host="0.0.0.0", port="5000")
```

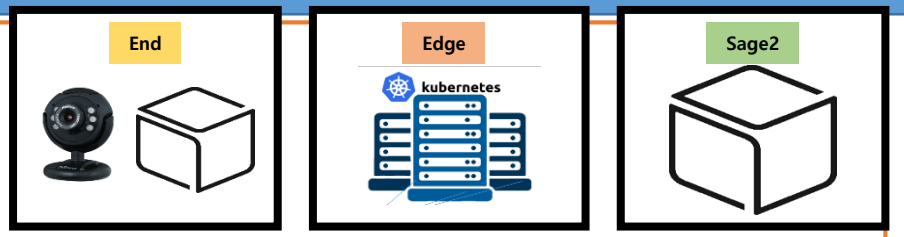
```
>>> app.run(host="0.0.0.0", port="5000")
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Web browser
: http://__yourNucIP__:5000/



1) Prerequisite [4/15]

- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0



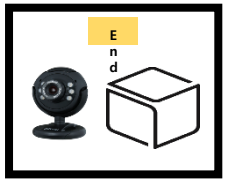
Installation

1) Install java-JDK

```
$sudo add-apt-repository ppa:openjdk-r/ppa
$sudo apt-get update
$sudo apt-get install openjdk-8-jdk
```

2) Prepare git pulling to each box

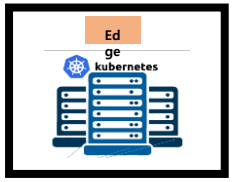
```
$mkdir Kafka
$cd Kafka
$git config core.sparseCheckout true
$git remote add –f origin https://github.com/GuruneLee/NetCS_AI-X-Demo
```



End box

```
$echo “01End” >> .git/info/sparse-checkout
$git pull origin master
```

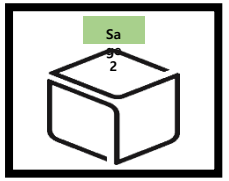
```
01End/
├── kafka_2.12-2.2.1
│   ├── bin
│   ├── config
│   ├── libs
│   ├── LICENSE
│   ├── logs
│   ├── NOTICE
│   └── site-docs
└── kafka-python-video-streaming
    └── producer1.py
```



Edge box

```
$echo “02Edge” >> .git/info/sparse-checkout
$git pull origin master
```

```
02Edge/
├── kafka_2.12-2.2.1
│   ├── bin
│   ├── config
│   ├── libs
│   ├── LICENSE
│   ├── logs
│   ├── NOTICE
│   └── site-docs
└── kafka-python-video-streaming-car
    ├── consumer1_yolo_producer2.py
    └── yolo-coco
```



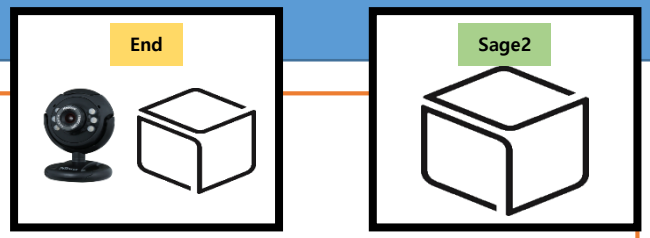
Sage2 box

```
$echo “03Sage” >> .git/info/sparse-checkout
$git pull origin master
```

```
03Sage/
└── consumer2.py
```

1) Prerequisite [5/15]

- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0



Step#1. Prerequisite for OpenCV

1) Upgrade existing pakages

$sudo apt-get update
$sudo apt-get upgrade

2) Install the pre-required pakages

$sudo apt-get install build-essential cmake
$sudo apt-get install pkg-config

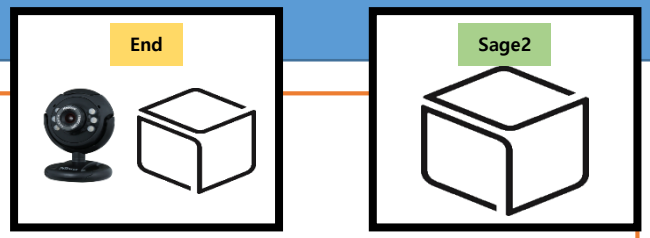
$sudo apt-get install libjpeg-dev libtiff5-dev libpng-dev
$sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libxvidcore-dev libx264-dev libxine2-dev
$sudo apt-get install libv4l-dev v4l-utils
$sudo apt-get install libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
$sudo apt-get install libgtk2.0-dev

$sudo apt-get install mesa-utils libgl1-mesa-dri libgtkgl2.0-dev libgtkglext1-dev
$sudo apt-get install libatlas-base-dev gfortran libeigen3-dev

참고 사이트: https://webnautes.tistory.com/1186

1) Prerequisite [6/15]

- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0



Step#2. Build

1) Download the OpenCV-4.2.0 src

$mkdir opencv
$cd opencv

$wget –O opencv.zip https://github.com/opencv/opencv/archive/4.2.0.zip
$unzip opencv.zip

```
opencv-4.2.0  opencv_contrib  opencv_contrib.zip
```

2) Build the OpenCV-4.2.0

$cd opencv-4.2.0
$mkdir build
$cd build
Write the commands followed (If you want, you can modifiy the configs)

```
cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=/usr/local \
-D WITH_TBB=OFF \
-D WITH_IPP=OFF \
-D WITH_1394=OFF \
-D BUILD_WITH_DEBUG_INFO=OFF \
-D BUILD_DOCS=OFF \
-D INSTALL_C_EXAMPLES=ON \
-D INSTALL_PYTHON_EXAMPLES=ON \
-D BUILD_EXAMPLES=OFF \
-D BUILD_TESTS=OFF \
-D BUILD_PERF_TESTS=OFF \
-D WITH_QT=OFF \
-D WITH_GTK=ON \
-D WITH_OPENGL=ON \
-D OPENCV_EXTRA_MODULES_PATH=../../opencv_contrib-4.2.0/modules \
-D WITH_V4L=ON  \
-D WITH_FFMPEG=ON \
-D WITH_XINE=ON \
-D BUILD_NEW_PYTHON_SUPPORT=ON \
-D OPENCV_GENERATE_PKGCONFIG=ON ../
```

If properly done, you can see this message

```
-- Configuring done
-- Generating done
-- Build files have been written to: /home/webnautes/opencv/opencv-4.2.0/build
```

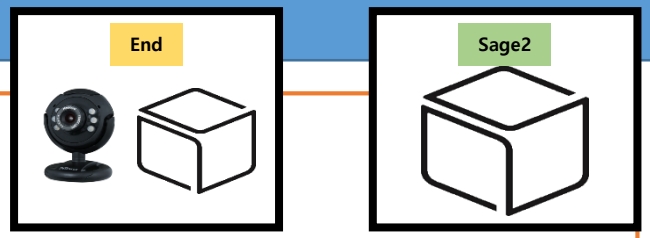
Also there my be a list of your configuration. But, If there are any python-related message, you should re-build with serveral additional

```
-D PYTHON2_INCLUDE_DIR=/usr/include/python2.7 \
-D PYTHON2_NUMPY_INCLUDE_DIRS=/usr/lib/python2.7/dist-packages/numpy/core/include/ \
-D PYTHON2_PACKAGES_PATH=/usr/lib/python2.7/dist-packages \
-D PYTHON2_LIBRARY=/usr/lib/x86_64-linux-gnu/libpython2.7.so \
-D PYTHON3_INCLUDE_DIR=/usr/include/python3.6m \
-D PYTHON3_NUMPY_INCLUDE_DIRS=/usr/lib/python3/dist-packages/numpy/core/include/  \
-D PYTHON3_PACKAGES_PATH=/usr/lib/python3/dist-packages \
-D PYTHON3_LIBRARY=/usr/lib/x86_64-linux-gnu/libpython3.6m.so \
```

