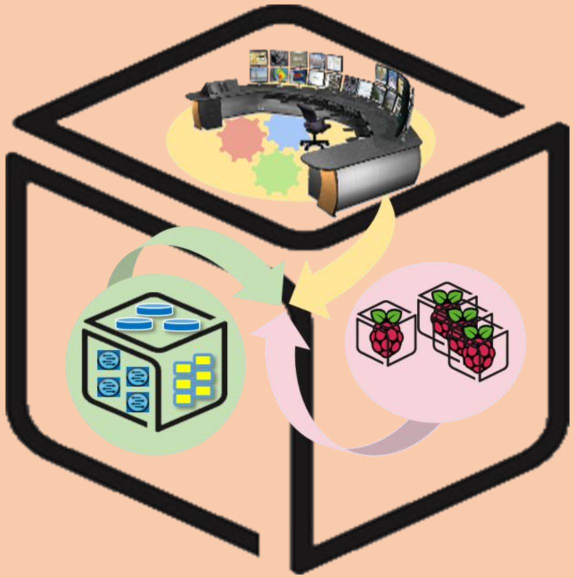
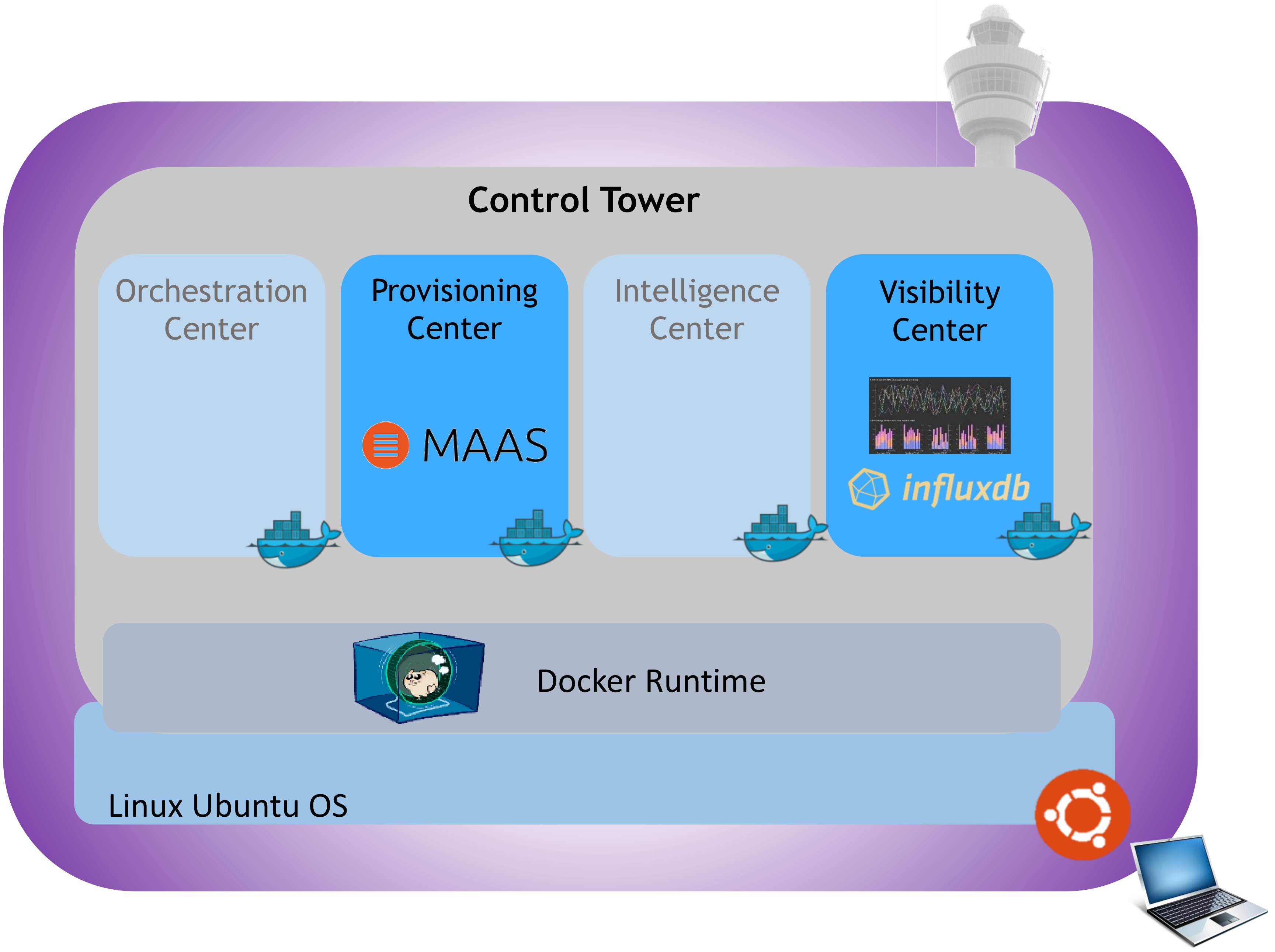
Computer Systems For AI-inspired Cloud Theory & Lab.

