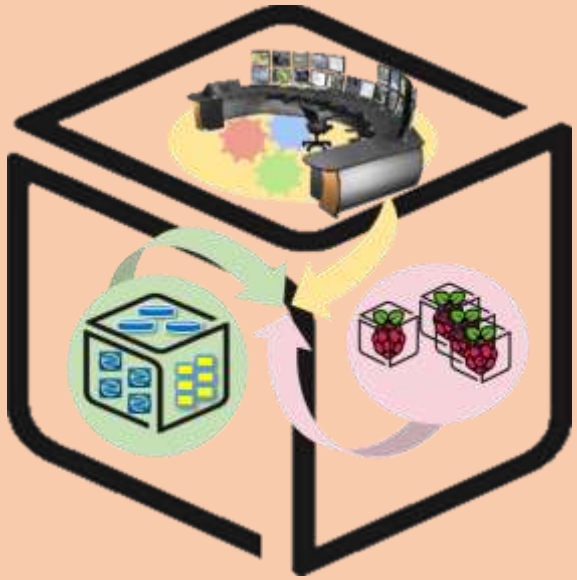
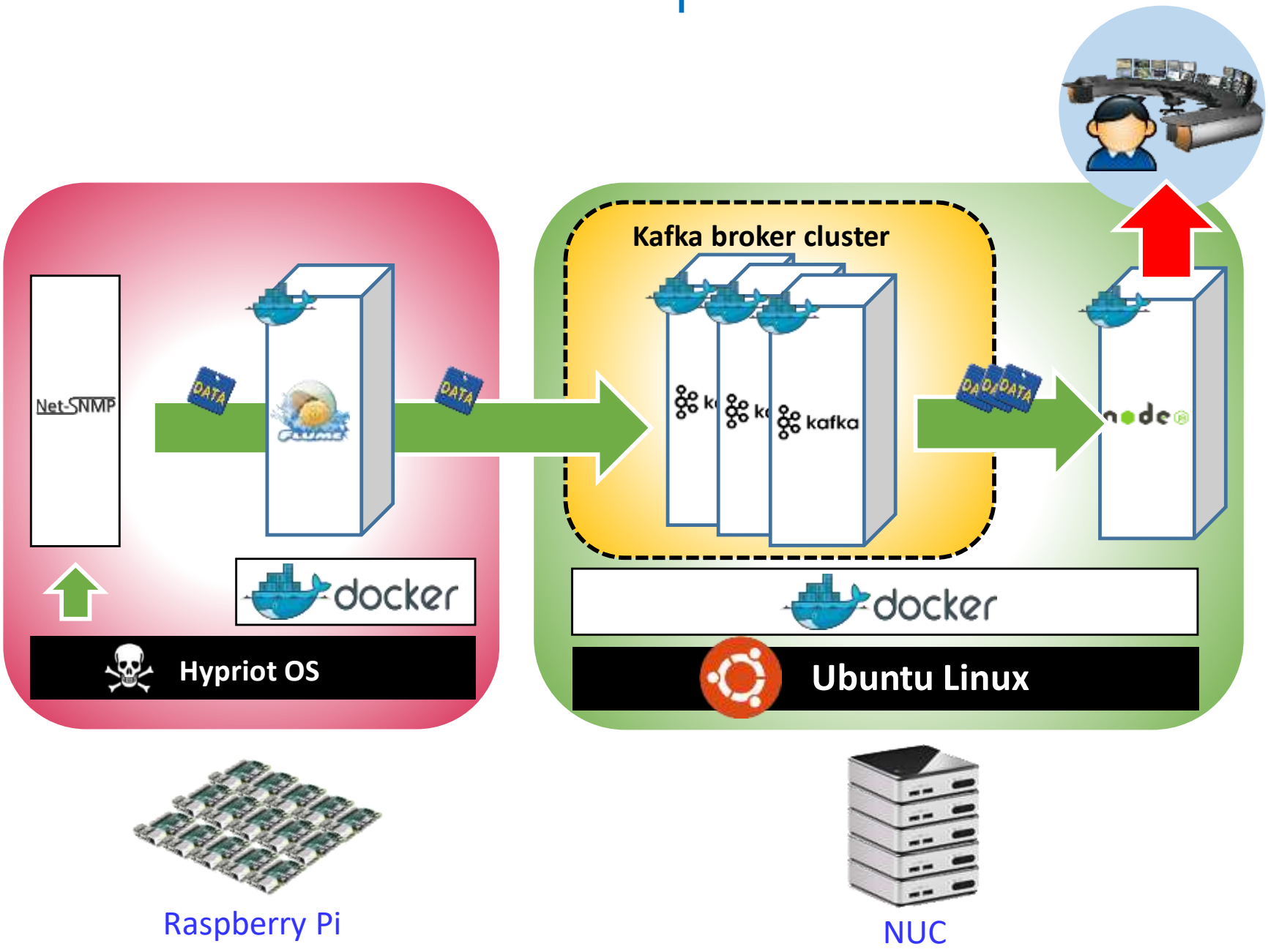
Computer Systems For AI-inspired Cloud Theory & Lab.

