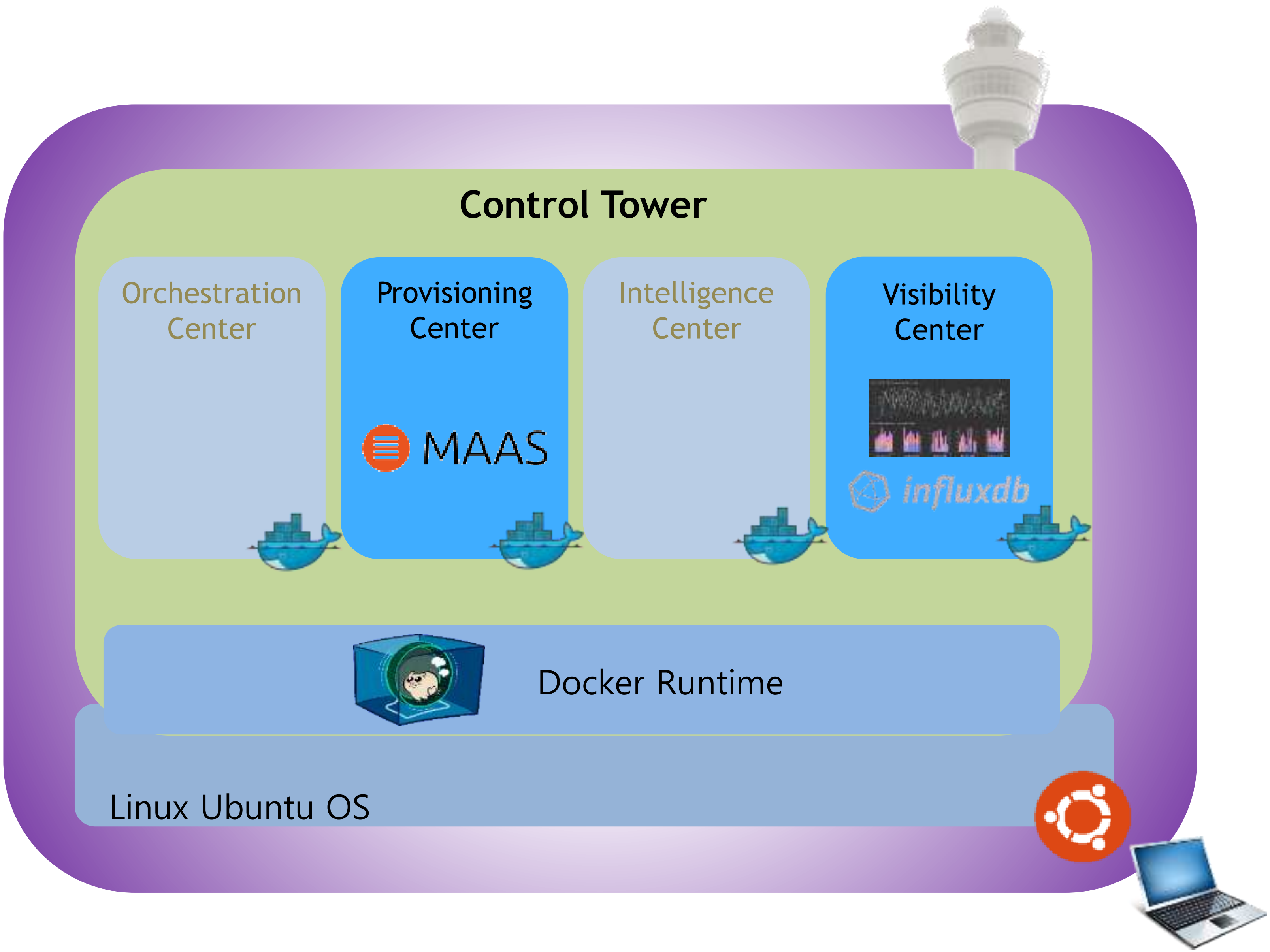
Computer Systems for AI-inspired Cloud Theory & Lab.