Lab #3.5: Tower 2

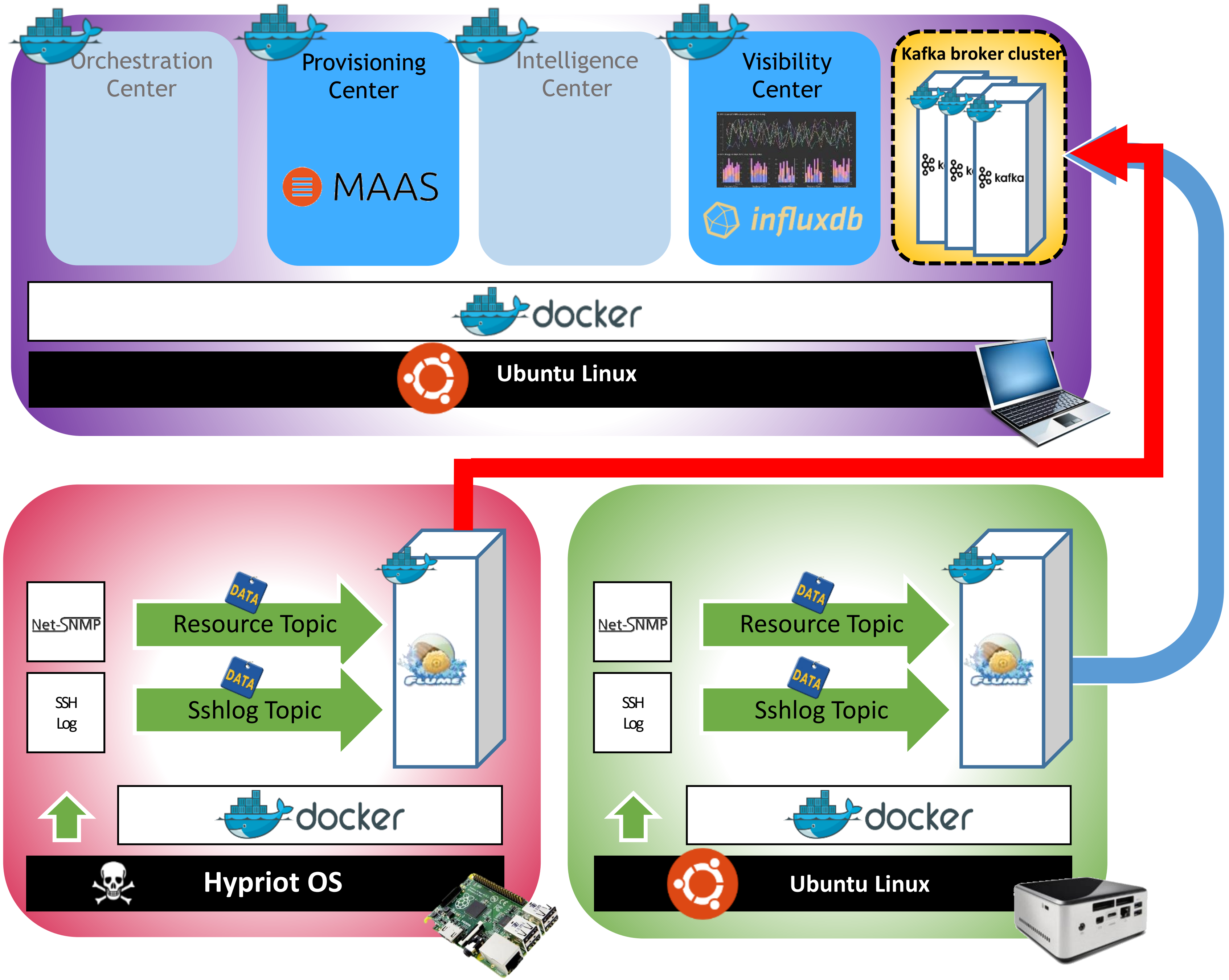


https://github.com/SmartX-Labs/SmartX-Mini-MOOC

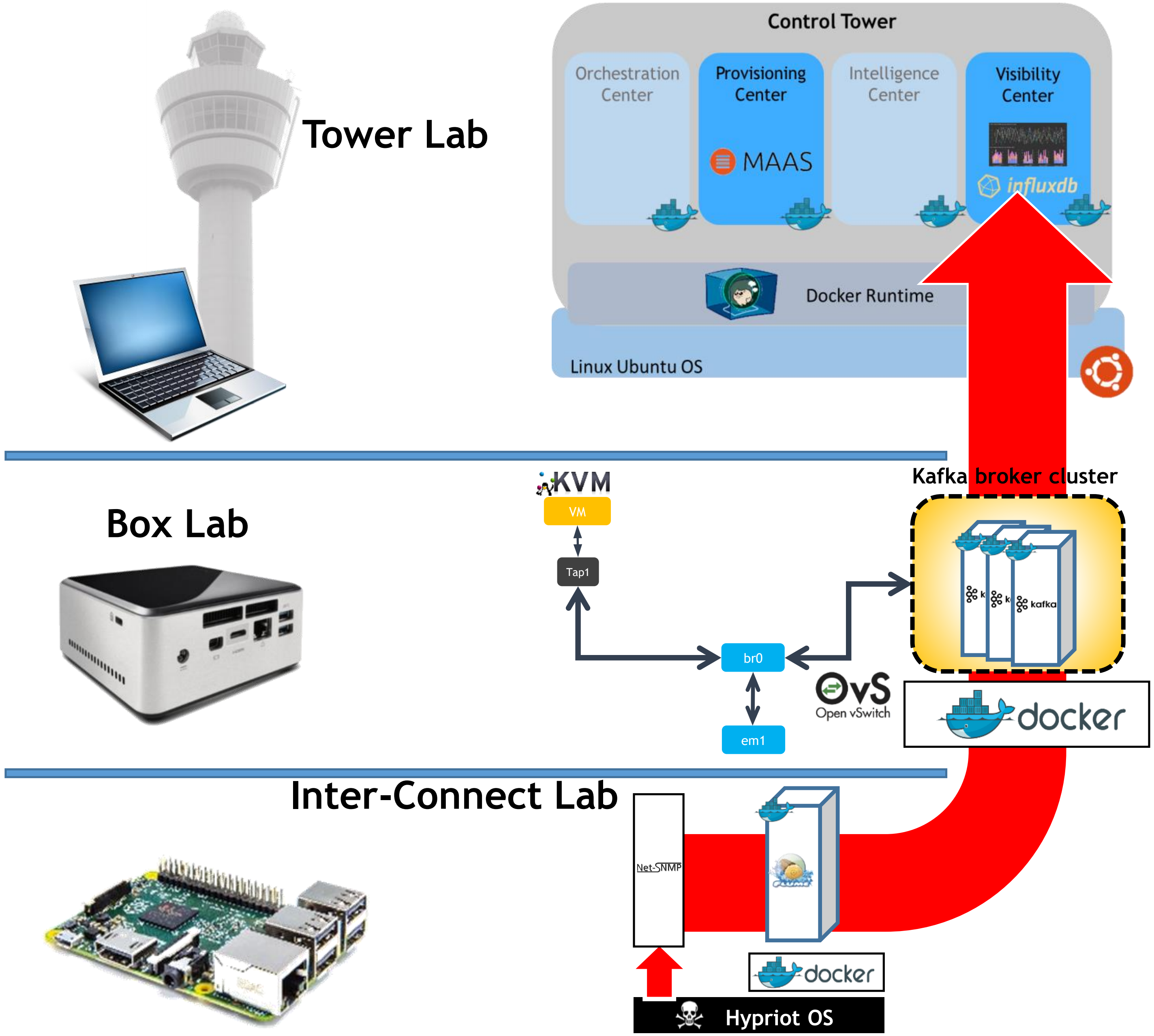
Tower Lab 2: Concept



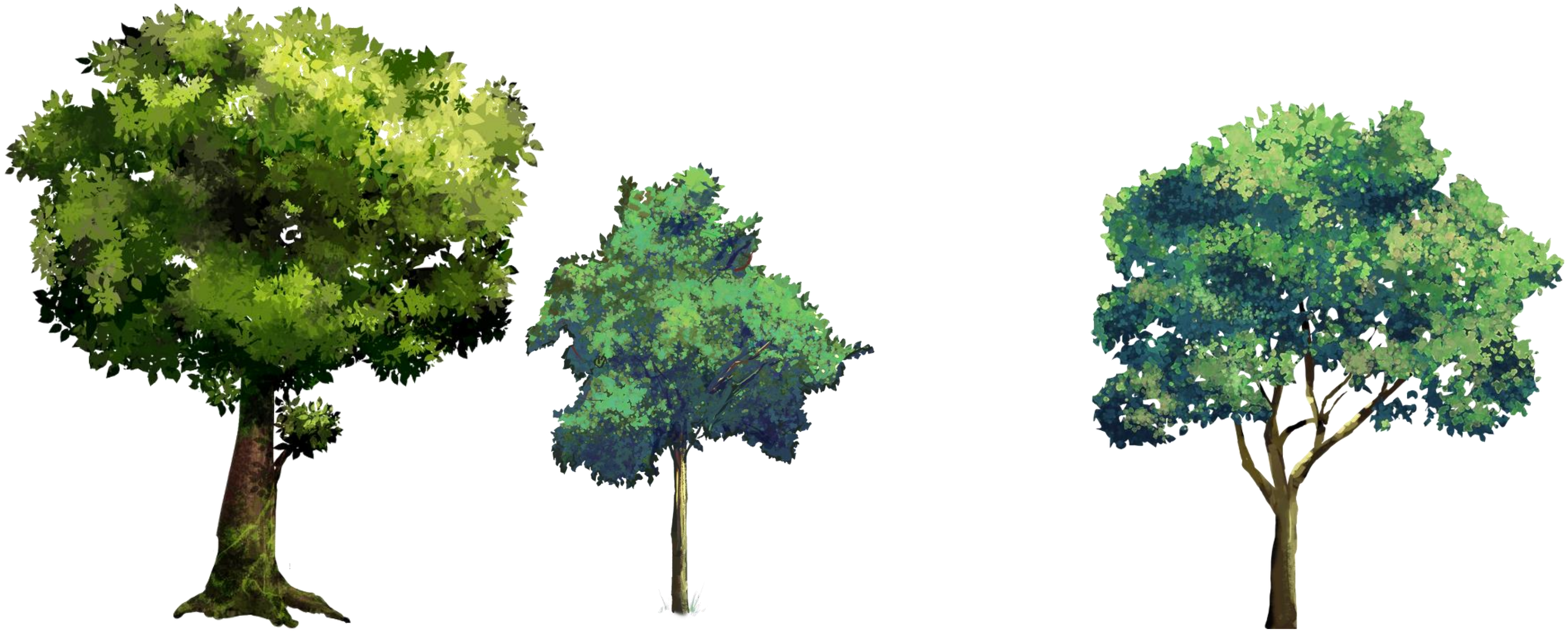
Lab #3.5: Tower 2



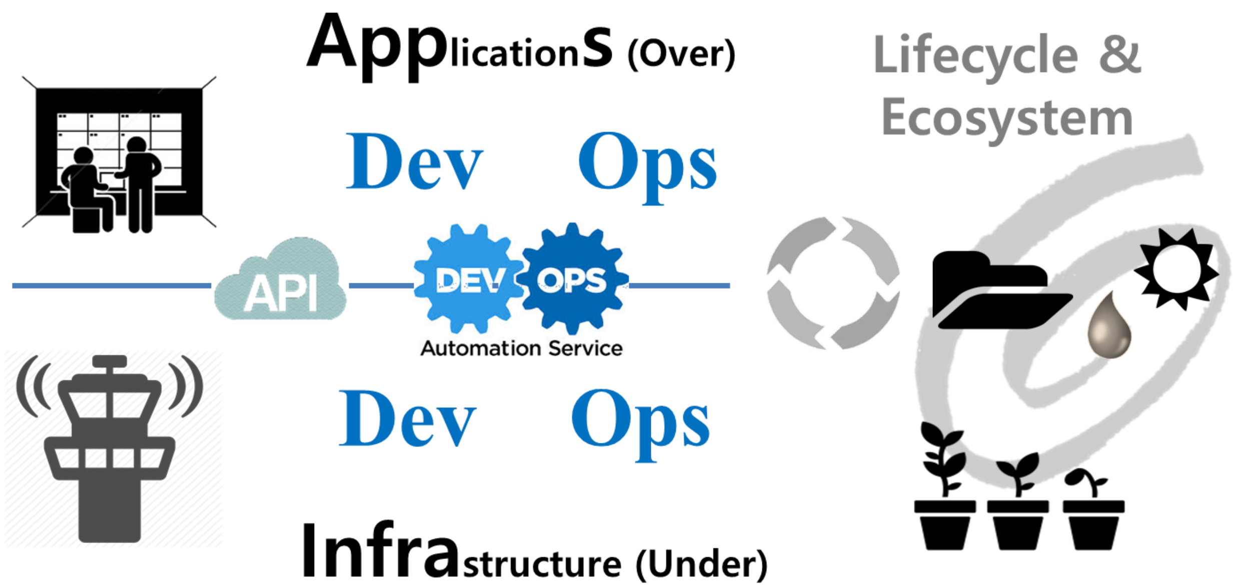
SmartX Labs #1~#3.5: Relationship

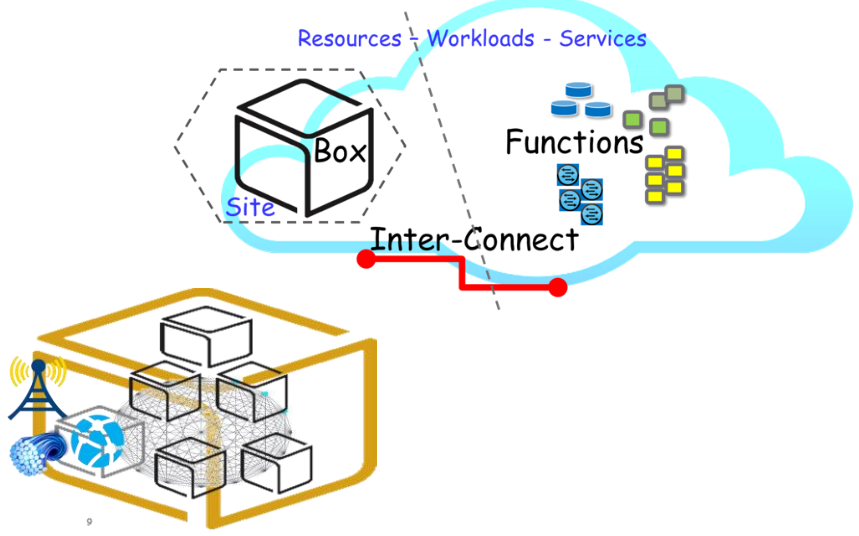


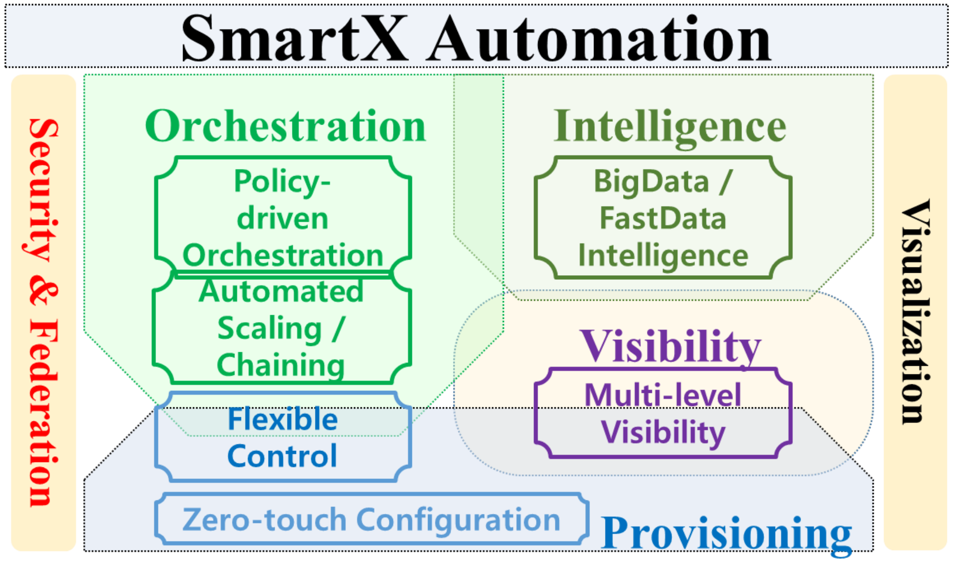
Theory



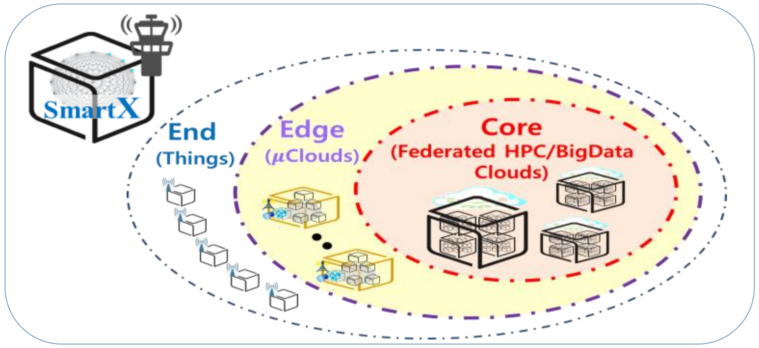
SmartX Automation Framework

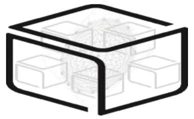






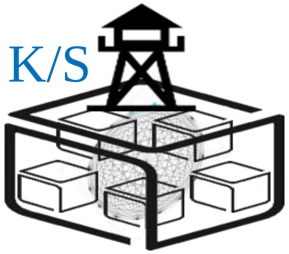
SmartX Composable Playground & Boxes



Type B/C

SmartX
Cloud Box

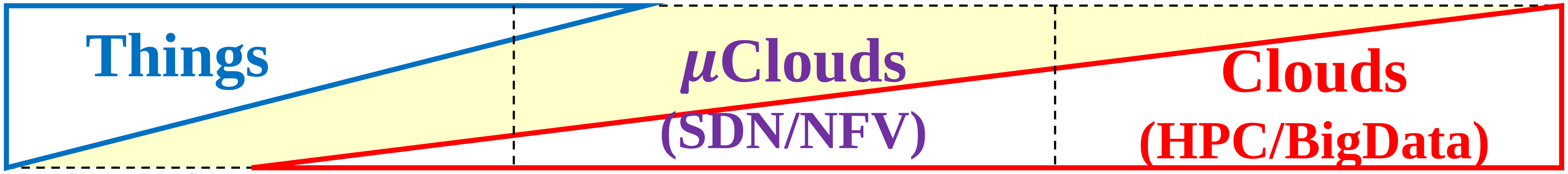
Type O

SmartX
Edge μBoxes

Type K/S

SmartX
Edge Cluster

SmartX
DevOps Tower
Cloud
with DataLake

End	Edge	Core



Visibility: TSDB (Time Series Database)

influxdb

cassandra

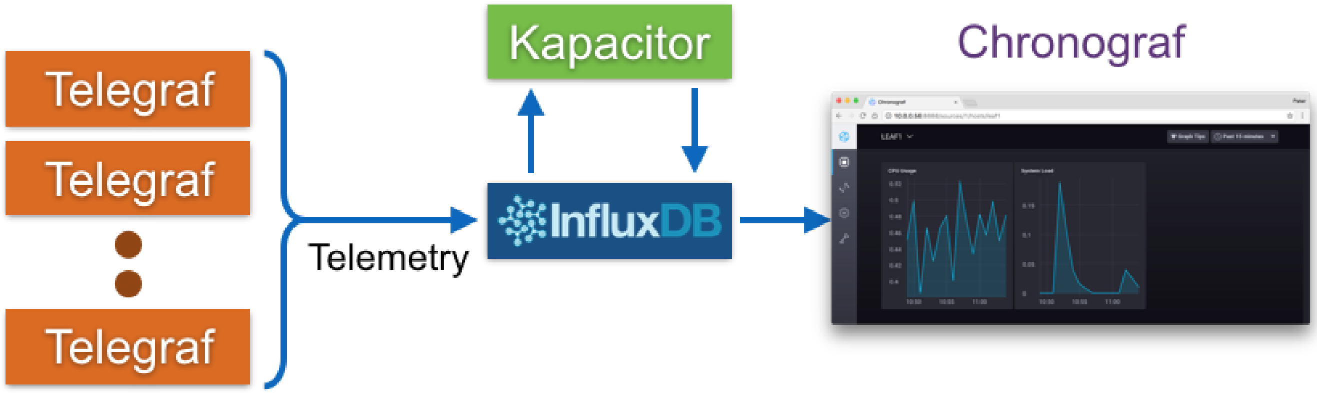
druid

riakTS

InfluxDB

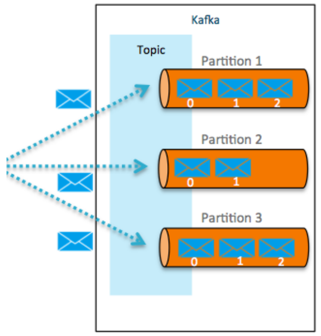
elasticsearch

- Time series data is arrays of numbers indexed by time.
- In some fields these time series are called profiles, curves, or traces.



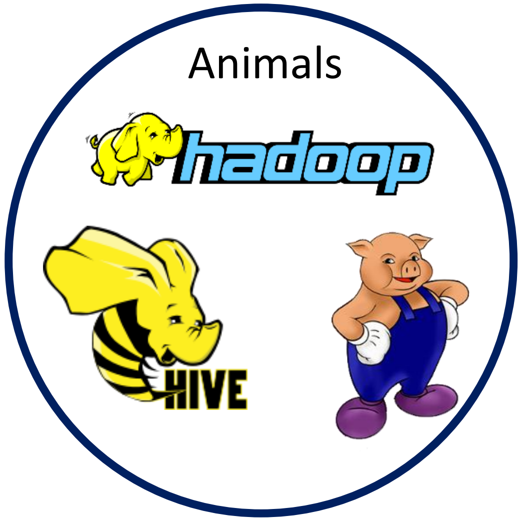