Lab #2: Inter-Connect

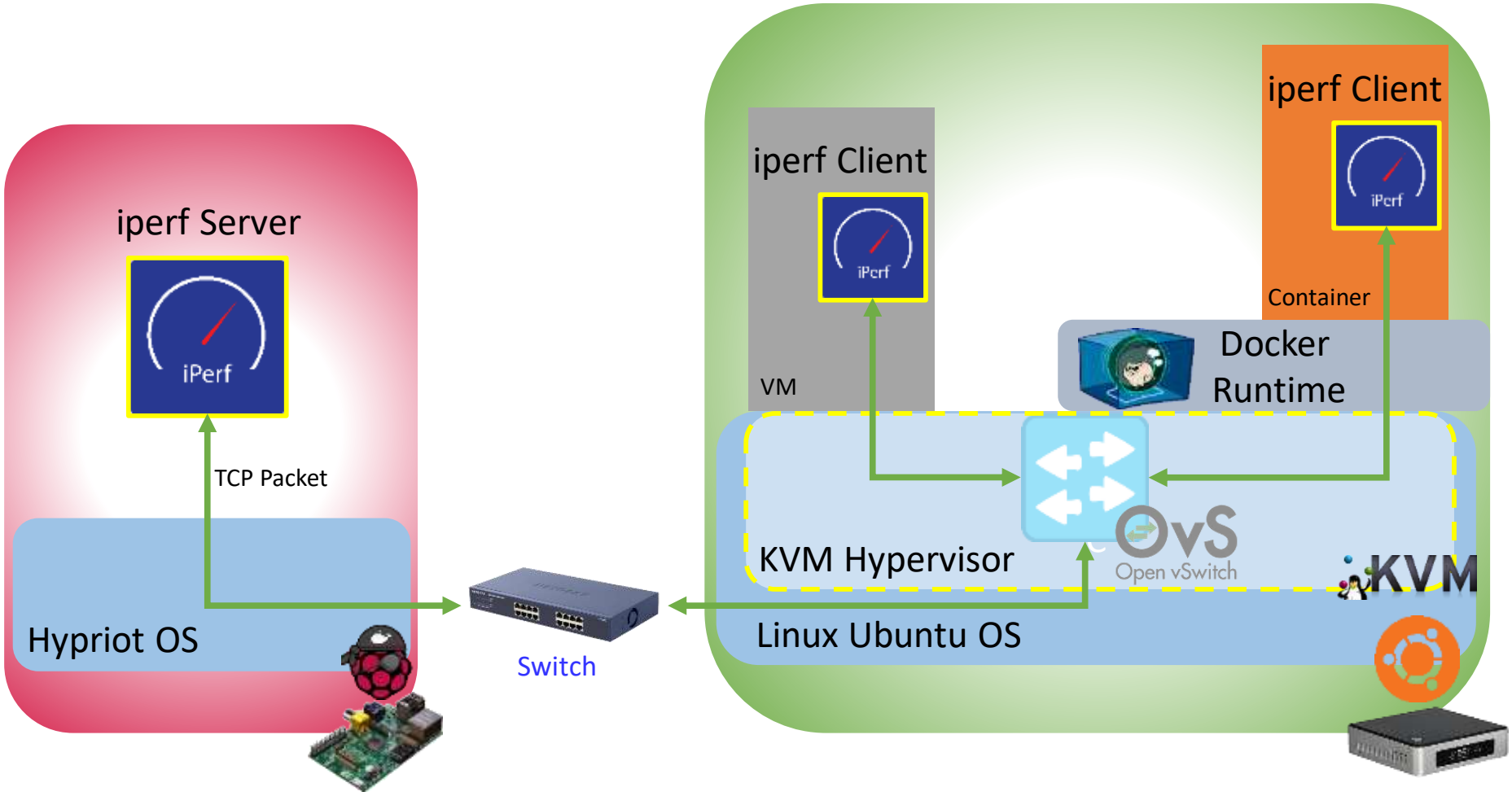


https://github.com/SmartX-Labs/SmartX-Mini-MOOC

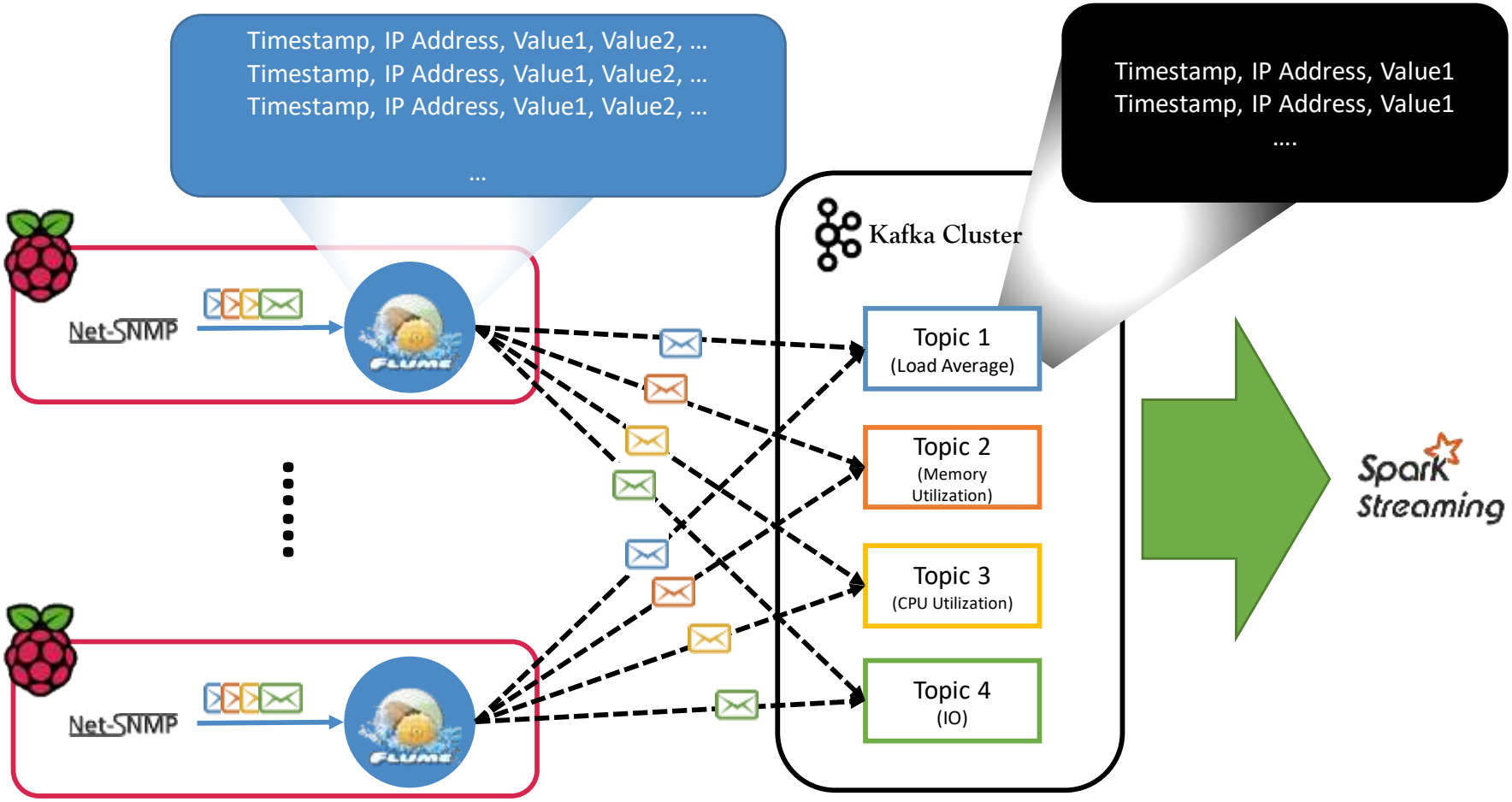
Inter-Connect Lab: Concept



Physical Inter-Connect

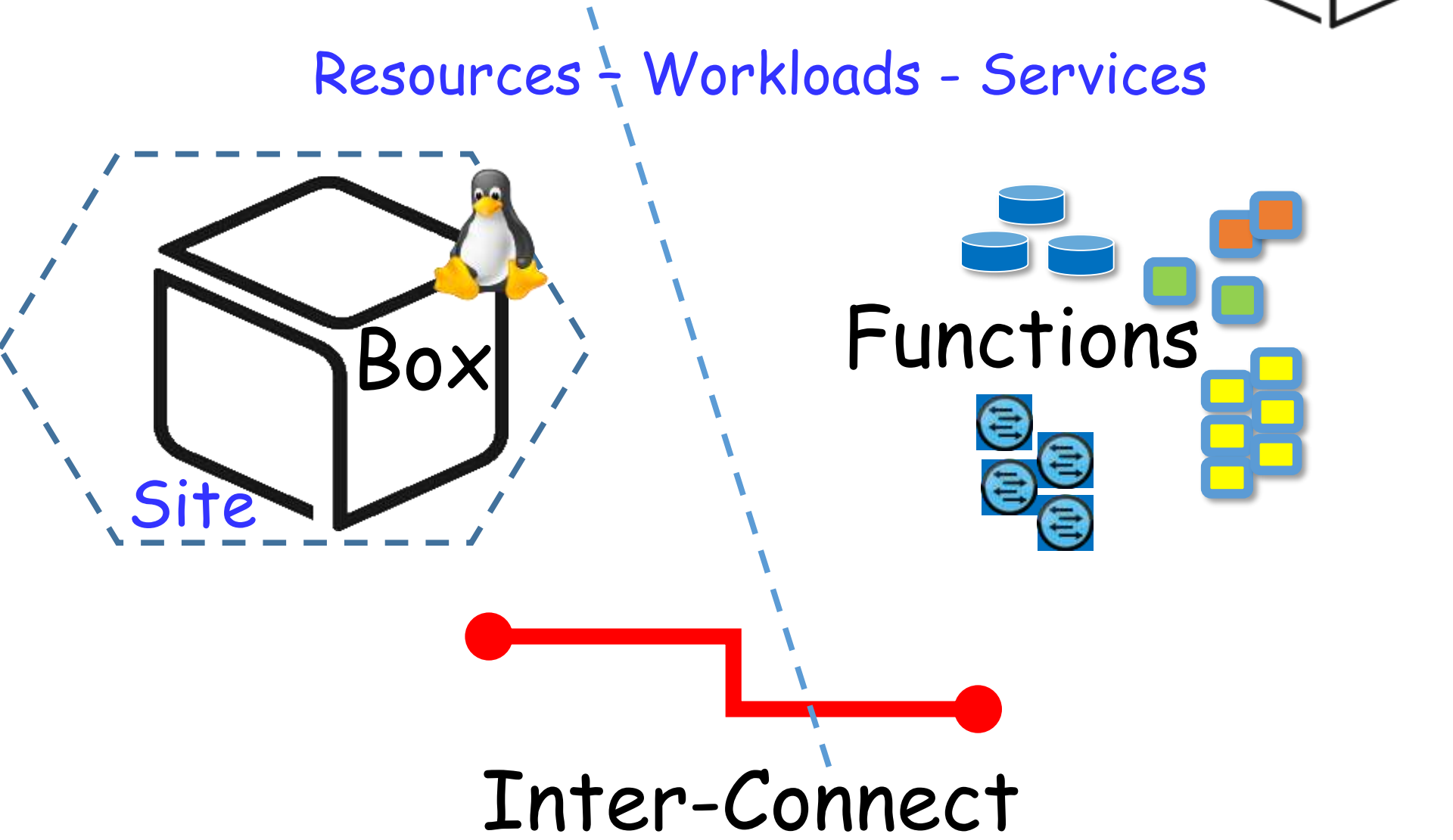


Data Inter-Connect



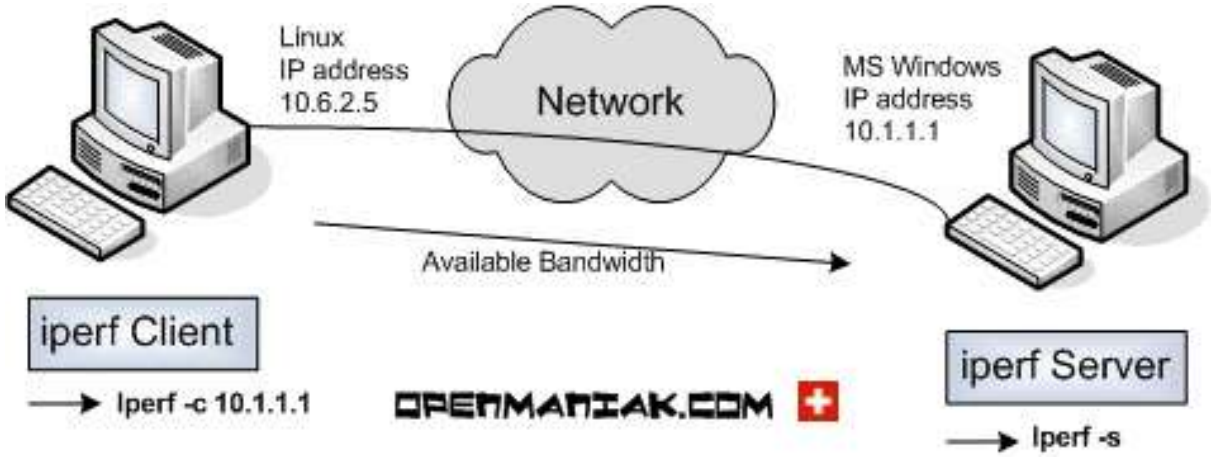
Theory

Computer System: Inter-Connected Functions inside/across Boxes/Sites



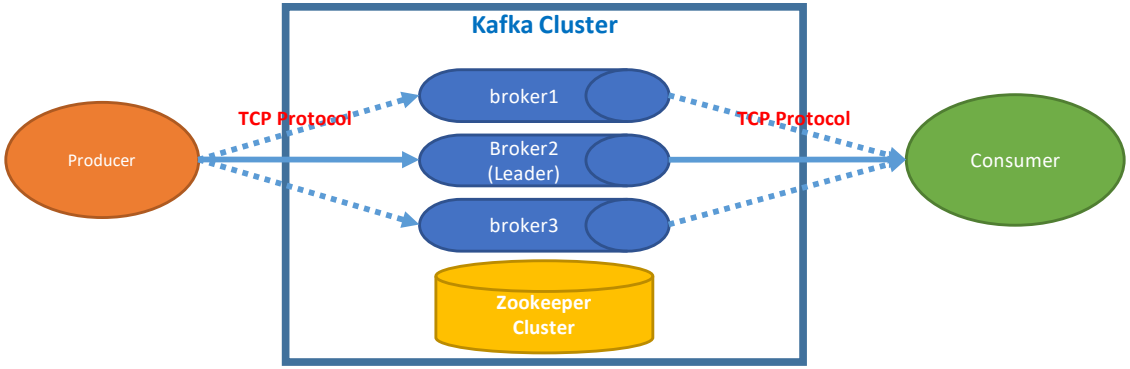
Inter-Connect Test: iperf

- Simply measures the maximum transmission capacity (bandwidth) in the wired / wireless network communication section
- Transmits the maximum traffic from Client node to the Server node and displays the result.
- Support TCP, UDP, SCTP and so on.
- Support variety of options
  - -s : iperf server
  - -c : iperf client
  - -i : check interval time
  - -t : total check time