참고 사이트: https://webnautes.tistory.com/1186

1) Prerequisite [7/15]

- Python 2.7.17
- Numpy 1.16.6
- Flask 1.1.2
- Kakfa 2.0.1
- OpenCV 4.2.0



Step#3. Compile

1) Check the number of CPU

$cat /proc/cpuinfo | grep processor | wc –l

```
$ cat /proc/cpuinfo | grep processor | wc -l
```

2) Make

$make –j (no space between –j and)

If properly done, you can see this message.

```
[100%] Building CXX object modules/python3/CMakeFiles/opencv_python3.dir/__/src2/cv2.cpp.o
[100%] Building CXX object modules/python2/CMakeFiles/opencv_python2.dir/__/src2/cv2.cpp.o
[100%] Linking CXX shared module ../../lib/python3/cv2.cpython-36m-x86_64-linux-gnu.so
[100%] Linking CXX shared module ../../lib/cv2.so
[100%] Built target opencv_python3
[100%] Built target opencv_python2
```

3) Install

$sudo make install

4) Check whether the config file exists

$cat /etc/ld.so.conf.d/*

If you cannot find that file

$sudo sh -c 'echo '/usr/local/lib' > /etc/ld.so.conf.d/opencv.conf'

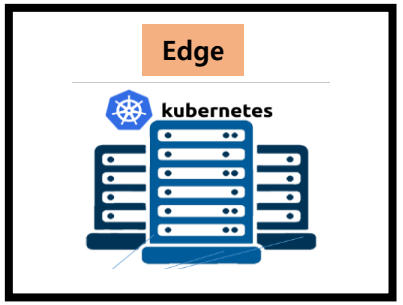
5) Finish the installation

$sudo ldconfig

참고 사이트: https://webnautes.tistory.com/1186

1) Prerequisite [8/15]

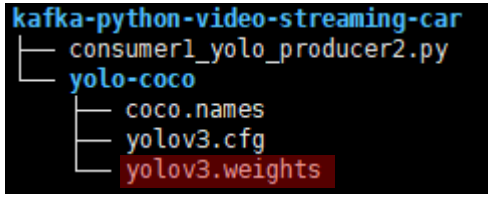
- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn



Installation

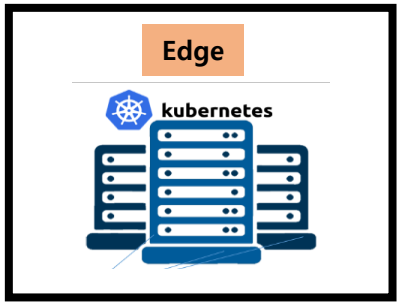
1) Access the 02Edge directory
$cd 02Edge
$cd kafka-python-video-streaming-car
$cd yolo-coco

2) Wget the yolov3-weight
$wget https://pjreddie.com/media/files/yolov3.weights



1) Prerequisite [9/15]

- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn



We will use OpenCV's 'dnn' module with NVIDIA GPUs, CUDA, and cuDNN.
So, we should install all of relevant stuffs for prerequisite.

Step#1. Nvidia Driver

1) Check your OS version

```
$release="ubuntu"$(lsb_release -sr | sed -e "s/\.//g")
$echo $release
```

2) Install the repository

```
$sudo apt install sudo gnupg
$sudo apt-key adv --fetch-keys "http://developer.download.nvidia.com/compute/cuda/repos/"$release"/x86_64/7fa2af80.pub"
$sudo sh -c 'echo "deb http://developer.download.nvidia.com/compute/cuda/repos/'$release'/x86_64 /" >
/etc/apt/sources.list.d/nvidia-cuda.list'
$sudo sh -c 'echo "deb http://developer.download.nvidia.com/compute/machine-learning/repos/'$release'/x86_64 /" >
/etc/apt/sources.list.d/nvidia-machine-learning.list'
$sudo apt-get update
```