Orchestration: Apache Kafka

- Kafka is a high-throughput distributed messaging system.

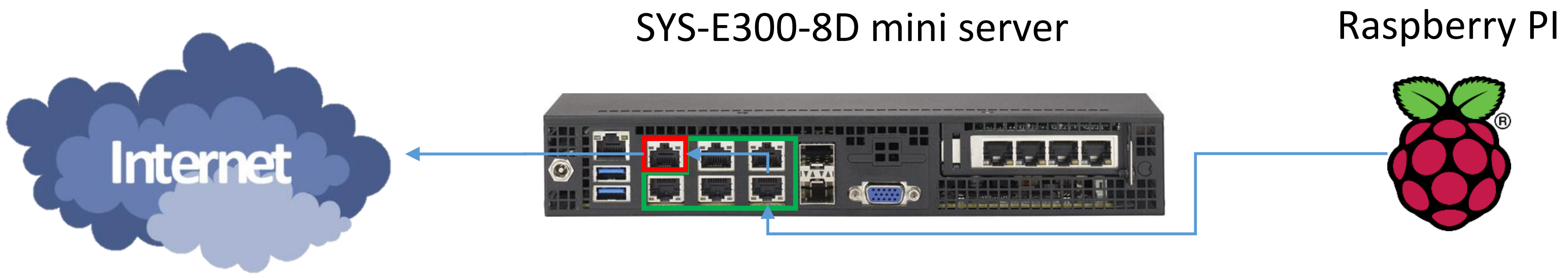


- We can consume message at-most-once, at-least-once or exact-once.
- ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Zookeeper is well used together with Kafka.
- Many subprojects from Apache (especially big data projects including Hadoop) are taking logo originated from animals. 'Zookeeper' maintain and sustain connection between these projects(animals)

takes care of



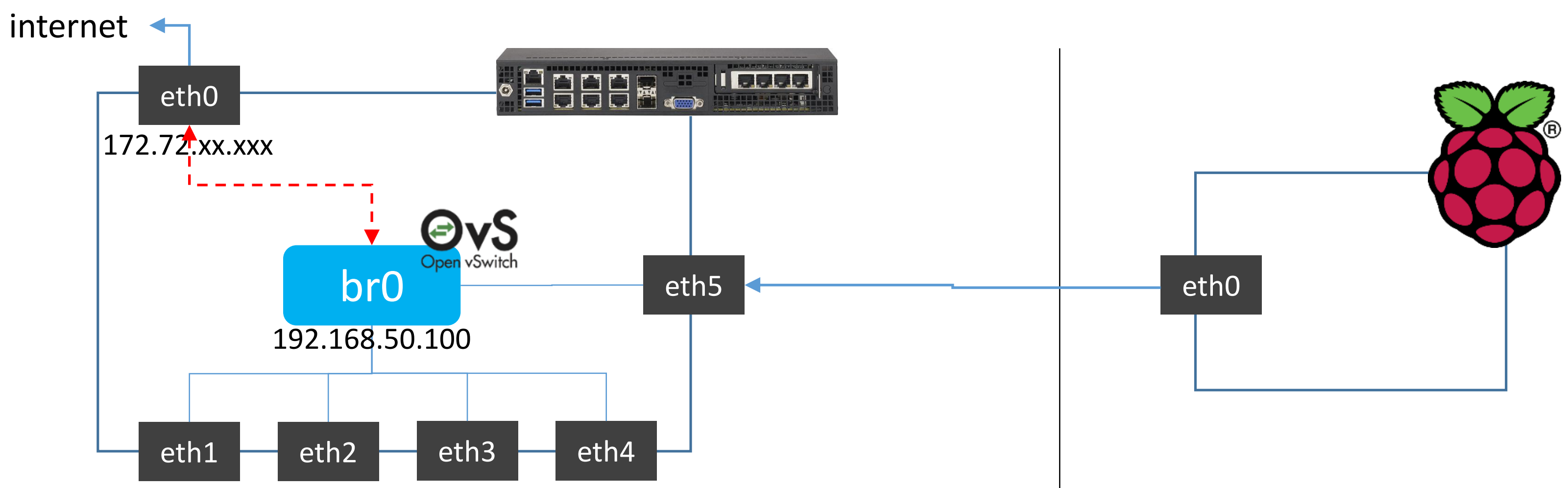
WAN through Tower - 1



- We can configure raspberry pi to connect to public internet through tower
- The three essential techniques are needed: NAT, Masquerade, DHCP
- NAT (Network Address Translation)
  - Exchange network traffic through routers while rewrite the TCP/UDP port number and source and destination of IP address
- Masquerade
  - All network requests generated by internal computers are converted to external public IP address
- DHCP(Dynamic Host Configuration Protocol)
  - Protocol that automatically provides the client with the IP address of the host and the basic setting of various TCP/IP Protocols