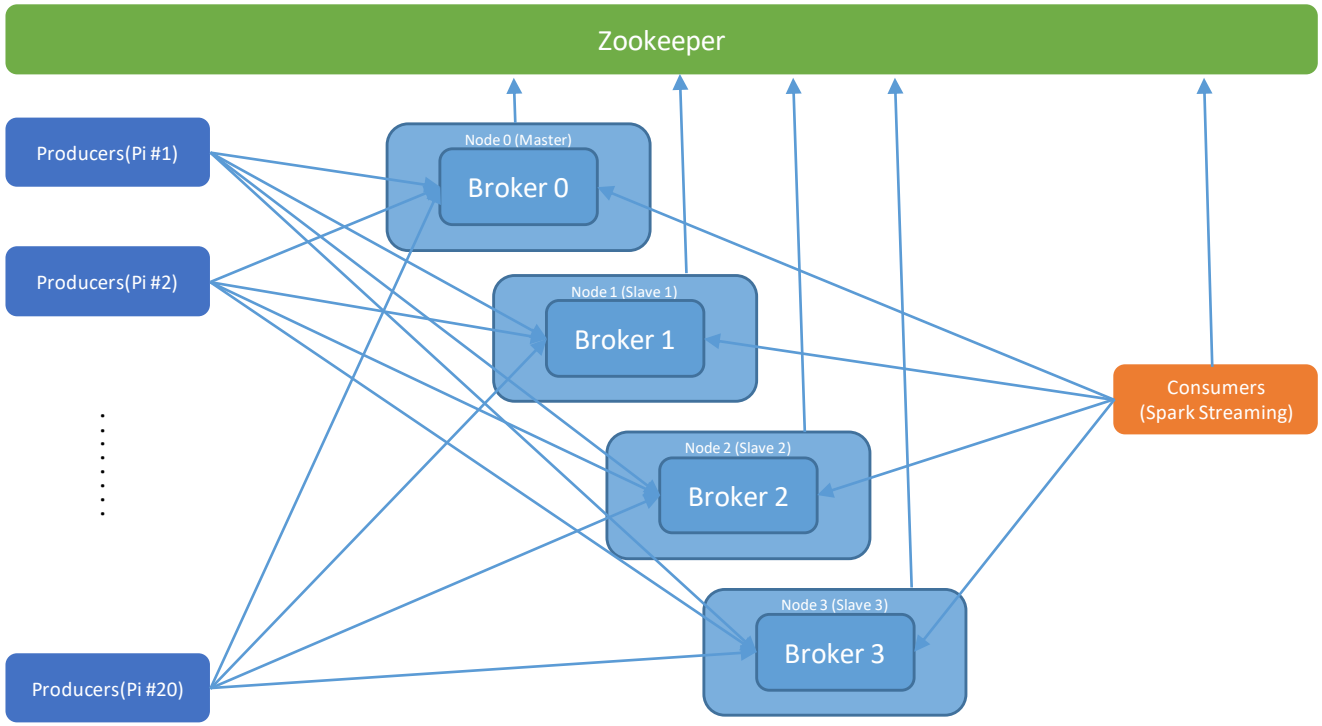
http://sfixer.tistory.com/entry/iperf-The-Easy-Tutorial

Kafka Messaging System



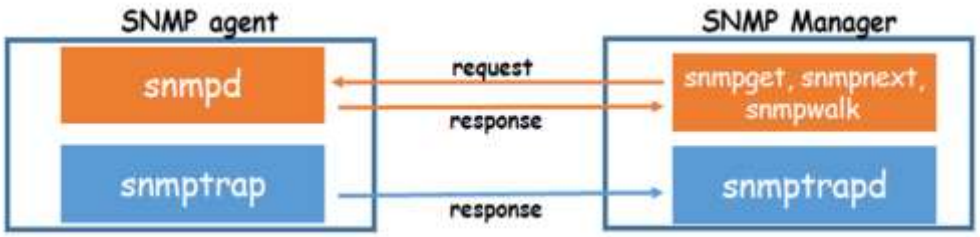
- Provides **the functionality of a messaging system**, but with a unique design

Topics: maintains feeds of messages in categories
Producers: processes that publish messages to a Kafka topic
Consumers: processes that subscribe to topics and process the feed of published messages
Broker: run as a cluster comprised of one or more servers



Net-SNMP: SNMP Agent/Manager

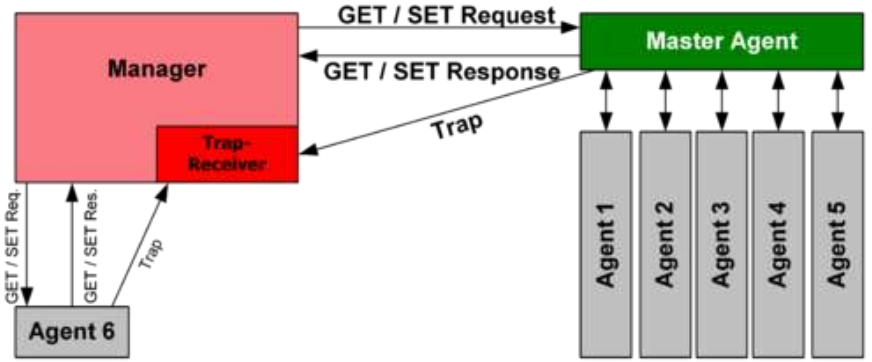
Net-SNMP A suite of software for using and deploying the SNMP Protocol



Manager : polls agents on the network, correlates and displays information
Agent : collects and stores information, responds to manager requests for information, generates traps

SNMP

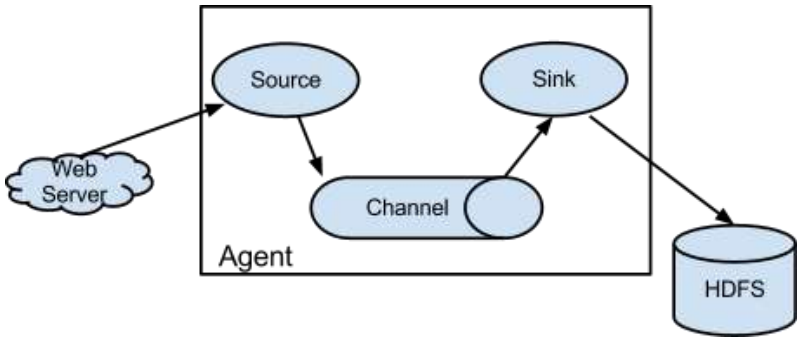
(Simple Network Management Protocol):
Used in network management systems to monitor network-attached devices, which include routers, switches, servers, workstations, printers, modem racks and more

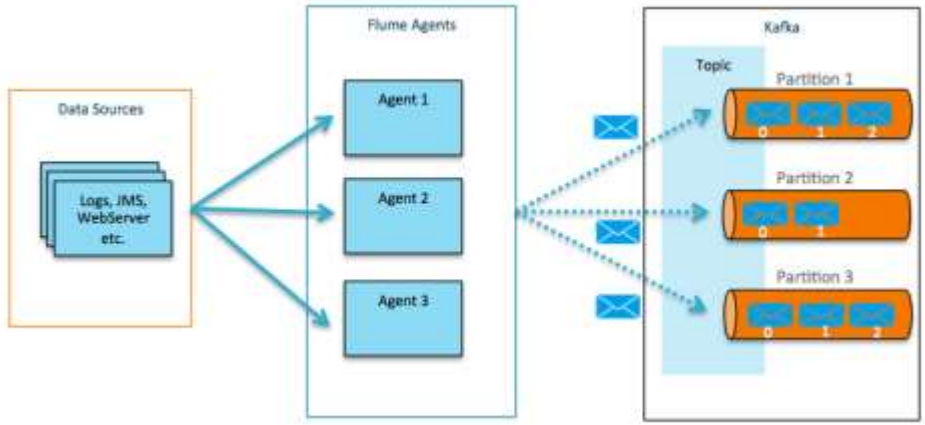


Apache Flume: Log Collector

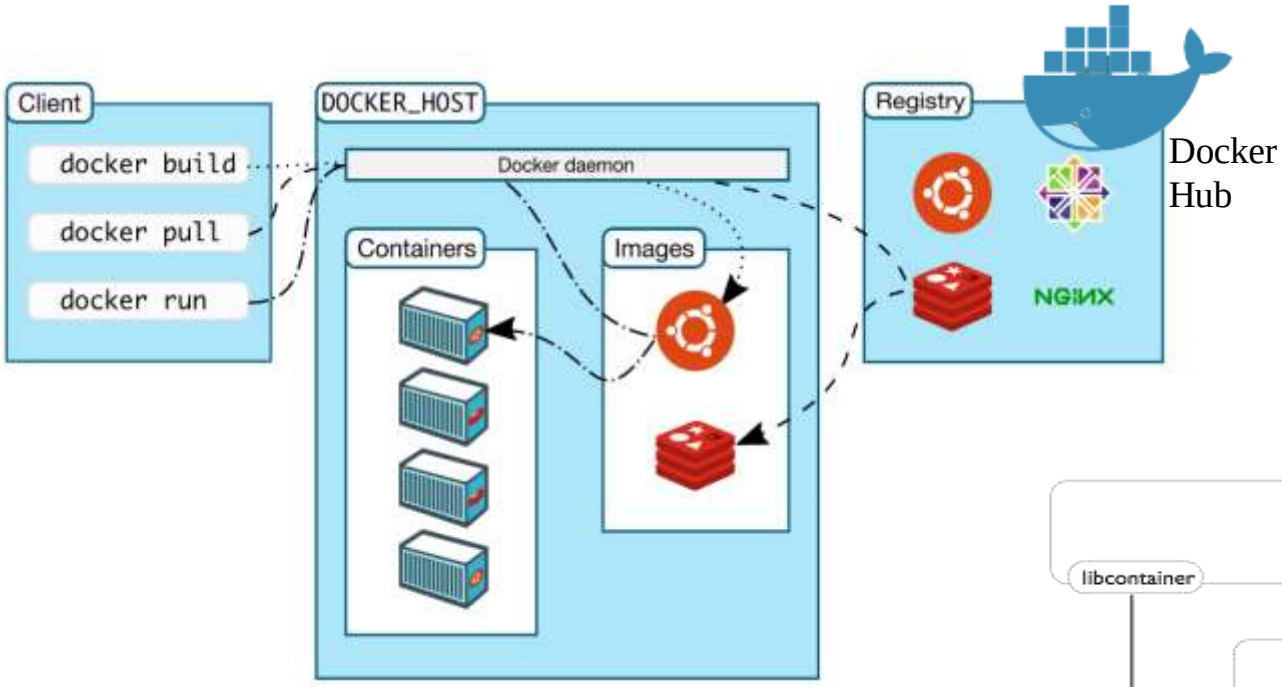
A distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data