3) Check your Driver version

: Go to https://www.nvidia.co.kr/Download/index.aspx?lang=kr. And found the version of your gpu resource

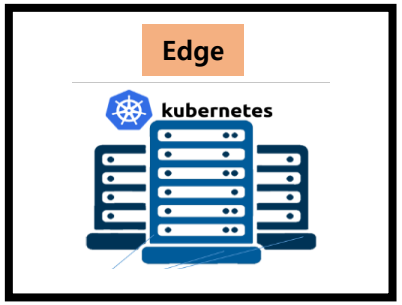
4) Install the Driver

```
$sudo dpkg -i <NVIDIA_DRIVER_DEBFILE>
$sudo apt-get update
$sudo apt-get install cuda-drivers
$sudo reboot // reboot을 해야 nvidia driver가 적용
```

참고 사이트: https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/

1) Prerequisite [10/15]

- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn



Step#2. CUDA

1) Go to https://developer.nvidia.com/cuda-downloads

And choose the each category that works for your environment. You should choose 'deb(local)' for last.
Then you will find command lines to install CUDA

2) Type the commands

And choose the each category that works for your environment. You should choose 'deb(local)' for last.
Then you will find command lines to install CUDA

3) Check whether the installation is done

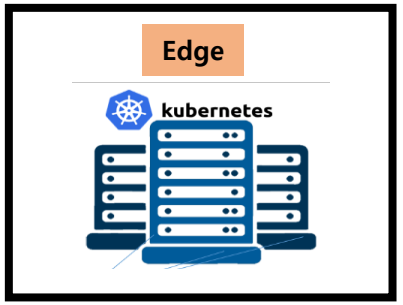
$nvidia-smi

```
netcs@edgebox2:~$ nvidia-smi
Wed May 27 18:43:45 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.64.00    Driver Version: 440.64.00    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:17:00.0 Off |                    0 |
| N/A   28C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

참고 사이트: https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/

1) Prerequisite [11/15]

- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn



Step#3. cuDNN

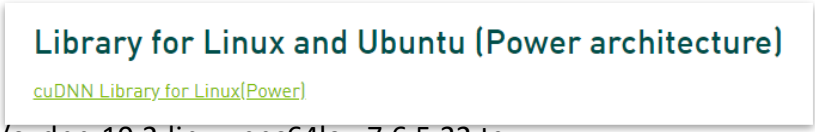
1) Login

Go to https://developer.nvidia.com/rdp/form/cudnn-download-survey and login.

2) Find the cuDNN version fitted with the CUDA

: you can find it in the web site

3) Install the 'cuDNN library for Linux'



$wget https://developer.nvidia.com/compute/machine-learning/cudnn/secure/7.6.5.32/Production/10.2_20191118/cudnn-10.2-linux-ppc64le-v7.6.5.32.tgz

4) Unzip the .tar file and finish the installation

$tar -xzvf cudnn-10.2-linux-x64-v7.6.5.32.tgz
$sudo cp cuda/include/cudnn.h /usr/local/cuda/include
$sudo cp cuda/lib64/libcudnn* /usr/local/cuda/lib64
$sudo chmod a+r /usr/local/cuda/include/cudnn.h /usr/local/cuda/lib64/libcudnn*

5) Add the below two lines into '~/.bashrc'

$ sudo vi ~/.bashrc
export PATH=/usr/local/cuda-10.2/bin${PATH:+:${PATH}}
export LD_LIBRARY_PATH=/usr/local/cuda-10.2/lib64\${LD_LIBRARY_PATH:+:${LD_LIBRARY_PATH}}
$sudo ln -sf /usr/local/cuda-10.2/targets/x86_64-linux/lib/libcudnn.so.7.6.5 /usr/local/cuda-10.2/targets/x86_64-linux/lib/libcudnn.so.7

6) Check whether the installation is done

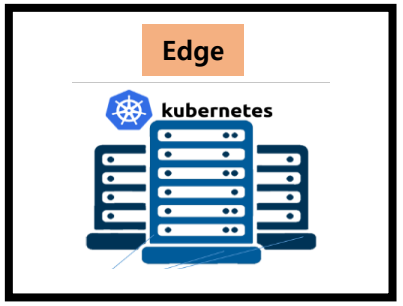
$nvcc --version