WAN through Tower – 2

- Network Configuration
  - Use ‘dnsmasq’ DHCP deamon to provide DHCP server
  - Use linux command ‘iptables’ to modify NAT (Configure to provide Masquerade)



Dnsmasq DHCP Server Configuration

```
$   sudo apt-get install dnsmasq
$   sudo vi /etc/dnsmasq.conf
```

```
server=8.8.8.8
bind-interfaces
domain-needed
bogus-priv
dhcp-range=interface:br0,192.168.50.81,192.168.50.99,12h
dhcp-option=3,192.168.50.100
```

```
$   Sudo /etc/init.d/dnsmasq restart
```

Modify NAT

```
$   sysctl –w net.ipv4.ip_forward=1
$   iptables –A FORWARD –i br0 –j
    ACCEPT
$   iptables –t nat –A POSTROUTING
    –o eth0 –j MASQERADE
```

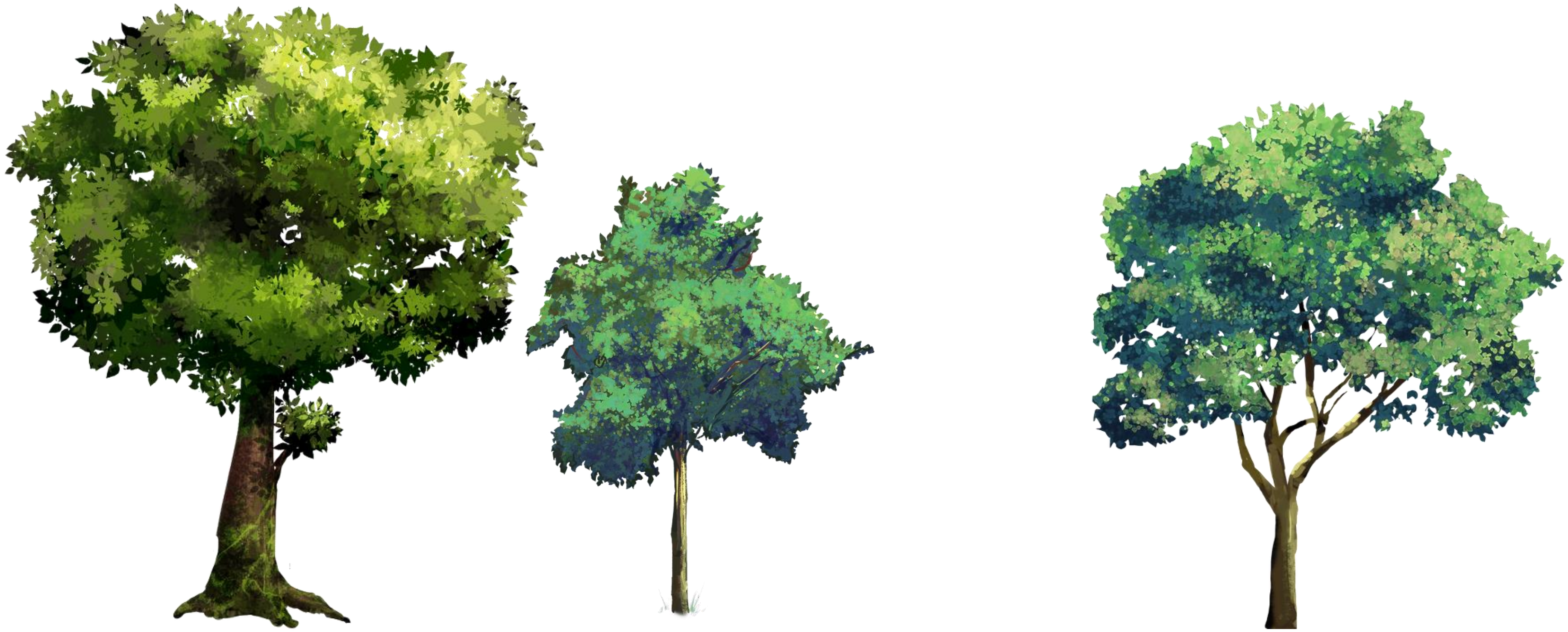
DHCP Client configuation

```
$   sudo vi /etc/network/interfaces
```

```
allow-hotplug
auto eth0
iface eth0 inet dhcp
```

```
$   sudo
    /etc/init.d/networking/restart
```

Practice



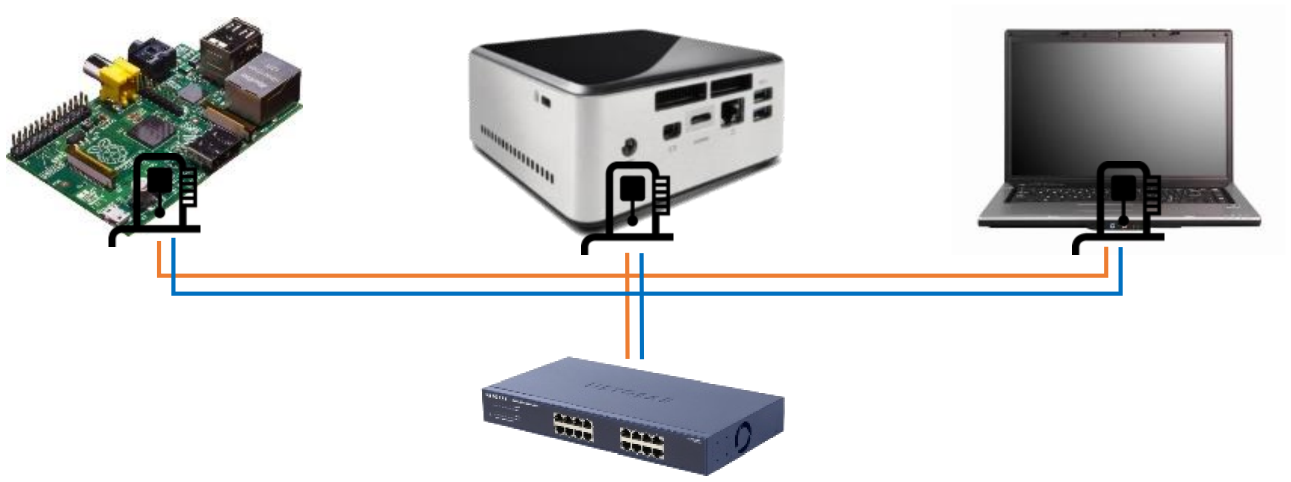
#0 - Lab Preparation (1/1)

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

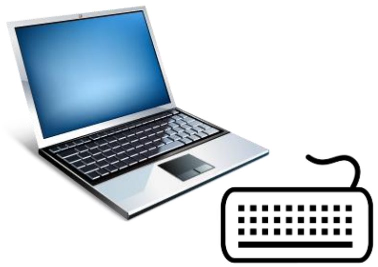
NAME: NT900X3A
CPU: i5-2537U @1.40GHz
CORE: 2
Memory: 4GB DDR3
HDD: 128GB



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

#1 – Tower Setup (1/4)

- Docker installation
  (From Box Lab - 'Making a Docker Container')



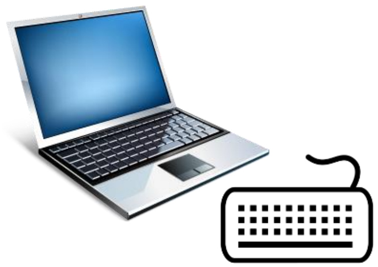
Docker installation.

```
$sudo wget -qO- https://get.docker.com/ | sh
$sudo systemctl start docker
$sudo adduser [Your_account] docker

(Session restart)

$sudo docker run hello-world
```

reference: http://docs.docker.com/linux/step_one/

#1 – Tower Setup (2/4)

- Install & Run Kafka
  (From Inter-Connect Lab 'Kafka deployment')



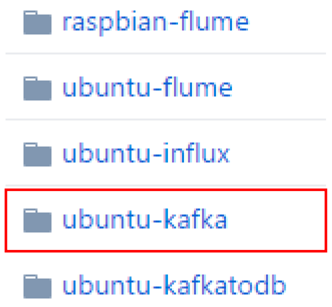
Install git, vim, and nmap

```
$ apt-get install git vim nmap
```

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

#1 – Tower Setup (2/4)

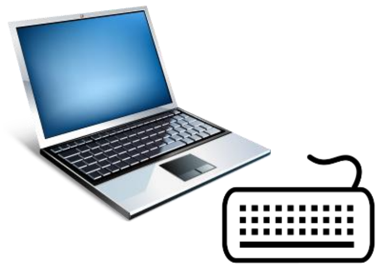
- Install & Run Kafka
  (From Inter-Connect Lab 'Kafka deployment')

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		-	-

#1 – Tower Setup (2/4)

- Install & Run Kafka
  (From Inter-Connect Lab 'Kafka deployment')



- Build Docker Image

```
$ cd ~/SmartX-mini/ubuntu-kafka
```

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

```
$ docker build --tag ubuntu-kafka .
```

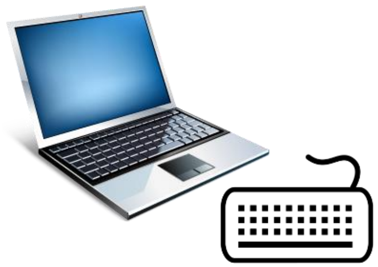
- If you want to check Docker instruction words

```
$ docker --help
```

ex) docker ps : List containers
docker start : Start one or more stopped containers
docker rm : Remove one or more containers

#1 – Tower Setup (2/4)

- Install & Run Kafka
  (From Inter-Connect Lab 'Kafka deployment')