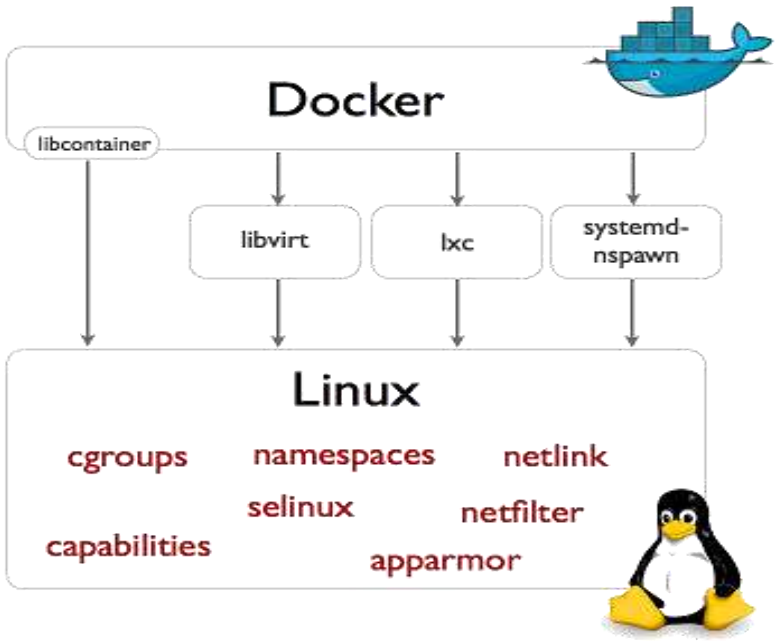
- Log aggregator with many customizable data sources, which runs asynchronously
- Flume Agent
  - Source consumes events having a specific format
  - Channel holds the event until consumed
  - Sink removes an event from the channel and puts it into on external repository or another source





Docker: Light-weight Process (Application) Container





Practice

#0 Lab Preparation (1/3)

Wired connection

NAME: Raspberry Pi Model B (Pi)
CPU: ARM Cortex A7 @900MHz
CORE: 4
Memory: 1GB
SD Card: 32GB

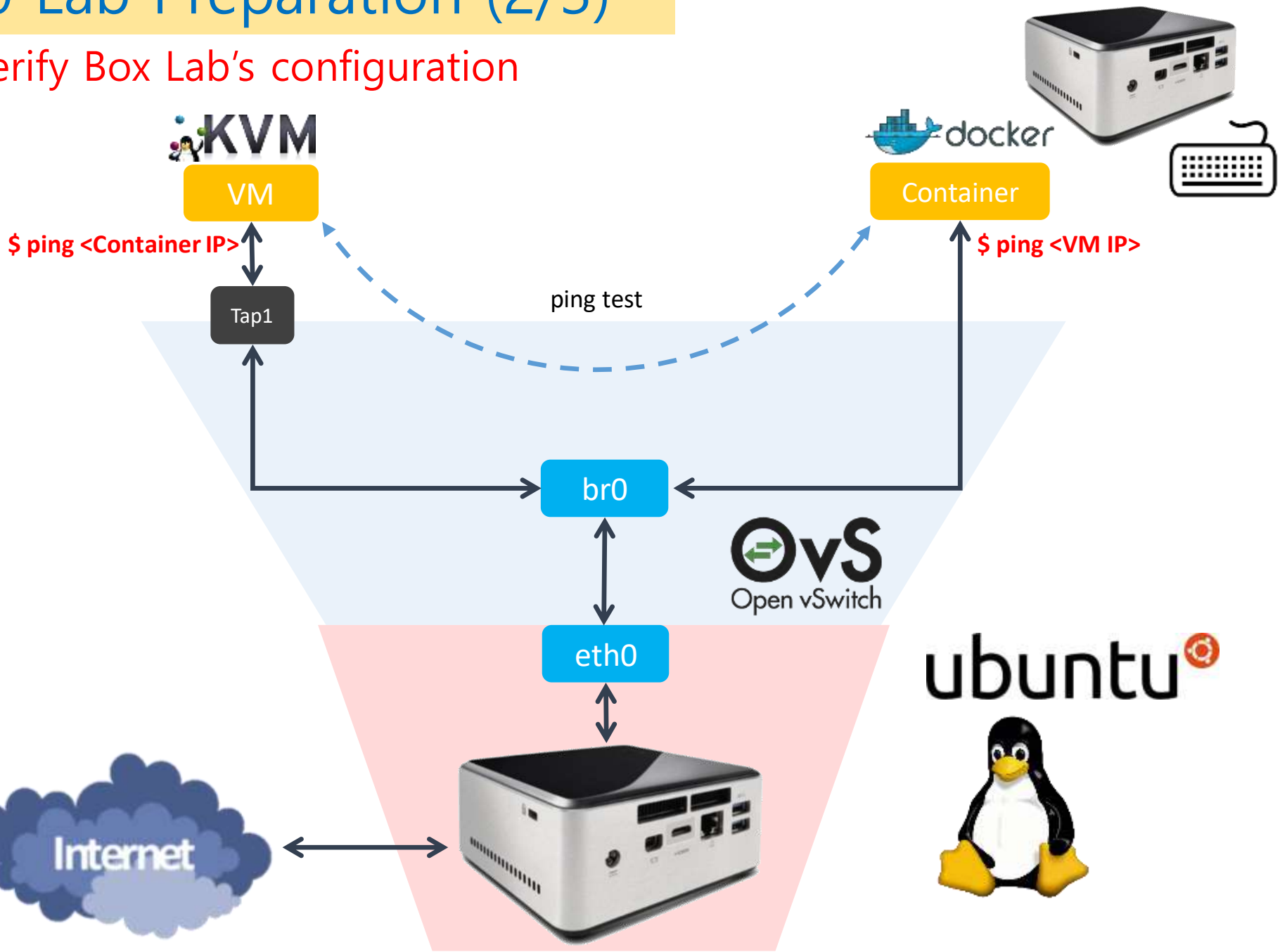
NAME: NUC5i5MYHE (NUC PC)
CPU: i5-5300U @2.30GHz
CORE: 4
Memory: 16GB DDR3
HDD: 94GB

NAME: NT900X3A
CPU: i5-2537U @1.40GHz
CORE: 2
Memory: 4GB DDR3
HDD: 128GB

NAME: netgear prosafe 16 port gigabit switch(Switch)
Network Ports: 16 auto-sensing 10/100/1000 Mbps Ethernet ports

#0 Lab Preparation (2/3)

-Verify Box Lab's configuration



#0 Lab Preparation (3/3)

- Prepare 3 terminal on NUC
  - Bare metal, VM, Container

- Running and Access to KVM on NUC (Open KVM terminal)

  $ sudo kvm -m [memory capacity] -name [name] -smp cpus=[#cpu],maxcpus= [#maxcpu] -device virtio-net-pci,netdev=net0,mac= [EE:EE:EE:EE:EE:EE] –netdev tap,id=net0,ifname= [tap_name],script=no -boot d [name].img -vnc :[#] -daemonize

  $ xvnc4viewer localhost :5

- Access to Container on NUC (Open Container terminal)

  $ docker attach [container name]

#1–1 Raspberry PI: OS Installation (1/3)

- Before we start, your Raspberry Pi must be ready with proper OS.
- In this lab, we will use “HypriotOS” Linux for it.

- Eject a MicroSD card from your Raspberry Pi, and insert it into your SD card reader and attach the reader to your NUC.
- Issue the commands below to get “flash” script for the OS setup.

```
$ sudo apt update && sudo apt install -y pv curl python-pip unzip hdparm
$ sudo pip install awscli
$ curl -O https://raw.githubusercontent.com/hypriot/flash/master/flash
$ chmod +x flash
$ sudo mv flash /usr/local/bin/flash
```

- Issue “flash” command to see if it’s installed correctly.

#1–2 Raspberry PI: OS Installation (2/3)

- Download & edit HypriotOS configuration file for your Raspberry Pi.

  $ wget -O hypriot-init.yaml https://mirror.nm.gist.ac.kr/getHypriotConf

  Let's open the "hypriot-init.yaml" file and edit its network section.

```
$ sudo vi hypriot-init.yaml

…
 # static IP configuration:
     interface eth0
     static ip_address=172.29.0.250/24 # Write your Raspberry Pi address
     static routers=172.29.0.254
     static domain_name_servers=8.8.8.8 8.8.4.4
…
```

- The assigned IP address will be automatically applied, when you're initially booting your Raspberry Pi.