```
netcs@edgebox2:~$ nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2019 NVIDIA Corporation
Built on Wed_Oct_23_19:24:38_PDT_2019
Cuda compilation tools, release 10.2, V10.2.89
```

참고 사이트: https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/

1) Prerequisite [12/15]

- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn



To install the OpenCV with dnn-module is not quite different from above.

Step#1. Prerequisite for OpenCV

1) Upgrade existing pakages

```
$sudo apt-get update
$sudo apt-get upgrade
```

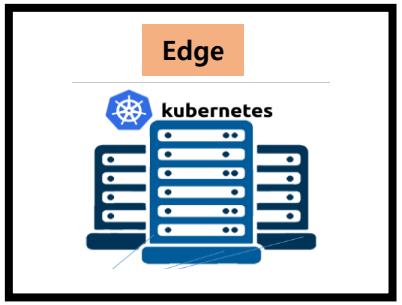
2) Install the pre-required pakages

```
$sudo apt-get install build-essential cmake
$sudo apt-get install pkg-config

$sudo apt-get install libjpeg-dev libtiff5-dev libpng-dev
$sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libxvidcore-dev libx264-dev libxine2-dev
$sudo apt-get install libv4l-dev v4l-utils
$sudo apt-get install libatlas-base-dev gfortran
$sudo apt-get install libgtk-3-dev

$sudo apt-get install python3-dev
```

참고 사이트: https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/

1) Prerequisite [13/15]

- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn



Step#2. Download OpenCV source code

$cd ~
$wget –O opencv.zip https://github.com/opencv/opencv/archive/4.2.0.zip
$wget –O opencv_contrib.zip https://github.com/opencv/opencv_contrib/archive/4.2.0.zip
$unzip opencv.zip
$unzip opencv_contrib.zip
$mv opencv-4.2.0 opencv
$mv opencv_contrib-4.2.0 opencv_contrib

Step#3. Determine your CUDA architecture version

$nvidia-smi

```
netcs@edgebox2:~$ nvidia-smi
Wed May 27 18:43:45 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 440.64.00    Driver Version: 440.64.00    CUDA Version: 10.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla T4            On   | 00000000:17:00.0 Off |                    0 |
| N/A   28C    P8     9W /  70W |      0MiB / 15109MiB |      0%      Default |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Tesla T4	7.5
Tesla V100	7.0
Tesla P100	6.0
Tesla P40	6.1
Tesla P4	6.1
Tesla M60	5.2
Tesla M40	5.2

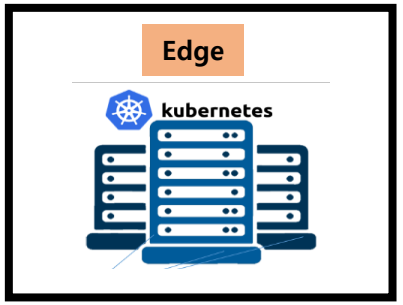
//You can find your NVIDIA GPU architecture version for your particular GPU using this page:
https://developer.nvidia.com/cuda-gpus

=> Please make note of this number

참고 사이트: https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/

1) Prerequisite [14/15]

- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn



Step#4. Configure OpenCV with NVIDIA GPU support

$cd ~/opencv
$mkdir build
$cd build
Write the commands followed (If you want, you can modifiy the configs)

```
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
    -D CMAKE_INSTALL_PREFIX=/usr/local \
    -D INSTALL_PYTHON_EXAMPLES=ON \
    -D INSTALL_C_EXAMPLES=OFF \
    -D OPENCV_ENABLE_NONFREE=ON \
    -D WITH_CUDA=ON \
    -D WITH_CUDNN=ON \
    -D OPENCV_DNN_CUDA=ON \
    -D ENABLE_FAST_MATH=1 \
    -D CUDA_FAST_MATH=1 \
    -D CUDA_ARCH_BIN=7.0 \
    -D WITH_CUBLAS=1 \
    -D OPENCV_EXTRA_MODULES_PATH=~/opencv_contrib/modules \
    -D HAVE_opencv_python3=ON \
    -D PYTHON_EXECUTABLE=~/.virtualenvs/opencv_cuda/bin/python \
    -D BUILD_EXAMPLES=ON ..
```

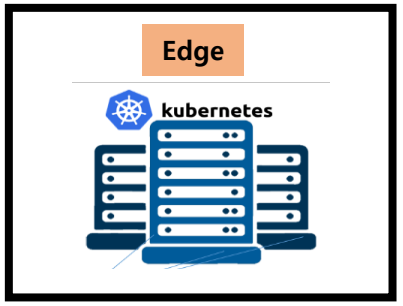
Making sure you set the 'CUDA_ARCH_BIN' variable based on your NVIDIA GPU

Tesla T4	7.5
Tesla V100	7.0
Tesla P100	6.0
Tesla P40	6.1
Tesla P4	6.1
Tesla M60	5.2
Tesla M40	5.2

참고 사이트: https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/

1) Prerequisite [15/15]

- yolov3-320 weight file
- Driver 440.64.00
- CUDA 10.2
- cuDNN 7.6.5
- OpenCV 4.2.0 with dnn



Step#5. Compile OpenCV with “dnn” GPU support

1) Check the number of CPU

$cat /proc/cpuinfo | grep processor | wc –l

```
$ cat /proc/cpuinfo | grep processor | wc -l
```

2) Make

$make –j (no space between –j and)

If properly done, you can see this message.

```
[100%] Building CXX object modules/python3/CMakeFiles/opencv_python3.dir/__/src2/cv2.cpp.o
[100%] Building CXX object modules/python2/CMakeFiles/opencv_python2.dir/__/src2/cv2.cpp.o
[100%] Linking CXX shared module ../../lib/python3/cv2.cpython-36m-x86_64-linux-gnu.so
[100%] Linking CXX shared module ../../lib/cv2.so
[100%] Built target opencv_python3
[100%] Built target opencv_python2
```

3) Install

$sudo make install

4) Check whether the config file exists

$cat /etc/ld.so.conf.d/*

If you cannot find that file

$sudo sh -c 'echo '/usr/local/lib' > /etc/ld.so.conf.d/opencv.conf'

5) Finish the installation

$sudo ldconfig

참고 사이트: https://www.pyimagesearch.com/2020/02/03/how-to-use-opencvs-dnn-module-with-nvidia-gpus-cuda-and-cudnn/

2) Excution[1/6]

Time line	End	Yolo-Edge	Sage2
	1) Zookeeper 서버 실행 2) Kafka 서버 실행 (broker)	1) Zookeeper 서버 실행 2) Kafka 서버 실행 (broker)	
	End-produce.pyr 실행		
		Edge-consumer_producer.py 실행	
			Sage-consumer.py실행

2) Excution[2/6]

Time line	End	Yolo-Edge	Sage2
	1) Zookeeper 서버 실행 2) Kafka 서버 실행 (broker)	1) Zookeeper 서버 실행 2) Kafka 서버 실행 (broker)	

command

Terminal01

1) cd kafka_2.12-2.2.1

2) bin/zookeeper-server-start.sh config/zookeeper.properties

Terminal02

1) cd kafka_2.12-2.2.1

2) bin/kafka-server-start.sh config/server.properties

2) Excution[3/6]