- Run Docker Container

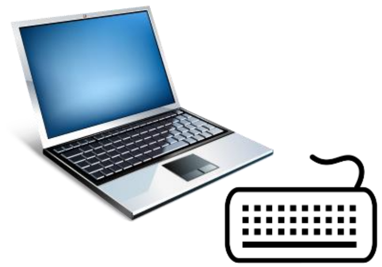
```
$ docker run –it --net=host --name [container name] ubuntu-kafka
```

- 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/

#1 – Tower Setup (2/4)

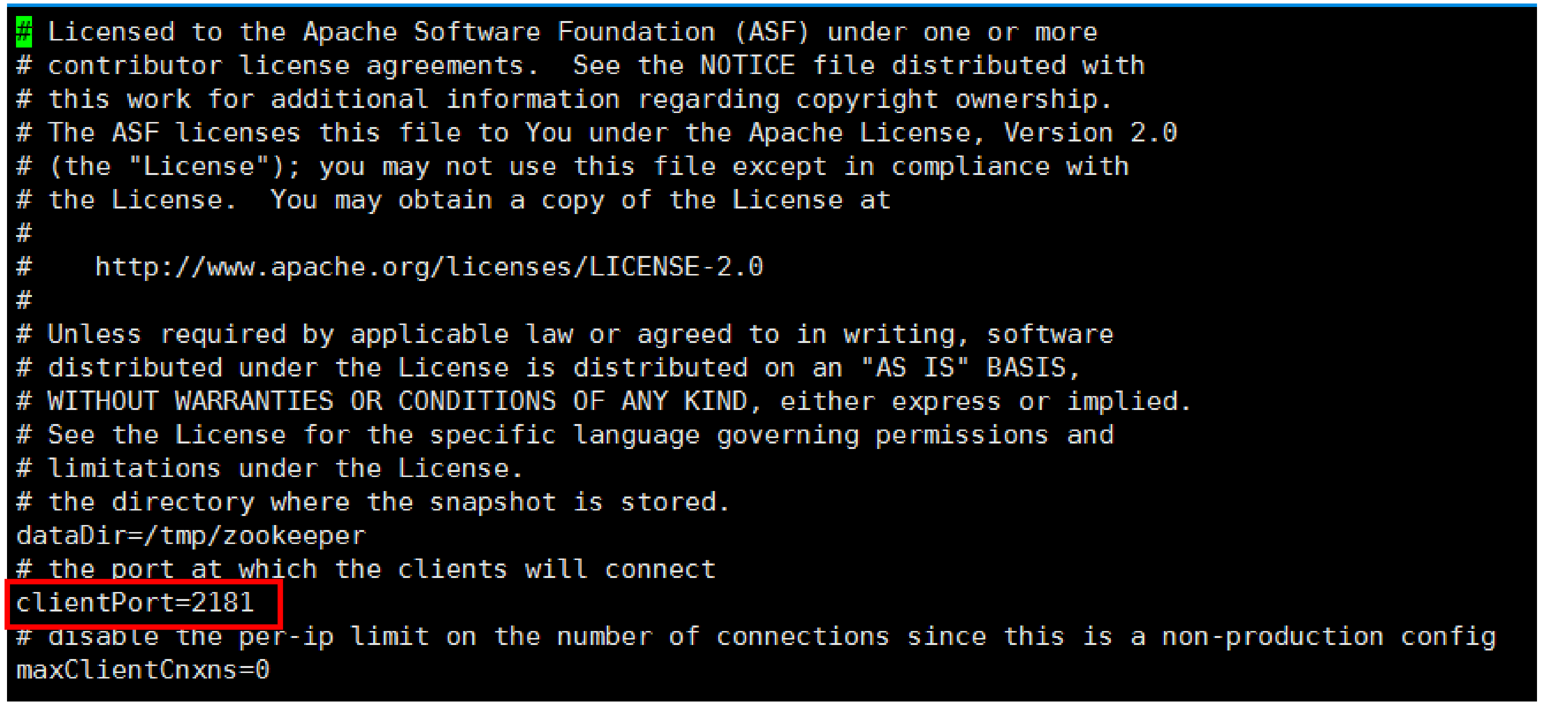
- Install & Run Kafka
  (From Inter-Connect Lab 'Kafka Containers')

✓ Actually we use default configuration



1. Open zookeeper properties file
   $ vi config/zookeeper.properties

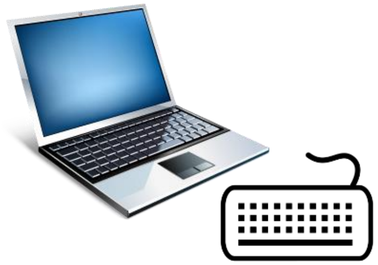
2. Check the client port



```
# 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
```

#1 – Tower Setup (2/4)

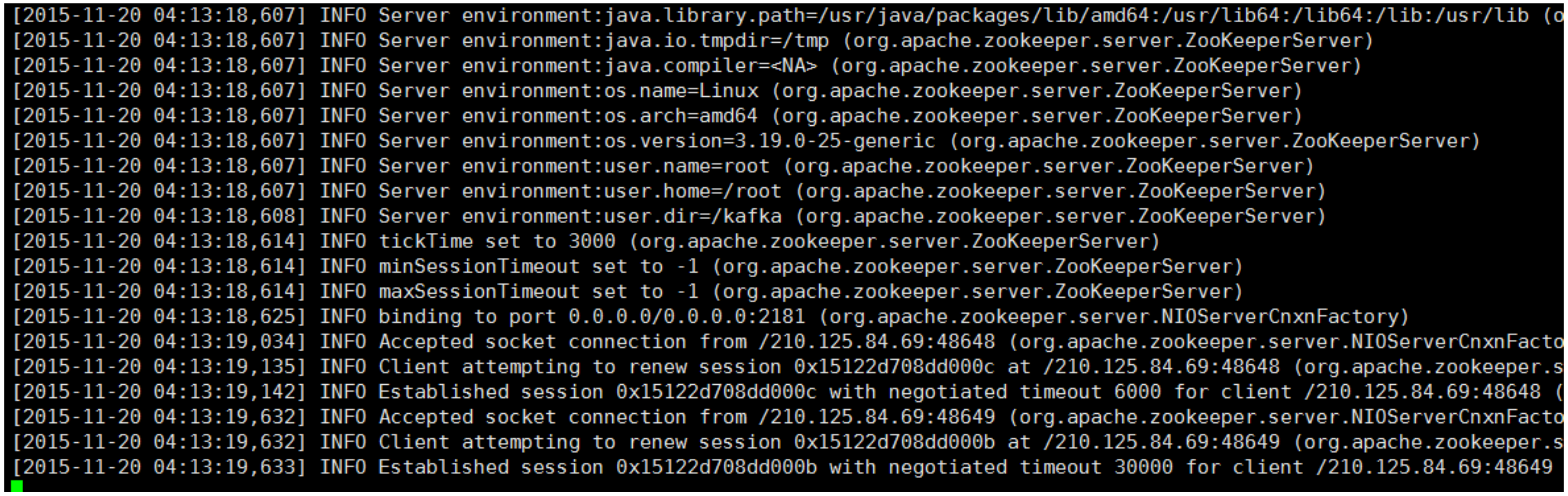
- Install & Run Kafka
  (From Inter-Connect Lab 'Kafka Containers')



✓ zookeeper must be executed FIRST

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

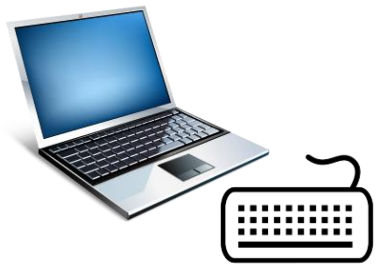
(Leave Zookeeper running and open a new terminal for next tasks)



```
[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
```

#1 – Tower Setup (2/4)

- Install & Run Kafka
  (From Inter-Connect Lab 'Kafka Containers')

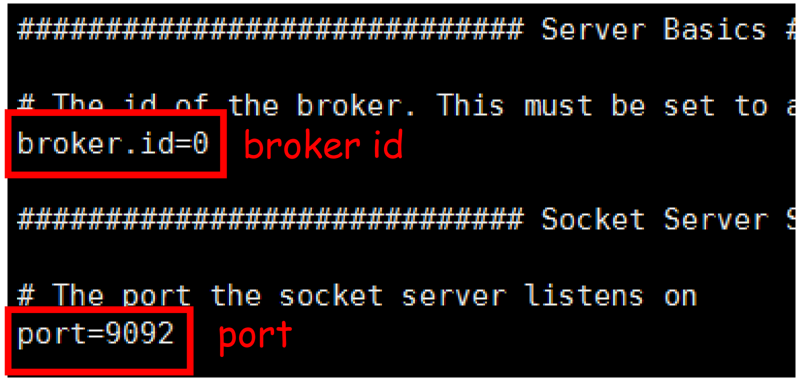


- Create a Kafka container with the docker command before

```
$ 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
```

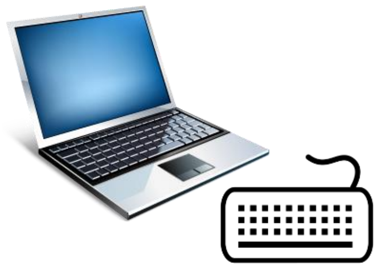


```
############################# Server Basics #

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

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

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

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

Consumer container will not run any brokers

#1 – Tower Setup (2/4)



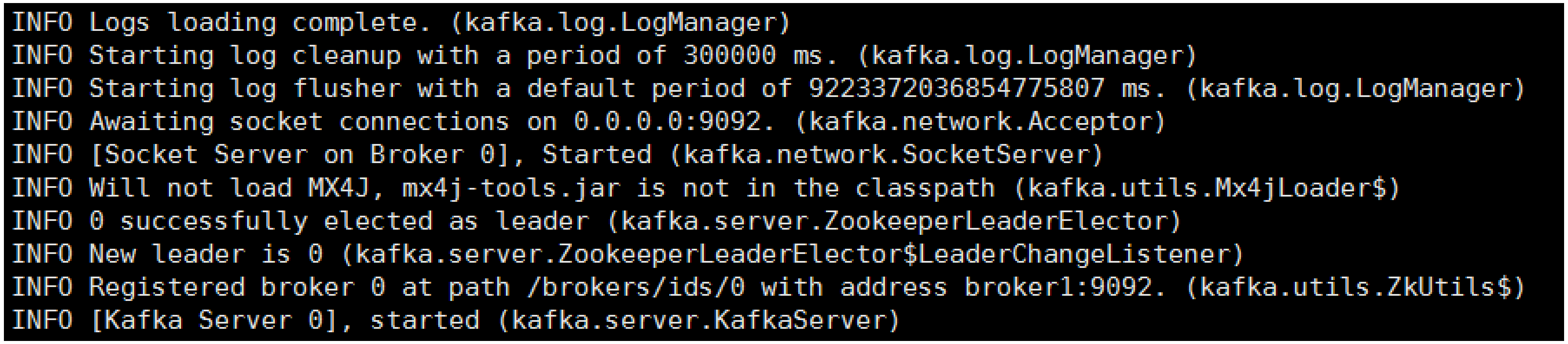
- Install & Run Kafka
  (From Inter-Connect Lab 'Kafka Containers')

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

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

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

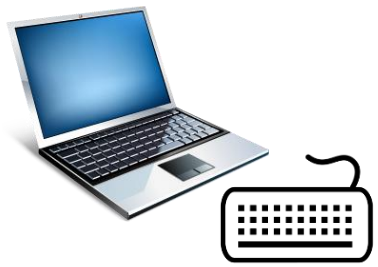
✓ When it successfully works, each broker containers will show messages like the below