#1–2 Raspberry PI: OS Installation (3/3)

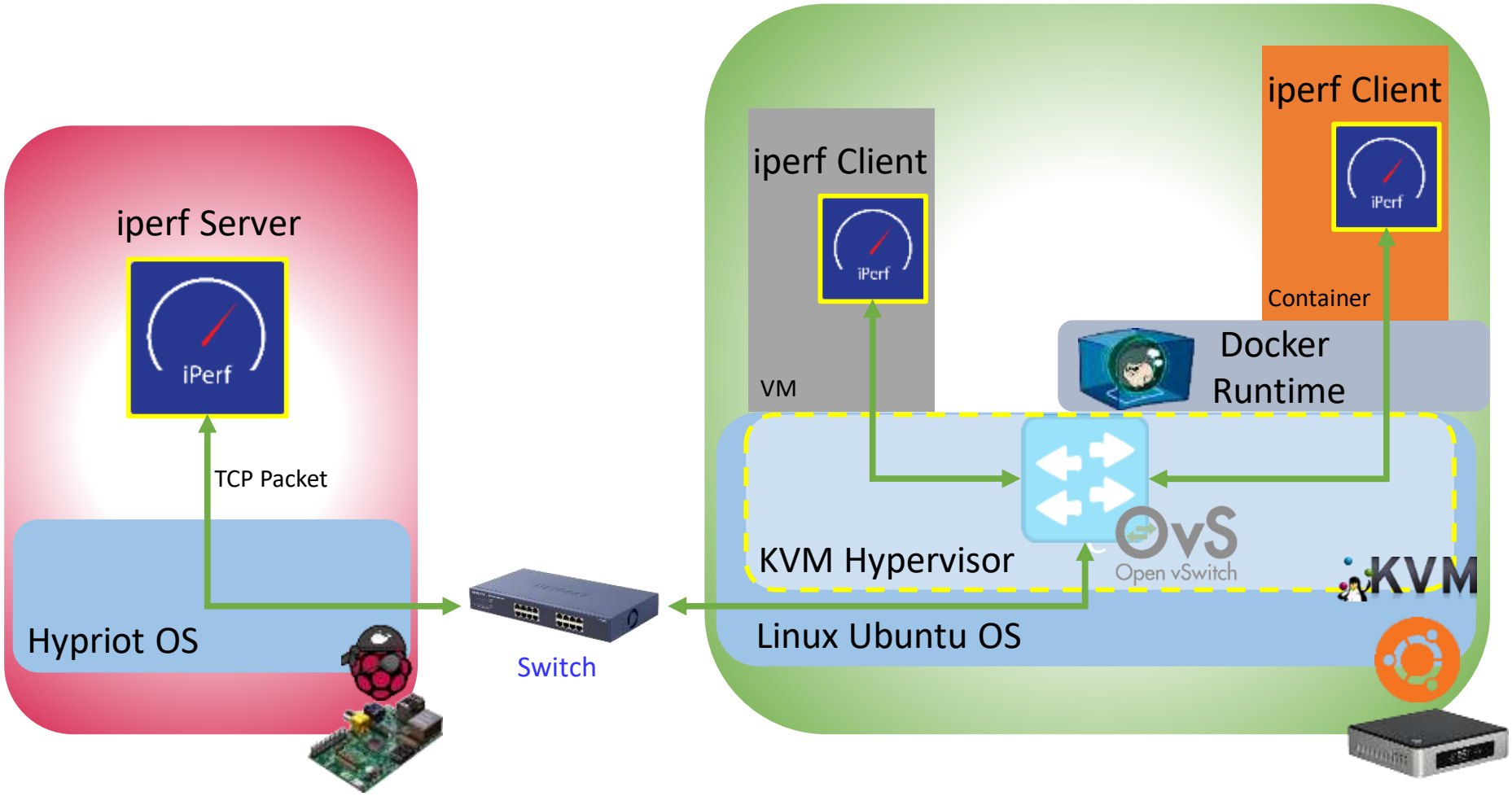
$ sudo fdisk –l

```
Disk /dev/sdc: 29.8 GiB, 32010928128 bytes, 62521344 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe6a544c8

Device     Boot  Start     End Sectors  Size Id Type
/dev/sdc1         8192  131071  122880   60M  c W95 FAT32 (LBA)
/dev/sdc2       131072 2658303 2527232  1.2G 83 Linux
```

$ flash –u hypriot-init.yaml –d /dev/sdc –f https://mirror.nm.gist.ac.kr/getHypriot

- And, that’s it! Now HypriotOS is flashed to your MicroSD card.
- Insert the SD card back to your Raspberry PI and boot it up.

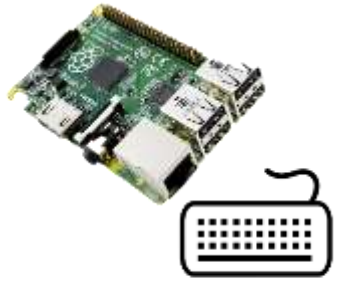
Physical Inter-Connect test



Raspberry Pi Box
for IoT Sensors
& Actuators

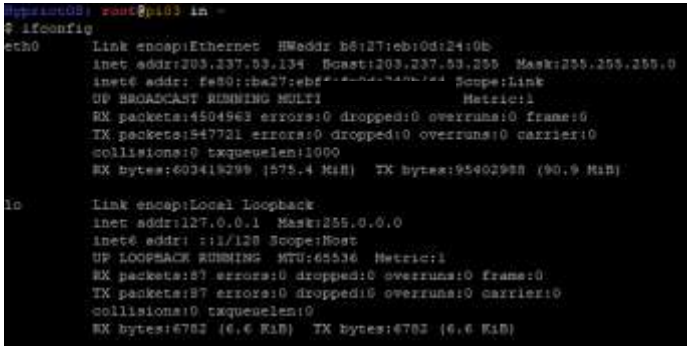
NUC Box
for IoT-Cloud Hub

#2 Network Configuration: Check Network & install packages



- Check network interface configuration

  $ ifconfig



- Check routing table and install packages

  $ netstat –rn

```
HypriotOS: root@pi03 in ~
$ netstat -rn
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
0.0.0.0         203.237.53.254  0.0.0.0         UG        0 0          0 eth0
203.237.53.0    0.0.0.0         255.255.255.0   U         0 0          0 eth0
```

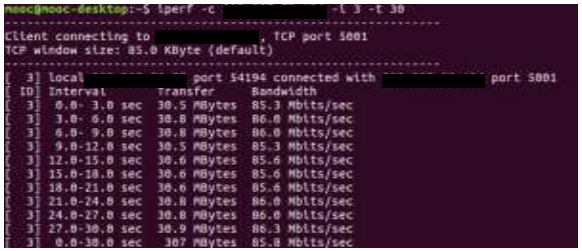
  $ sudo apt-get install openssh-server git vim

#3-1 iperf test: installation

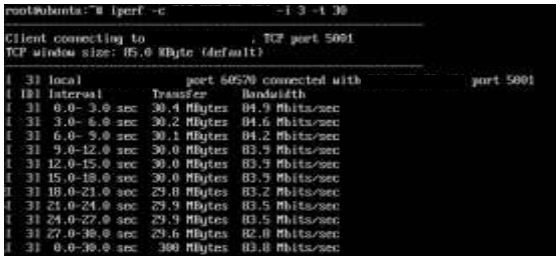
- iperf server install on Raspberry PI
  $ sudo apt-get install iperf
- iperf client install on NUC baremetal
  $ sudo apt-get install iperf
- iperf client install on NUC KVM
  $ sudo apt-get install iperf
- iperf client install on NUC Container
  $ sudo apt-get install iperf

#3–2 iperf test: Execute iperf

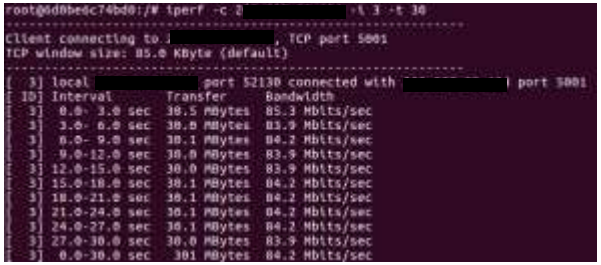
- Execute iperf Server on Raspberry PI
  - $ iperf –s
  - Iperf server must execute before you execute client
- Execute iperf client on NUC baremetal, VM, and Container
  - $ iperf –c <Raspberry PI IP address> -i 1 –t 30
  - (Do not execute at the same time)



<iperf result (baremetal)>



<iperf result (VM)>



<iperf result (Container)>

#3–3 iperf test: Check Result

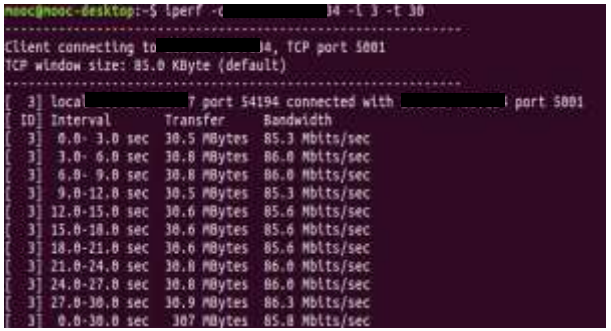

```
^CHypriotOS: root@pi03 in ~
$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  4] local                 port 5001 connected with              port 54194
[ ID] Interval       Transfer     Bandwidth
[  4]  0.0-30.0 sec   307 MBytes  85.7 Mbits/sec
[  5] local                 port 5001 connected with              port 60570
[  5]  0.0-30.0 sec   300 MBytes  83.8 Mbits/sec
[  4] local                 port 5001 connected with              port 52130
[  4]  0.0-30.0 sec   301 MBytes  84.2 Mbits/sec
```

Baremetal Result

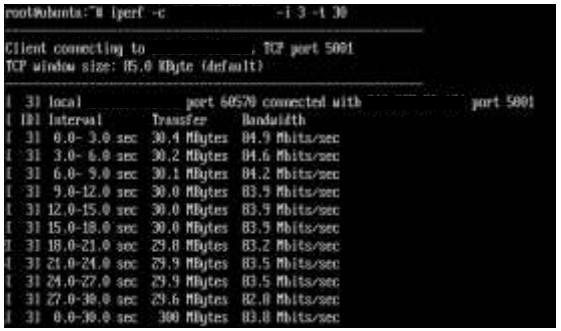
VM Result

Container Result

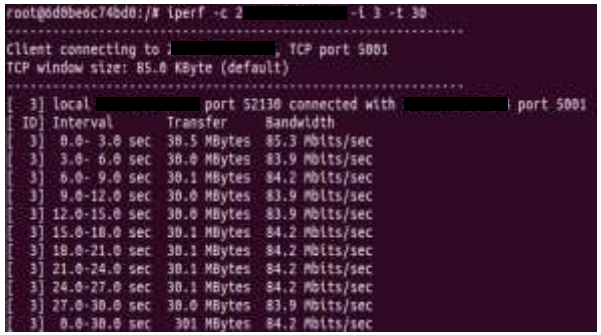
<iperf Server result>



<iperf result (baremetal)>



<iperf result (VM)>



<iperf result (Container)>

#4–1 Hostname preparation for Kafka (1/2)

- Every machine which communicate with themselves must know their own address.