Time line	End	Yolo-Edge	Sage2
	1) Zookeeper 서버 실행 2) Kafka 서버 실행 (broker)	1) Zookeeper 서버 실행 2) Kafka 서버 실행 (broker)	

command

Terminal01

1) cd kafka_2.12-2.2.1

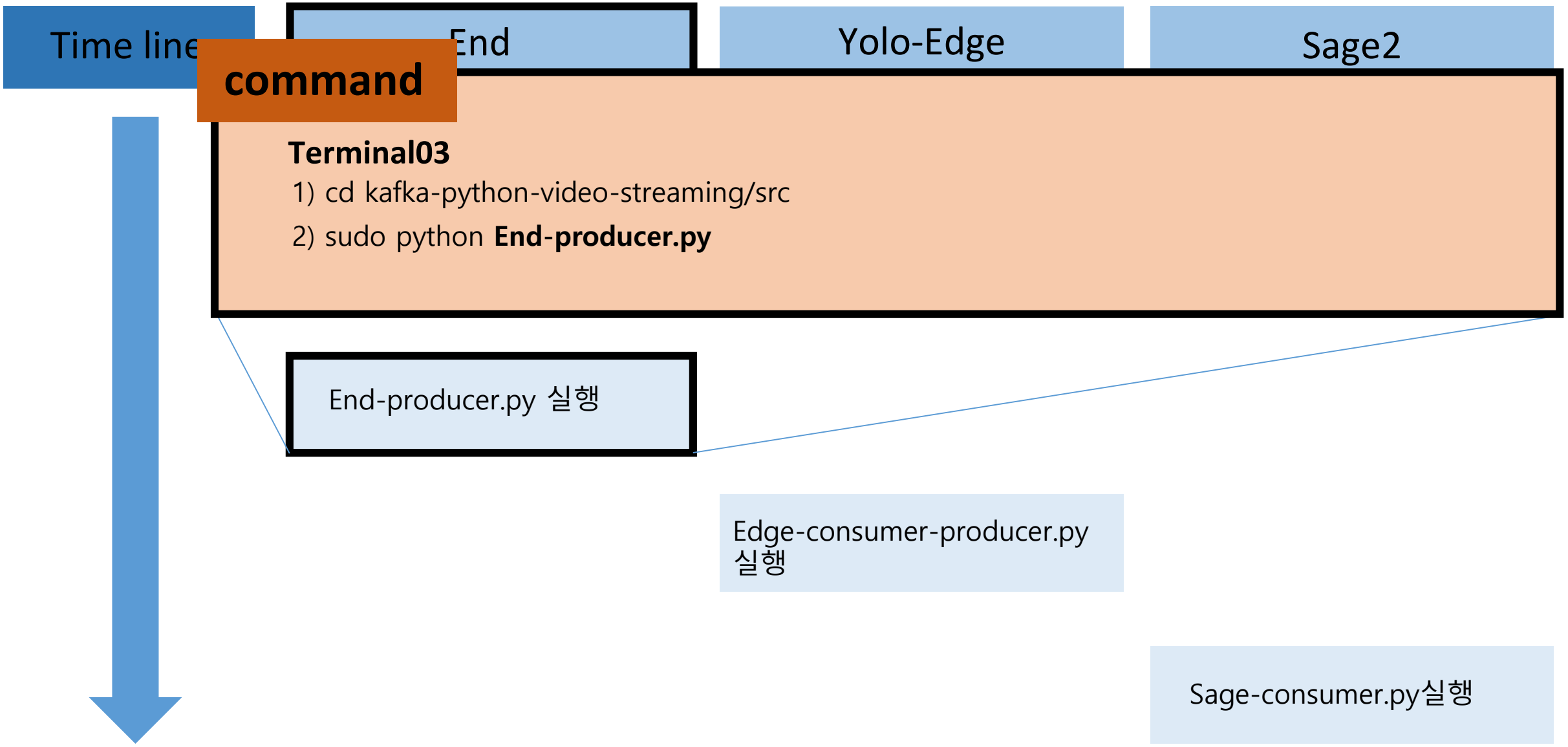
2) bin/zookeeper-server-start.sh config/zookeeper.properties