```
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)
```

#1 – Tower Setup (2/4)

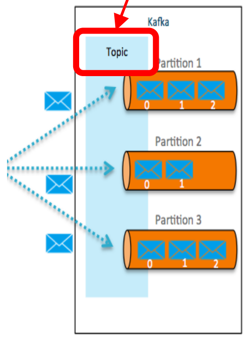
- Install & Run Kafka



- Create topic

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

$ bin/kafka-topics.sh --create --zookeeper localhost:2181 --
replication-factor 1 --partitions 3 --topic sshlog
```

resource



- We can check topics.

Topic List

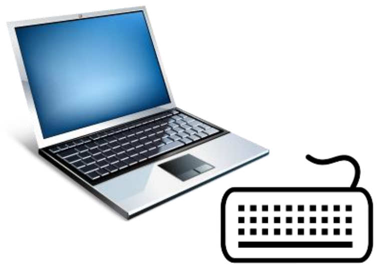
```
$ bin/kafka-topics.sh --list --zookeeper localhost:2181
```

Topic specification

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

#1 – Tower Setup (3/4)

- Install & Run InfluxDB & Chronograf
  (From Tower Lab 'Run InfluxDB & Chronograf Containers on NUC)



- Run InfluxDB Container

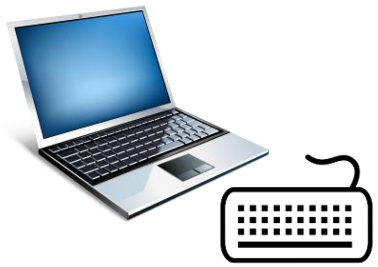
```
$ docker run –d ––name=influxdb ––net=host influxdb
```

- Make and run Chronograf container

```
$ docker run -p 8888:8888 --net=host chronograf --influxdb-url=http://<TOWER
IP>:8086
```

#1 – Tower Setup (3/4)

- Install & Run InfluxDB & Chronograf
  (From Tower Lab 'Run InfluxDB & Chronograf Containers on NUC)



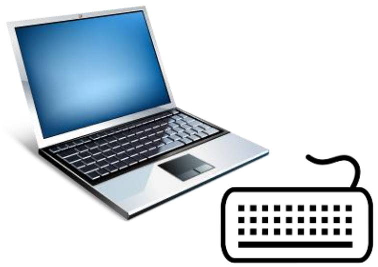
- Install python–pip

```
$ sudo apt-get install -y libcurl3 openssl curl
$ sudo apt-get install -y python2.7 python-pip
$ sudo apt-get install –y python3-pip
```

- Install python package

```
$ sudo pip install requests
$ sudo pip install kafka–python
$ sudo pip install influxdb
$ sudo pip install msgpack
```

#1 – Tower Setup (4/4)



- Send broker message to influxDB (in Tower)

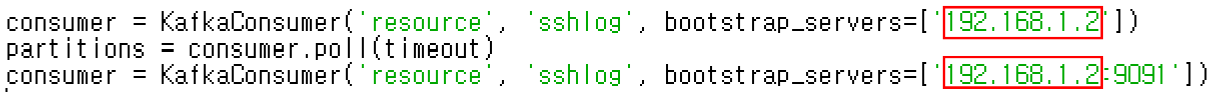
- Start new terminal

- Download python script file (INSIDE the container)
  wget -O broker_to_influxdb.py **https://raw.githubusercontent.com/yd8012mw2/SmartXLab_FileHost/master/broker_to_influxdb.py**

  (No space/new line)

- Edit File

  To Tower IP

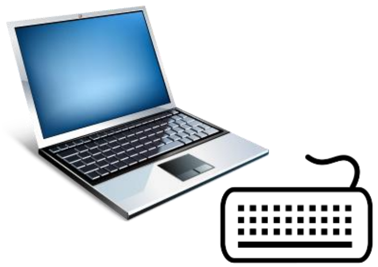


```
consumer = KafkaConsumer('resource', 'sshlog', bootstrap_servers=['192.168.1.2'])
partitions = consumer.poll(timeout)
consumer = KafkaConsumer('resource', 'sshlog', bootstrap_servers=['192.168.1.2:9091'])
```

- Run python script file
  python broker_to_influxdb.py

#1 – Tower Setup (4/4)

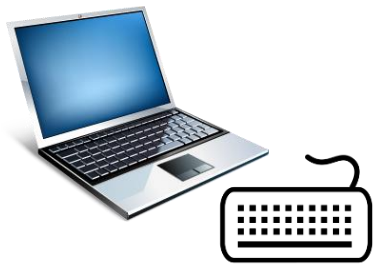
- Install simple management dashboard



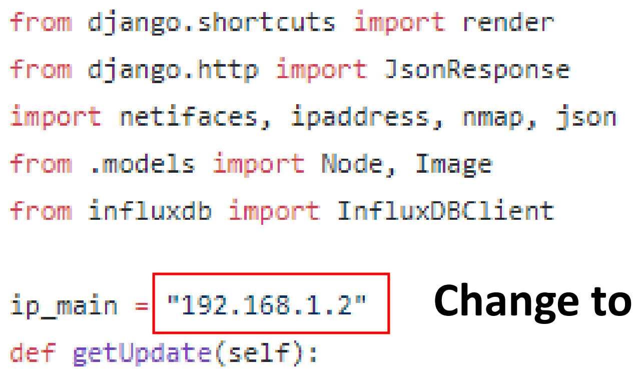
- Start new terminal
- Clone the project from github
  git clone https://github.com/yd8012mw2/GIST_TOWER2
- Move to project directory
  cd GIST_TOWER2
- Install packages
  pip3 install -r requirements.txt
- Move to tower2 folder
  cd tower2

#1 – Tower Setup (4/4)

- Install simple management dashboard



- Edit views.py (towersite/views.py)
  vi towersite/views.py



```
from django.shortcuts import render
from django.http import JsonResponse
import netifaces, ipaddress, nmap, json
from .models import Node, Image
from influxdb import InfluxDBClient

ip_main = "192.168.1.2"
def getUpdate(self):
```

Change to the tower's ip

#2 – NUC Setup

- Install SNMP package for NUC
(From InterConnect Lab '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
```

#2 – NUC Setup

- Install flume package for NUC

- Go to SmartX-mini/ubuntu-flume directory
  ```
  cd ~/SmartX-mini/ubuntu-flume
  ```

- Build flume container
  ```
  docker build --tag ubuntu-flume .
  ```
  (Put punctuation mark at the end of the command)

#3 – Box Setup

- Edit flume configuration

- Add tower ip and tower computer name at the end of file '/etc/hosts'

- Build flume container

  In case of NUC

  ```
  docker run –it --net=host --name flume2 –v /var/log/auth.log:/var/log/auth.log:ro ubuntu-flume
  ```

  In case of Raspberry Pi

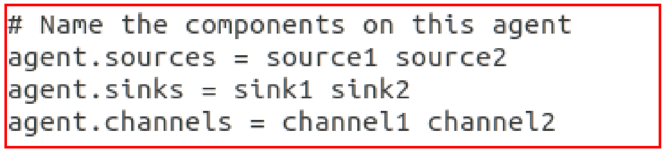
  ```
  docker run –it --net=host --name flume2 –v /var/log/auth.log:/var/log/auth.log:ro raspbian-flume
  ```

#3 – Box Setup

- Edit flume configuration

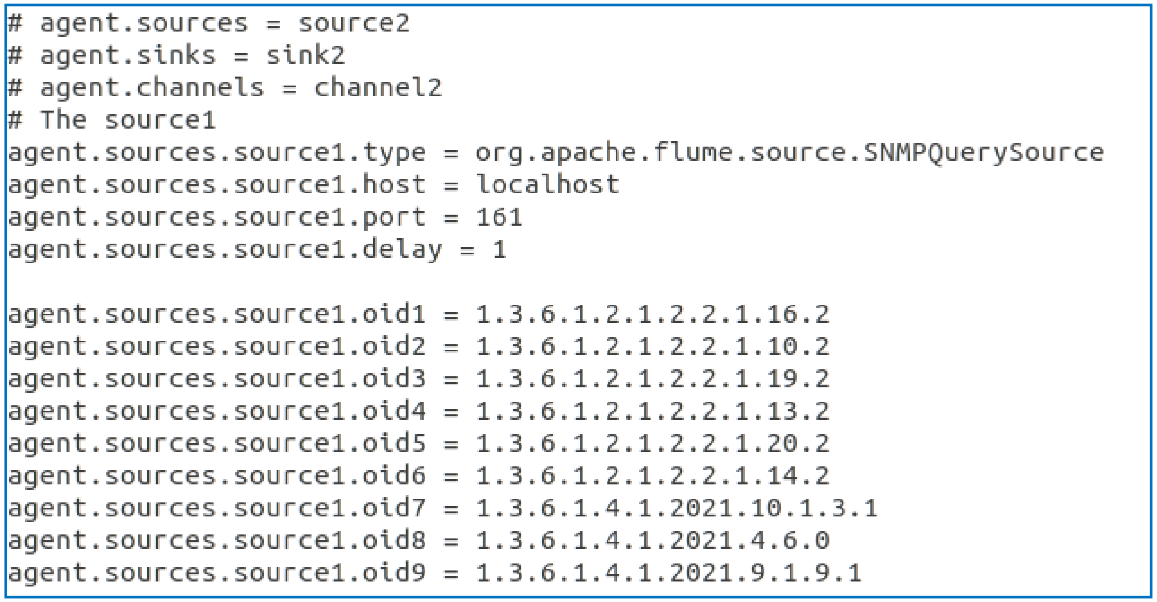
- Edit flume configuration