- Edit /etc/hosts

$ sudo vi /etc/hosts

(For Example)

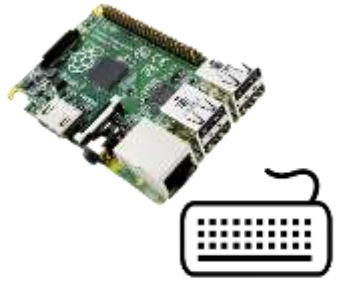
```
127.0.0.1       localhost
127.0.1.1       [redacted]

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

203.237.53.[redacted]  nuc
203.237.53.[redacted]  pi
```

Add two lines which describe the IP address and hostname of devices

IP address

hostname

#4–1 Hostname preparation for Kafka (2/2)



- Every machine which communicate with themselves must know their own address.

- SSH access to your PI (ID: pirate, PW: hypriot)

  $ ssh pirate@<your Raspberry PI IP Address>

- Edit /etc/hosts

  $ sudo vi /etc/hosts

(For Example)

```
127.0.0.1       localhost
127.0.1.1       [redacted]

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters

203.237.53.[redacted] nuc
203.237.53.[redacted] pi
```

Add two lines which describe the IP address and hostname of devices

#4–2 Verification for hostname preparation

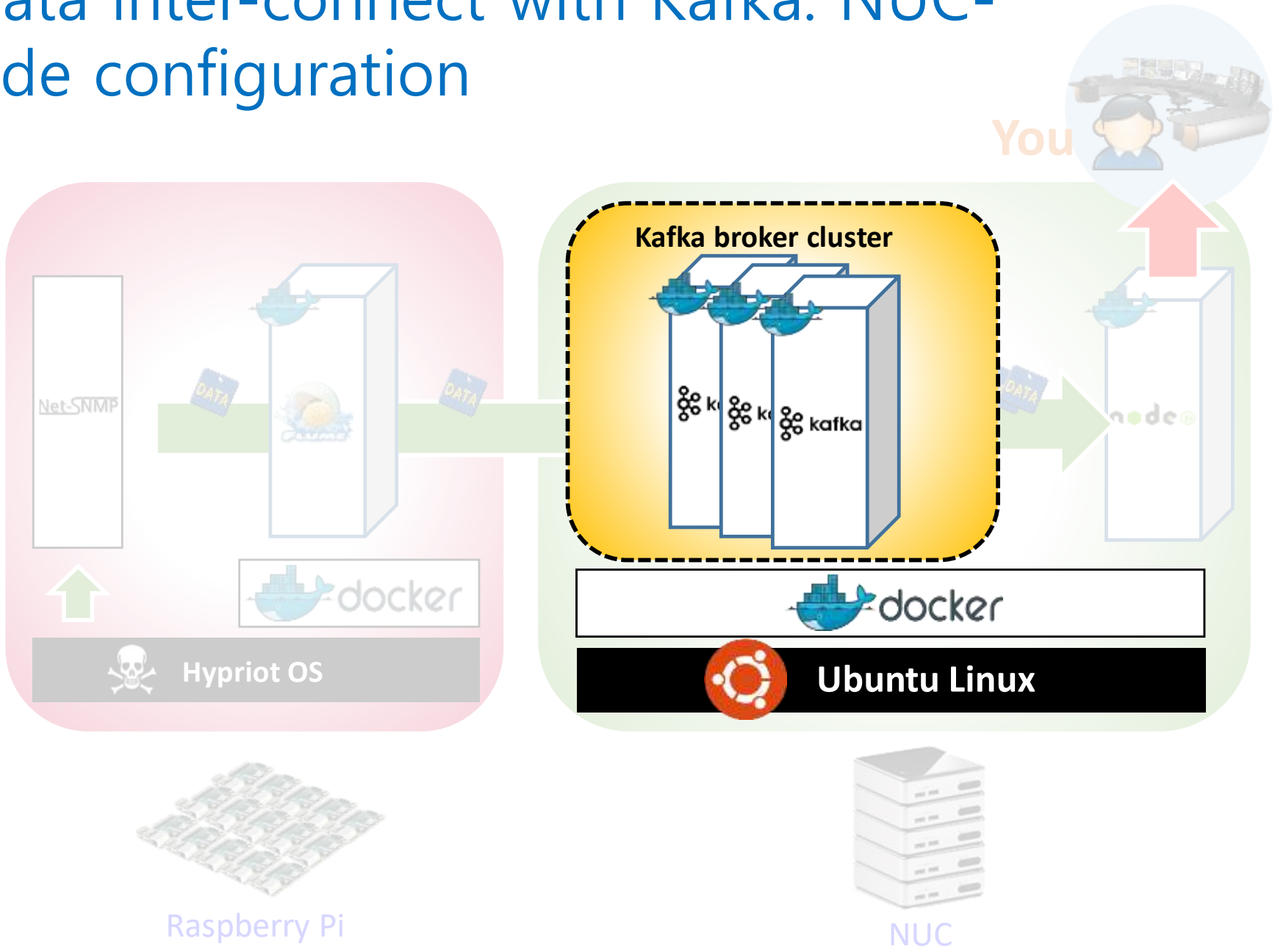
- After editing /etc/hosts, check the edit is correctly done

- For NUC
  - $ ping <Your NUC hostname>
  - $ ping <Your Raspberry PI hostname>

- For Raspberry PI
  - $ ping <Your NUC hostname>
  - $ ping <Your Raspberry PI hostname>

If it was successful, We can be sure that
NUC know its own hostname and Pi's hostname and Pi also know its own hostname and NUC's hostname.

Data inter-connect with Kafka: NUC-side configuration



#5–1 Kafka deployment: Docker file download

- Download all files from Github

  (http://github.com/SmartXBox/SmartX-mini)

  $ git clone https://github.com/SmartXBox/SmartX-mini.git

- Folder List

  - raspbian-flume
  - ubuntu-flume
  - ubuntu-influx
  - ubuntu-kafka
  - ubuntu-kafkatodb

In this section, we use this

#5–1 Download Files from Github

- Allocate Broker IDs and Ports

1. We'll use a one zookeeper, 3 brokers and one consumer containers which share host's public IP address
2. Zookeeper container doesn't have broker id.
3. Each Broker has a unique id and port to interact each other.
4. Consumer container just used to manage topic and check the data from brokers.

Function(Container) Name	IP address	Broker id	Listening port
Zookeeper	Host's public IP address	-	2181
Kafka broker0		0	9090
Kafka broker1		1	9091
Kafka broker2		2	9092
Kafka consumer		-	-

#5–2 Kafka deployment: Build Docker Image

- Build Docker Image

  $ cd ~/SmartX-mini/ubuntu-kafka

- Build Dockerfile※ It takes long time. You should type '.' !

  $ docker build --tag ubuntu-kafka .

- If you want to check Docker instruction words

  $ docker --help

ex) docker ps : List containers

docker start : Start one or more stopped containers

docker rm : Remove one or more containers

#5–3 Kafka deployment: Place Docker Containers

(Recommend open new terminal window)

- Run Docker Container

  $ docker run -it --net=host --name [container name] ubuntu-kafka

- We need to run 5 containers (zookeeper 1, broker 3, consumer 1)
- Let's assume the name of each containers,

  zookeeper, broker0, broker1, broker2, consumer
- Repeatedly type the above command with changing container name
- If you want to look for more details about Docker command, see https://docs.docker.com/reference/commandline/

#6–1 Kafka Containers: Zookeeper configuration

✓ Actually we use default configuration

1. Open zookeeper properties file

   $ vi config/zookeeper.properties

2. Check the client port

```
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
# the directory where the snapshot is stored.
dataDir=/tmp/zookeeper
# the port at which the clients will connect
clientPort=2181
# disable the per-ip limit on the number of connections since this is a non-production config
maxClientCnxns=0
```

#6–2 Kafka Containers: Zookeeper executing

- ✓ zookeeper must execute first

  $ bin/zookeeper-server-start.sh config/zookeeper.properties

  (Leave Zookeeper running and open a new terminal for next tasks)

```
[2015-11-20 04:13:18,607] INFO Server environment:java.library.path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib (o
[2015-11-20 04:13:18,607] INFO Server environment:java.io.tmpdir=/tmp (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,607] INFO Server environment:java.compiler=<NA> (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,607] INFO Server environment:os.name=Linux (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,607] INFO Server environment:os.arch=amd64 (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,607] INFO Server environment:os.version=3.19.0-25-generic (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,607] INFO Server environment:user.name=root (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,607] INFO Server environment:user.home=/root (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,608] INFO Server environment:user.dir=/kafka (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,614] INFO tickTime set to 3000 (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,614] INFO minSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,614] INFO maxSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2015-11-20 04:13:18,625] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2015-11-20 04:13:19,034] INFO Accepted socket connection from /210.125.84.69:48648 (org.apache.zookeeper.server.NIOServerCnxnFacto
[2015-11-20 04:13:19,135] INFO Client attempting to renew session 0x15122d708dd000c at /210.125.84.69:48648 (org.apache.zookeeper.s
[2015-11-20 04:13:19,142] INFO Established session 0x15122d708dd000c with negotiated timeout 6000 for client /210.125.84.69:48648 (
[2015-11-20 04:13:19,632] INFO Accepted socket connection from /210.125.84.69:48649 (org.apache.zookeeper.server.NIOServerCnxnFacto
[2015-11-20 04:13:19,632] INFO Client attempting to renew session 0x15122d708dd000b at /210.125.84.69:48649 (org.apache.zookeeper.s
[2015-11-20 04:13:19,633] INFO Established session 0x15122d708dd000b with negotiated timeout 30000 for client /210.125.84.69:48649
```