Terminal02

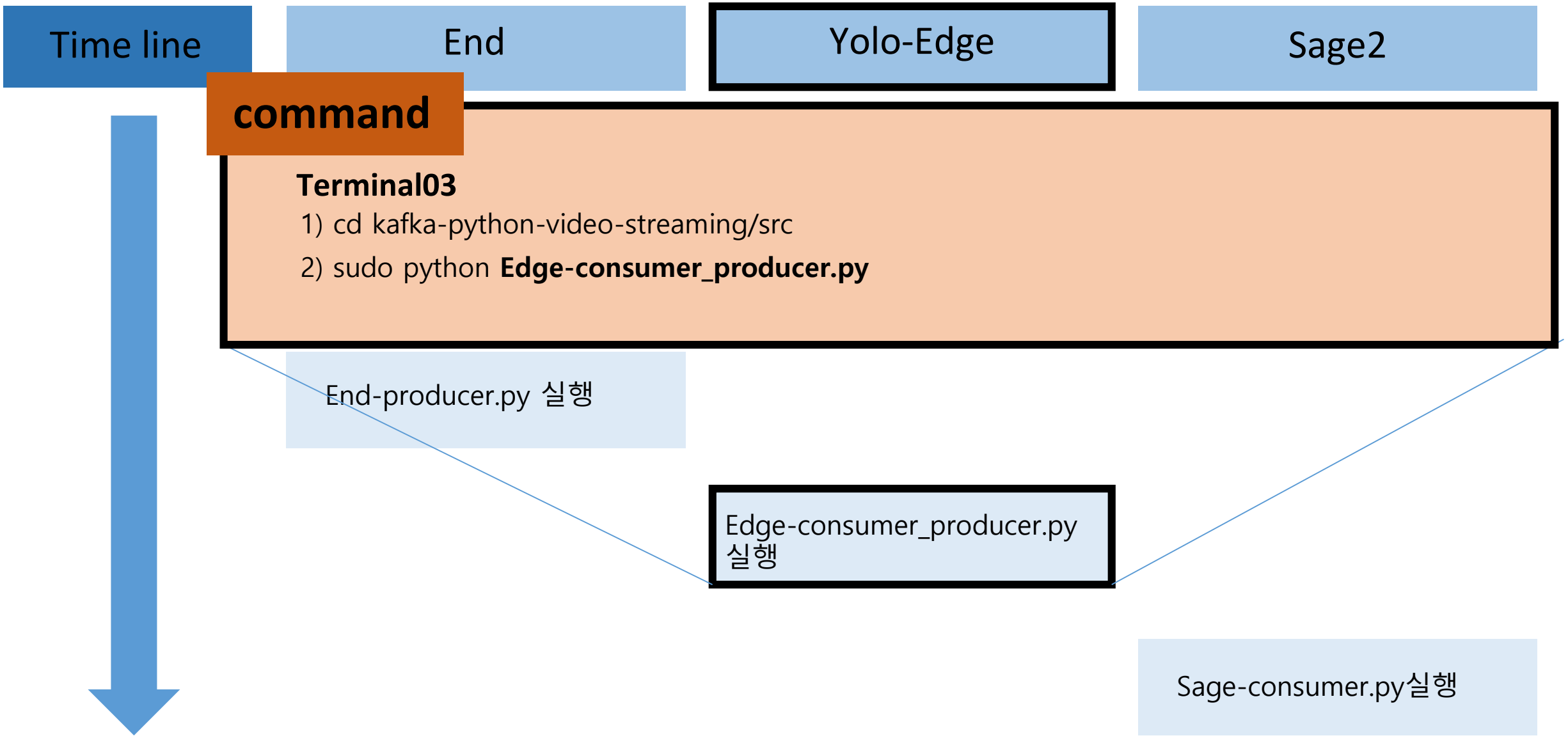
1) cd kafka_2.12-2.2.1

2) bin/kafka-server-start.sh config/server.properties

2) Excution[4/6]



2) Excution[5/6]



2) Excution[6/6]