```
docker start flume2
docker attach flume2
vi conf/flume-conf.properties
```



```
# Name the components on this agent
agent.sources = source1 source2
agent.sinks = sink1 sink2
agent.channels = channel1 channel2
```

You should change the content here



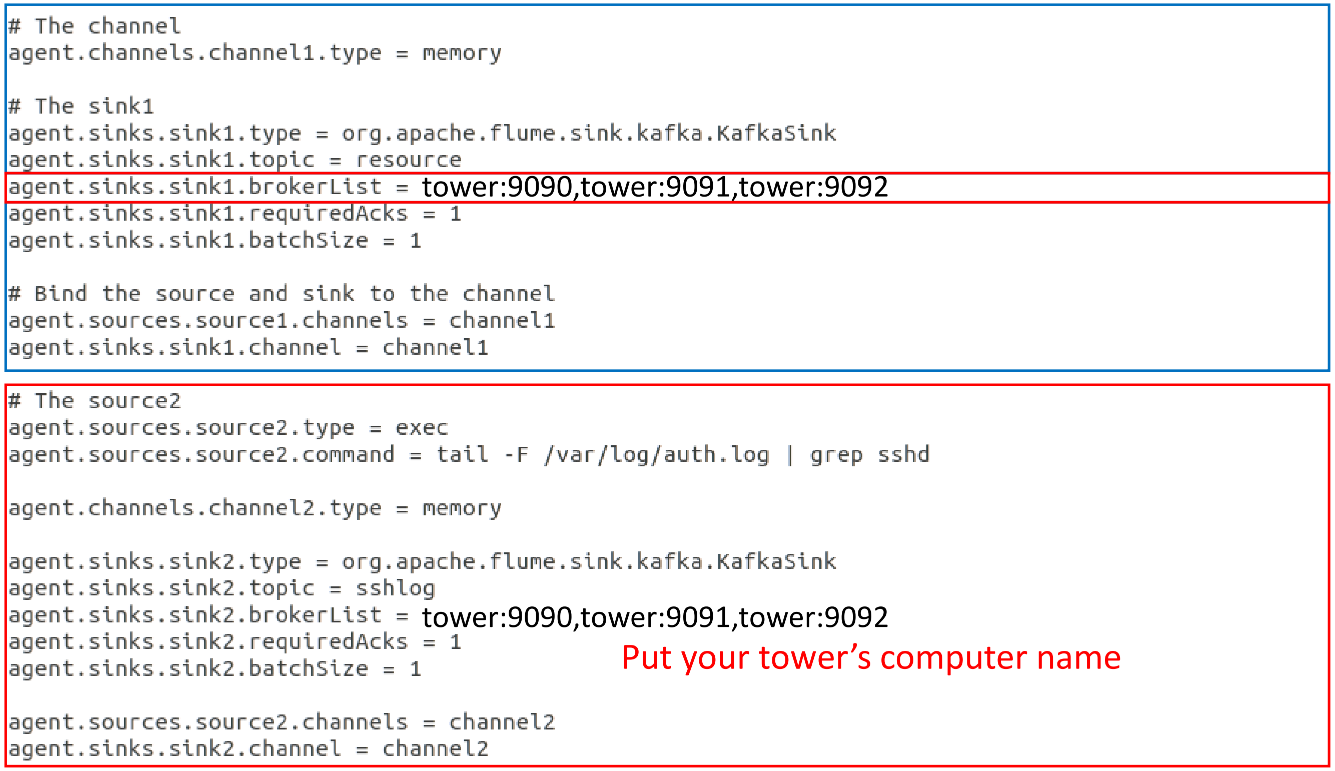
```
# agent.sources = source2
# agent.sinks = sink2
# agent.channels = channel2
# The source1
agent.sources.source1.type = org.apache.flume.source.SNMPQuerySource
agent.sources.source1.host = localhost
agent.sources.source1.port = 161
agent.sources.source1.delay = 1

agent.sources.source1.oid1 = 1.3.6.1.2.1.2.2.1.16.2
agent.sources.source1.oid2 = 1.3.6.1.2.1.2.2.1.10.2
agent.sources.source1.oid3 = 1.3.6.1.2.1.2.2.1.19.2
agent.sources.source1.oid4 = 1.3.6.1.2.1.2.2.1.13.2
agent.sources.source1.oid5 = 1.3.6.1.2.1.2.2.1.20.2
agent.sources.source1.oid6 = 1.3.6.1.2.1.2.2.1.14.2
agent.sources.source1.oid7 = 1.3.6.1.4.1.2021.10.1.3.1
agent.sources.source1.oid8 = 1.3.6.1.4.1.2021.4.6.0
agent.sources.source1.oid9 = 1.3.6.1.4.1.2021.9.1.9.1
```

You should NOT
change the content here

#3 – Box Setup

- Edit flume configuration

- Edit flume configuration
  docker start flume2
  docker attach flume2
  vi conf/flume-conf.properties



```
# The channel
agent.channels.channel1.type = memory

# The sink1
agent.sinks.sink1.type = org.apache.flume.sink.kafka.KafkaSink
agent.sinks.sink1.topic = resource
agent.sinks.sink1.brokerList = tower:9090,tower:9091,tower:9092
agent.sinks.sink1.requiredAcks = 1
agent.sinks.sink1.batchSize = 1

# Bind the source and sink to the channel
agent.sources.source1.channels = channel1
agent.sinks.sink1.channel = channel1
```

You should NOT change the content here

Change the broker ip to tower

You should NOT change the content here

```
# The source2
agent.sources.source2.type = exec
agent.sources.source2.command = tail -F /var/log/auth.log | grep sshd

agent.channels.channel2.type = memory

agent.sinks.sink2.type = org.apache.flume.sink.kafka.KafkaSink
agent.sinks.sink2.topic = sshlog
agent.sinks.sink2.brokerList = tower:9090,tower:9091,tower:9092
agent.sinks.sink2.requiredAcks = 1
agent.sinks.sink2.batchSize = 1

agent.sources.source2.channels = channel2
agent.sinks.sink2.channel = channel2
```

Put your tower's computer name

You should ADD the content here

#3 – Box Setup

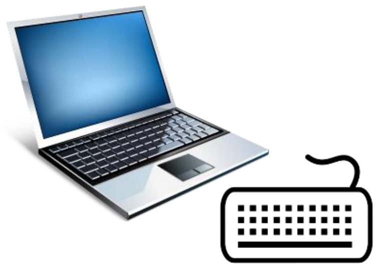
- Edit flume configuration

Run Flume on Raspberry Pi and NUC

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

#4 – Dashboard Management

- Run simple management dashboard



- Check Database migration

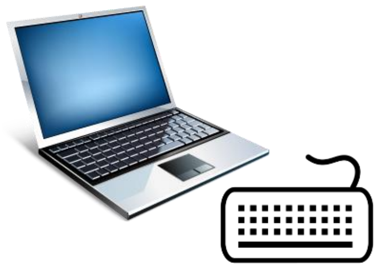
```
python3 manage.py makemigrations
python3 manage.py migrate
```

- Run server