#6–3 Kafka Containers: Kafka Broker configuration

- Create a Kafka container with the docker command before

  $ docker run -it --net=host --name [container name] ubuntu-kafka

- Open server properties file and change proper broker id and port (they must be unique to each other) (Only for broker0,1,2)

  $ vi config/server.properties

```
############################# Server Basics #

# The id of the broker. This must be set to a
broker.id=0
```
broker id

```
############################# Socket Server S

# The port the socket server listens on
port=9092
```
port

Container Name	Broker id	Listening port
broker0	0	9090
broker1	1	9091
broker2	2	9092
consumer	-	-

Consumer container will not run any brokers

This is the code associated with the client port we just checked.

```
############################# Zookeeper #############################

# Zookeeper connection string (see zookeeper docs for details).
# This is a comma separated host:port pairs, each corresponding to a zk
# server. e.g. "127.0.0.1:3000,127.0.0.1:3001,127.0.0.1:3002".
# You can also append an optional chroot string to the urls to specify the
# root directory for all kafka znodes.
zookeeper.connect=localhost:2181

# Timeout in ms for connecting to zookeeper
zookeeper.connection.timeout.ms=6000
```

#6–4 Kafka Containers: Kafka Broker executing

- Execute Kafka brokers (Only for broker0,1,2)

  $ bin/kafka-server-start.sh config/server.properties

- Repeat previous steps for broker0, broker1, broker2, consumer

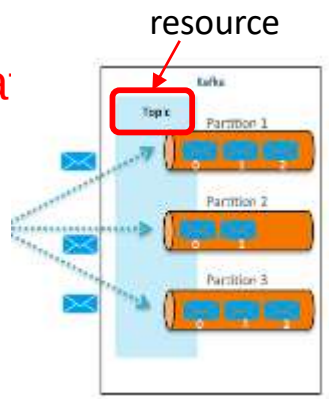
✓ When it successfully works, each broker containers will show messages like the below

```
INFO Logs loading complete. (kafka.log.LogManager)
INFO Starting log cleanup with a period of 300000 ms. (kafka.log.LogManager)
INFO Starting log flusher with a default period of 9223372036854775807 ms. (kafka.log.LogManager)
INFO Awaiting socket connections on 0.0.0.0:9092. (kafka.network.Acceptor)
INFO [Socket Server on Broker 0], Started (kafka.network.SocketServer)
INFO Will not load MX4J, mx4j-tools.jar is not in the classpath (kafka.utils.Mx4jLoader$)
INFO 0 successfully elected as leader (kafka.server.ZookeeperLeaderElector)
INFO New leader is 0 (kafka.server.ZookeeperLeaderElector$LeaderChangeListener)
INFO Registered broker 0 at path /brokers/ids/0 with address broker1:9092. (kafka.utils.ZkUtils$)
INFO [Kafka Server 0], started (kafka.server.KafkaServer)
```

#6–5 Kafka Containers: Make Topic

- Create topic

  $ bin/kafka-topics.sh --create --zookeeper localhost:2181 --replica
  factor 1 --partitions 3 --topic resource



- We can check topics.

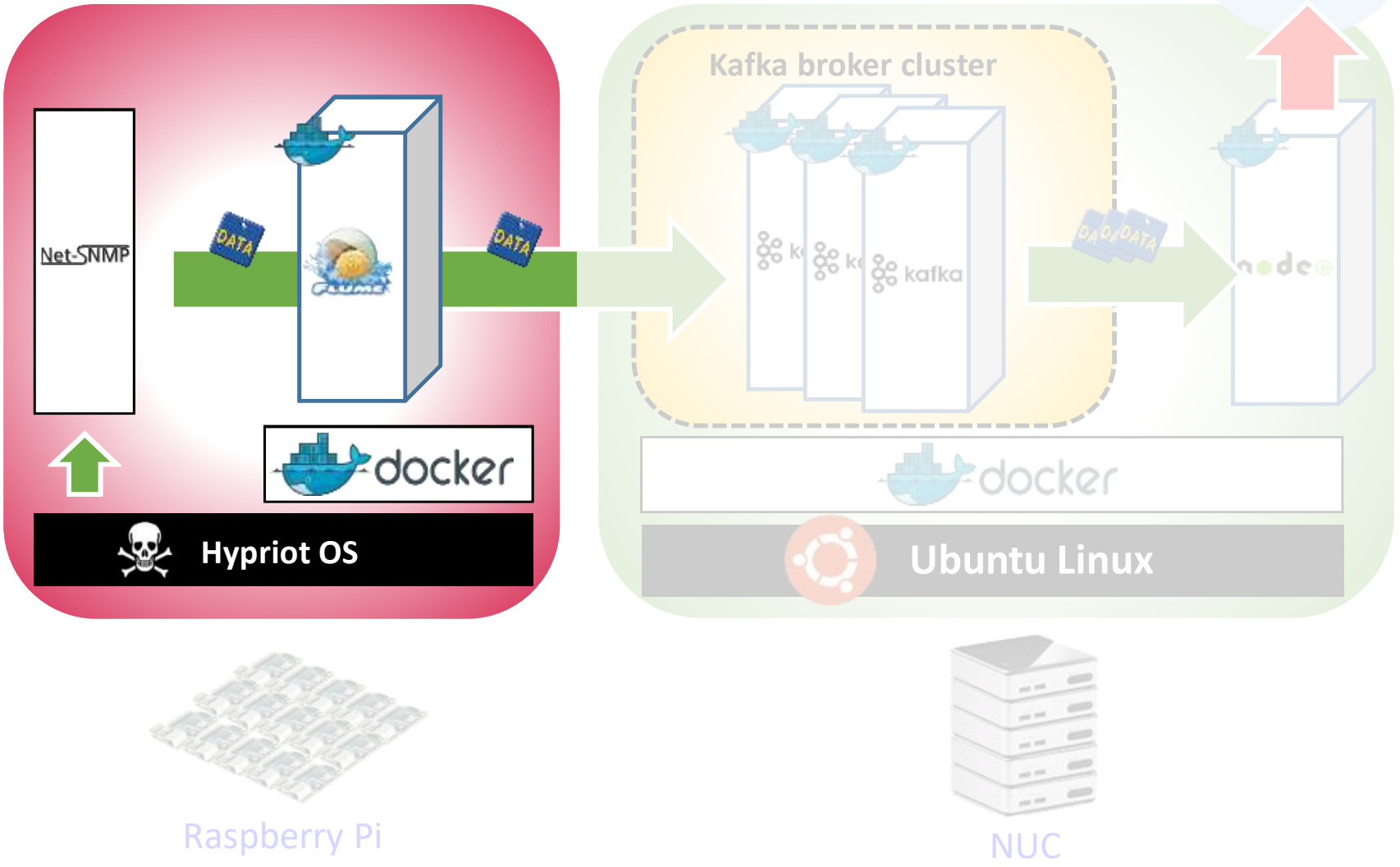
  Topic List

  $ bin/kafka-topics.sh --list --zookeeper localhost:2181

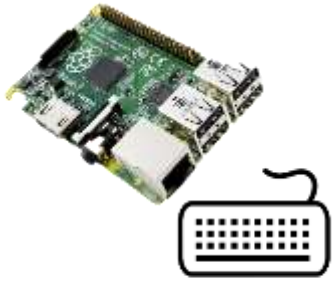
  Topic specification

  $ bin/kafka-topics.sh --describe --zookeeper localhost:2181 --topic
  resource

Data inter-connect with Kafka: PI-side configuration



#7–1 Flume on Raspberry PI: File download



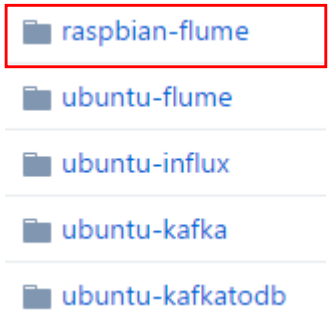
- Git package is already installed in Hypriot OS

- Download all files from Github

  (http://github.com/SmartXBox/SmartX-mini)

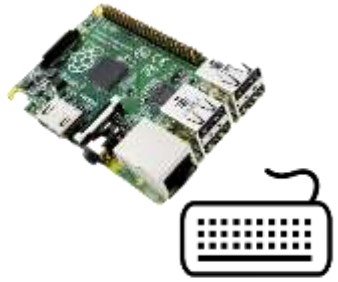
  - $ git clone https://github.com/SmartXBox/SmartX-mini.git

- Folder List



In this section, we use this file

#7–2 Flume on Raspberry PI: Net-SNMP installation



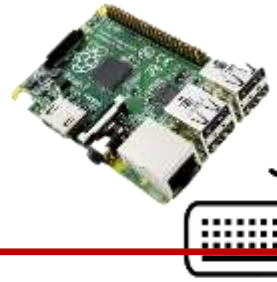
- Update packages
  $ sudo apt update

- Download Net-SNMP
  $ sudo apt install –y snmp snmpd snmp-mibs-downloader

- Download MIBs
  $ sudo download-mibs

- Modify configuration file
  $ sudo vi /etc/snmp/snmpd.conf
  #rocommunity public localhost -> Delete #
  $ sudo systemctl restart snmpd.service

#7–3 Flume on Raspberry PI: Flume installation



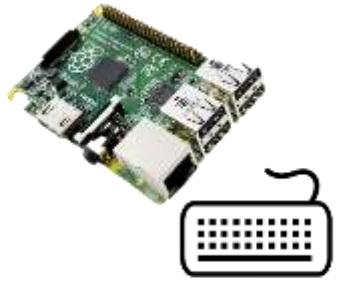
- Build Dockerfile ※ It takes long time. You should type '.'!
  - $ cd SmartX-mini/raspbian-flume
  - $ vi Dockerfile
    - modify resin/rpi-raspbian:wheezy → balenalib/rpi-raspbian [1st line]
    - modify iproute → iproute2 [6th line]
  - $ docker build --tag raspbian-flume .
  - $ docker run -it --net=host --name flume raspbian-flume

- Check the configuration file
  - $ vi conf/flume-conf.properties

- Modifying broker list
  - Default value sets "nuc"
  - Edit them into your own nuc's hostname

```
# The sink1
agent.sinks.sink1.type = org.apache.flume.sink.kafka.KafkaSink
agent.sinks.sink1.topic = resource
agent.sinks.sink1.brokerList = nuc:9091,nuc:9092,nuc:9093
agent.sinks.sink1.requiredAcks = 1
agent.sinks.sink1.batchSize = 1
```

#7–4 Flume on Raspberry PI: Executing Flume agent



- Run Flume on RPi

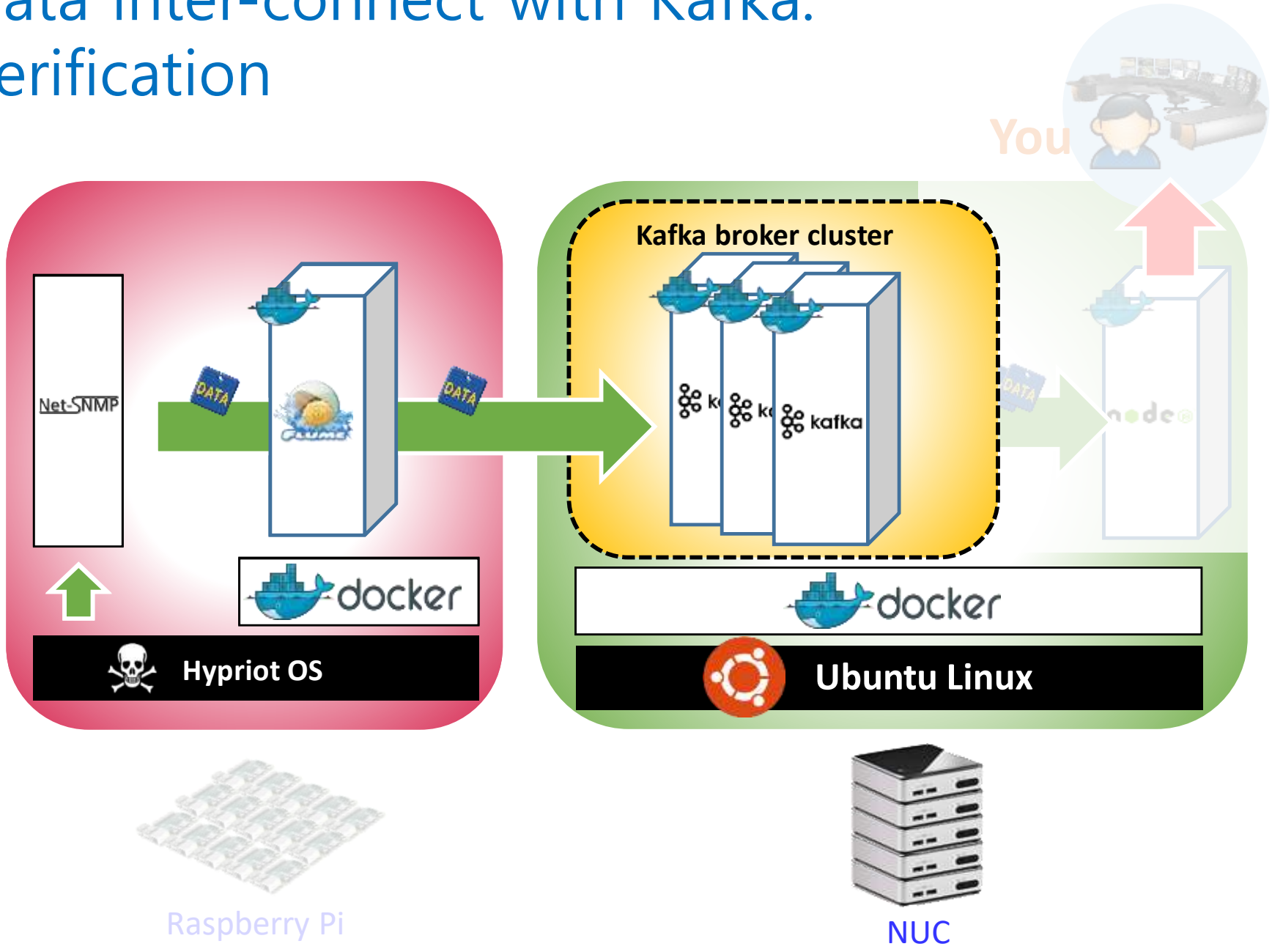
  $ bin/flume-ng agent --conf conf --conf-file conf/flume-conf.properties --name agent -Dflume.root.logger=INFO,console

```
root@black-pearl:/flume# bin/flume-ng agent --conf conf --conf-file conf/flume-conf.propert
ies --name agent -Dflume.root.logger=INFO,console
```

If an error occurs, check the host of pi again.

When pi is rebooted, the information in /etc/hosts disappears.

Data inter-connect with Kafka: Verification



#8 Consume message from brokers

- Launch consumer script on the **Consumer container**

  $ bin/kafka-console-consumer.sh --zookeeper localhost:2181 --topic resource --from-beginning

1 zookeeper | 2 broker1 | 3 broker2 | 4 broker3 | **5 consumer** | 6 pi01

```
1447989025957,172.17.42.1,0,0.06,12.00,82120,0,163164,506512,98,76170,0,0,673468,8
1447989026969,172.17.42.1,0,0.06,12.00,82120,0,163164,506512,98,76170,0,0,673468,8
1447989027986,172.17.42.1,0,0.06,12.00,82152,0,163168,506516,98,76170,0,0,673468,8
1447989029004,172.17.42.1,0,0.06,12.00,82152,0,163168,506516,98,76170,0,0,673468,8
1447989030019,172.17.42.1,0,0.06,12.00,82152,0,163168,506516,98,76170,0,0,673468,8
1447989031031,172.17.42.1,0,0.06,12.00,82152,0,163168,506516,98,76170,0,0,673468,8
1447989032042,172.17.42.1,0,0.06,12.00,82152,0,163168,506516,98,76170,0,0,673468,8
1447989033054,172.17.42.1,0,0.06,12.00,82152,0,163172,506516,98,76170,0,0,673468,8
1447989034067,172.17.42.1,0,0.06,12.00,82152,0,163172,506516,98,76170,0,0,673468,8
1447989035081,172.17.42.1,0,0.06,12.00,82152,0,163172,506516,98,76170,0,0,673468,8
1447989036094,172.17.42.1,0,0.06,12.00,82152,0,163172,506516,98,76170,0,0,673468,8
1447989037106,172.17.42.1,0,0.06,12.00,82152,0,163172,506516,98,76170,0,0,673468,8
1447989038119,172.17.42.1,0,0.06,12.00,82120,0,163180,506520,98,76171,0,0,673468,8
1447989039131,172.17.42.1,0,0.06,12.00,82120,0,163180,506520,98,76171,0,0,673468,8
1447989040142,172.17.42.1,0,0.06,12.00,82120,0,163180,506520,98,76171,0,0,673468,8
1447989041156,172.17.42.1,0,0.06,12.00,82120,0,163180,506520,98,76171,0,0,673468,8
```

Review

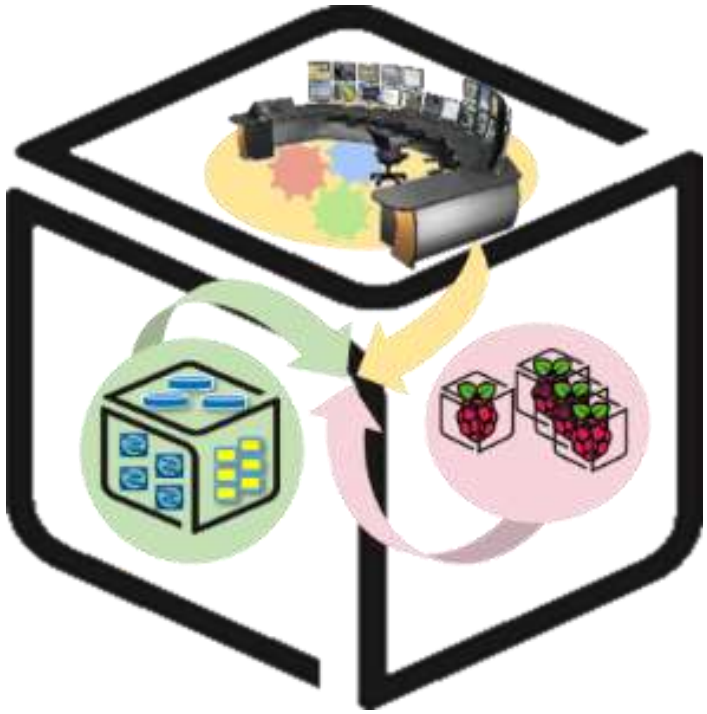
Lab Summary

With Inter-Connect Lab, you have experimented

1. How to **physically inter-connect** two kinds of Boxes (NUC and Raspberry PI)
2. How to **inter-connect data transfer** (via Kafka messaging) between functions located in different boxes

Differentiation between two types of “Inter-connect”: You need to distinguish physical Inter-connect from data Inter-connect!

Thank You for Your Attention
Any Questions?



mini@smartx.kr