```
python3 manage.py runserver 0.0.0.0:8000
```

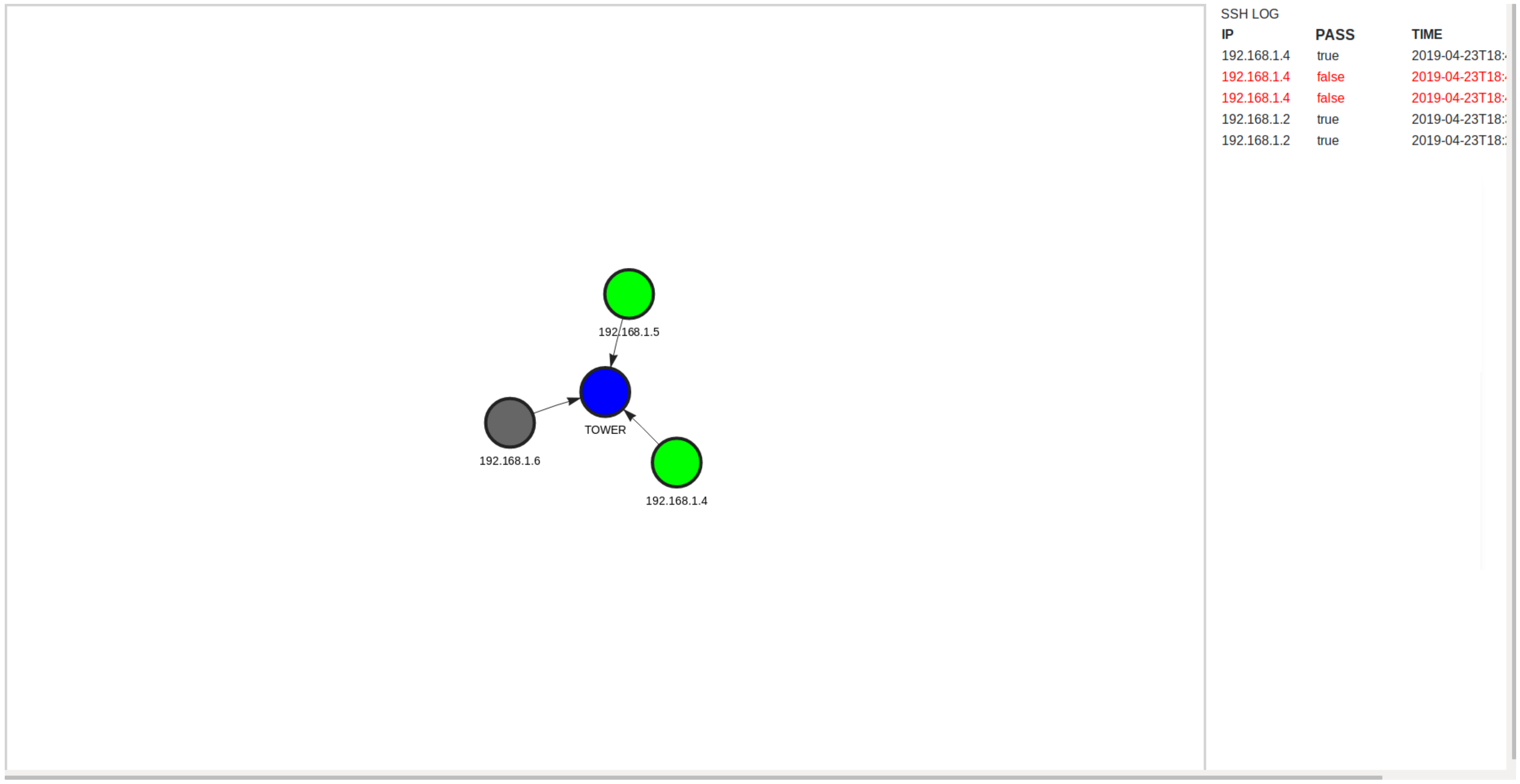
- Start the web browser and open ‘localhost:8000’

#4 – Dashboard Management



- Run simple management dashboard

- Sample images



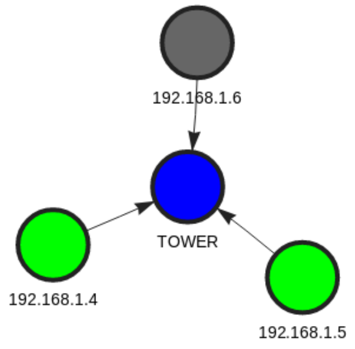
SSH LOG

IP	PASS	TIME
192.168.1.4	true	2019-04-23T18:
192.168.1.4	false	2019-04-23T18:
192.168.1.4	false	2019-04-23T18:
192.168.1.2	true	2019-04-23T18:
192.168.1.2	true	2019-04-23T18:

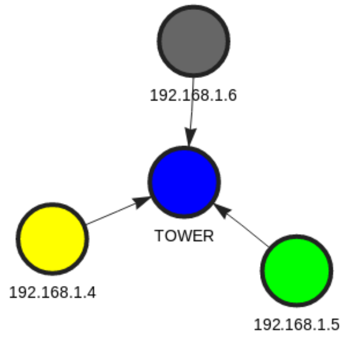
#4 – Dashboard Management

- Run simple management dashboard

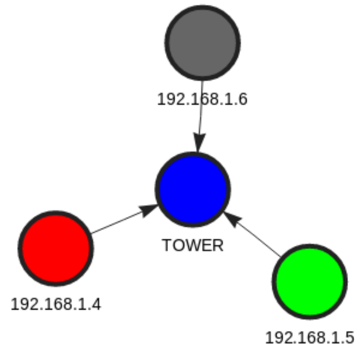
- Sample images



CPU Usage : Low



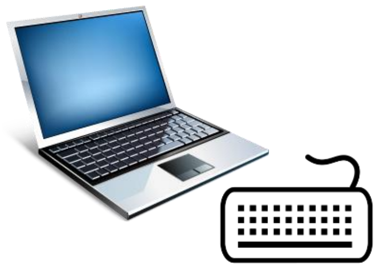
CPU Usage : Moderate



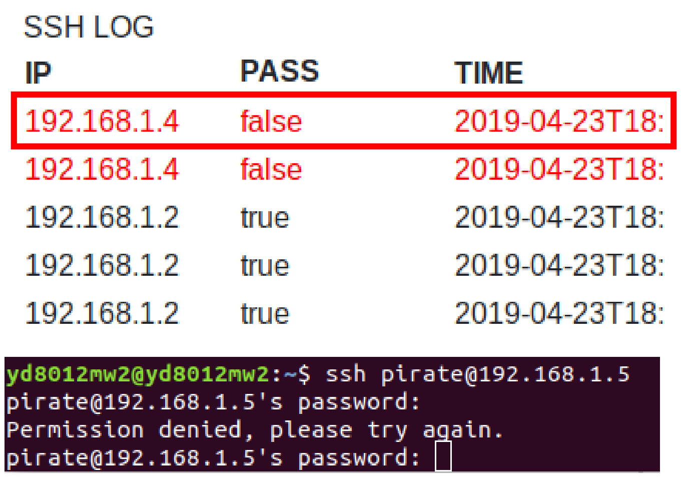
CPU Usage : Excessive

#4 – Dashboard Management

- Run simple management dashboard



- Sample images

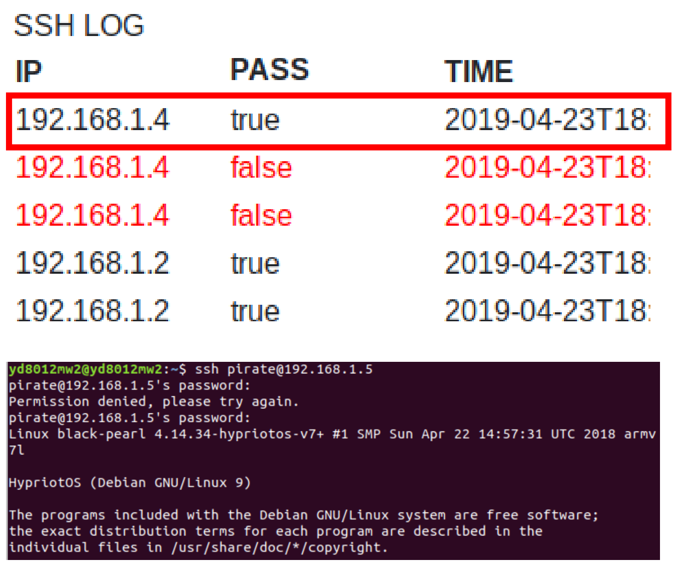


SSH LOG

IP	PASS	TIME
192.168.1.4	false	2019-04-23T18:
192.168.1.4	false	2019-04-23T18:
192.168.1.2	true	2019-04-23T18:
192.168.1.2	true	2019-04-23T18:
192.168.1.2	true	2019-04-23T18:

```
yd8012mw2@yd8012mw2:~$ ssh pirate@192.168.1.5
pirate@192.168.1.5's password:
Permission denied, please try again.
pirate@192.168.1.5's password:
```

If SSH Login fails



SSH LOG

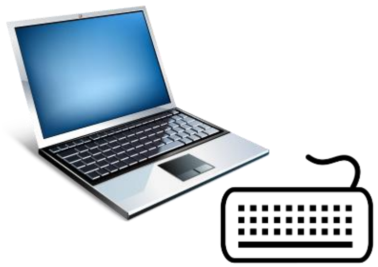
IP	PASS	TIME
192.168.1.4	true	2019-04-23T18:
192.168.1.4	false	2019-04-23T18:
192.168.1.4	false	2019-04-23T18:
192.168.1.2	true	2019-04-23T18:
192.168.1.2	true	2019-04-23T18:

```
yd8012mw2@yd8012mw2:~$ ssh pirate@192.168.1.5
pirate@192.168.1.5's password:
Permission denied, please try again.
pirate@192.168.1.5's password:
Linux black-pearl 4.14.34-hypriotos-v7+ #1 SMP Sun Apr 22 14:57:31 UTC 2018 armv
7l

HypriotOS (Debian GNU/Linux 9)

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
```

If SSH Login succeeds

#4 – Dashboard Management

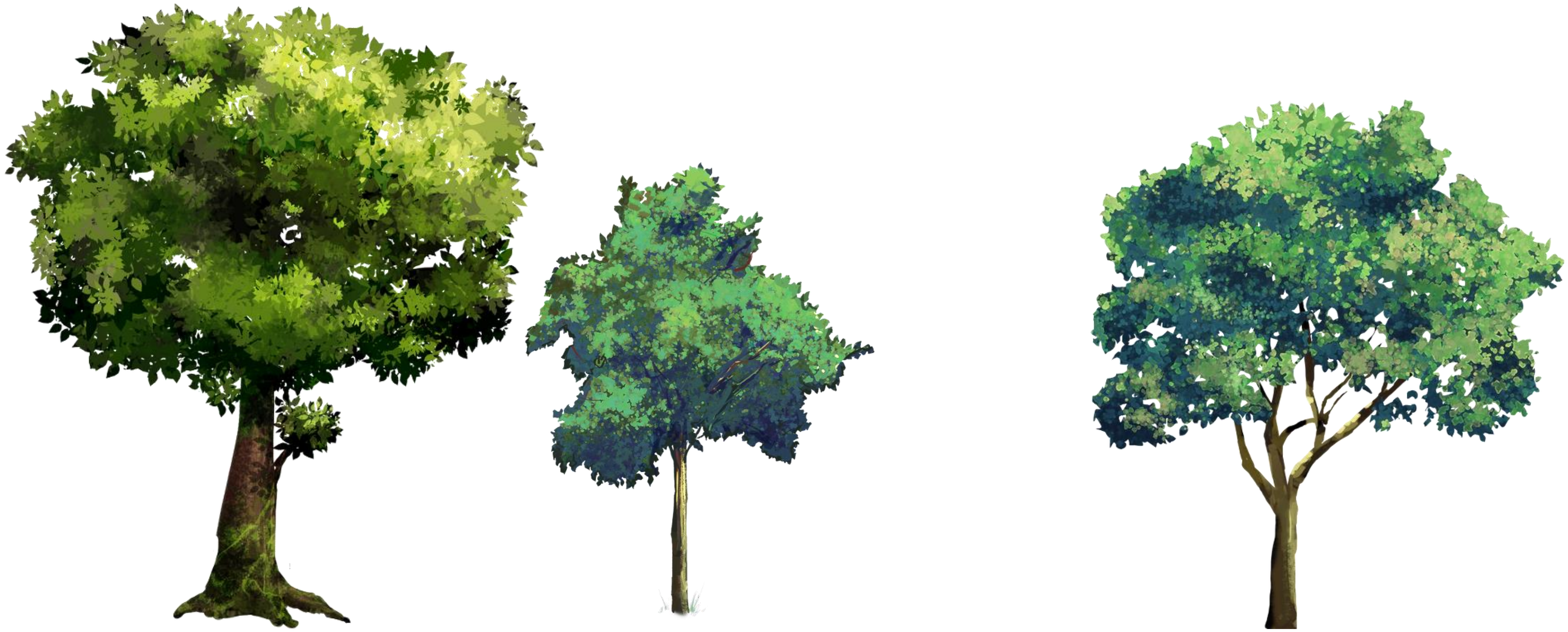
- Run simple management dashboard



- Stress computer (Not recommended for Raspberry Pi)

```
apt-get install stress
stress --cpu 4
```

Review



Lab Summary

With Tower Lab, you have experimented selected roles of Monitor/Control (관제) Tower

1. Visibility Center function to **enable 'distributed monitoring'** over remote Boxes and to **store 'monitoring information'** to time-size DB.

2. Provisioning Center function to **enable remote 'installation & configuration** (of OS and others)' of distributed Boxes.

Thank You for Your Attention
Any Questions?

mini@smartx.kr