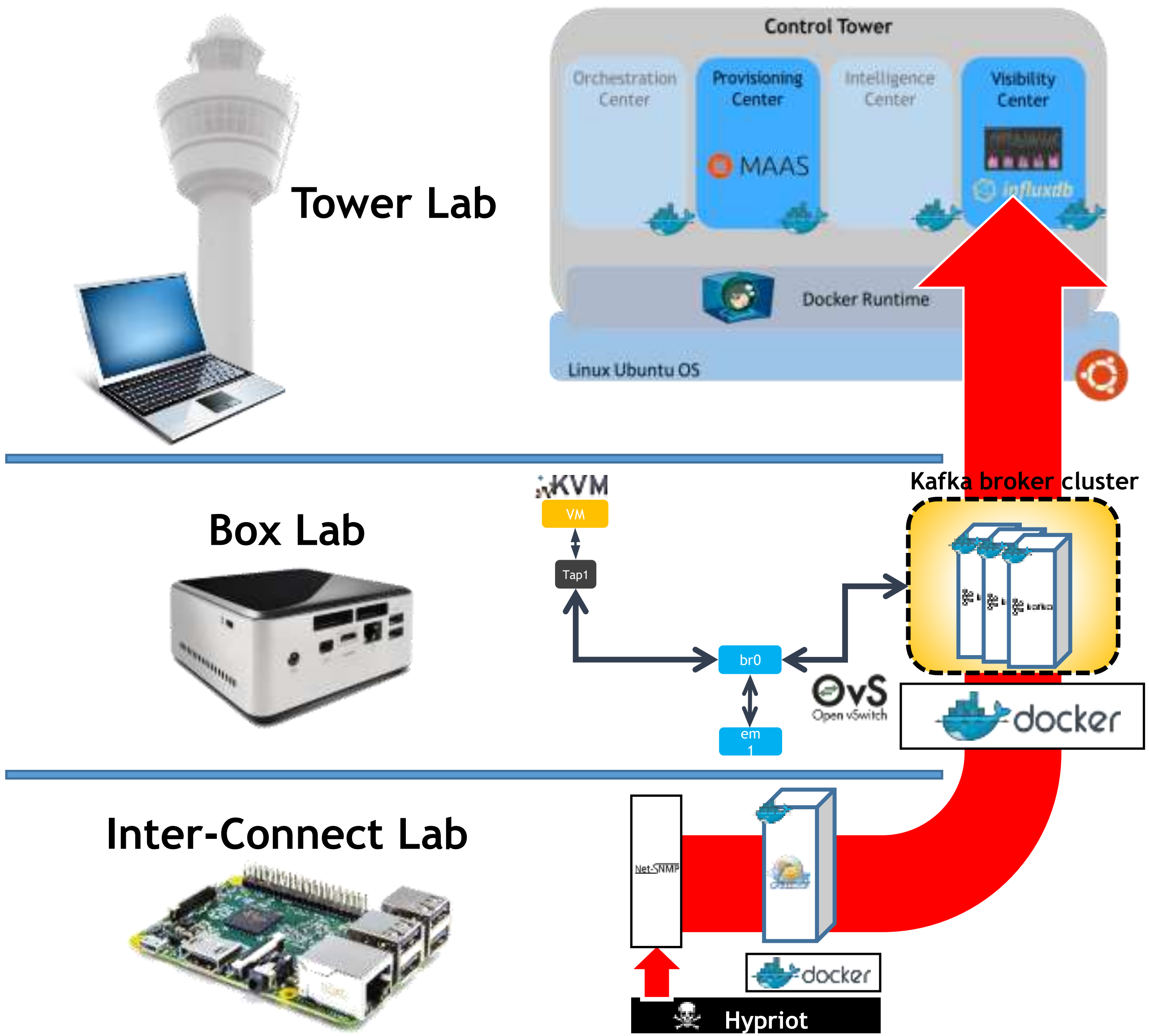
SmartX Labs – Mini (MOOC Selection)

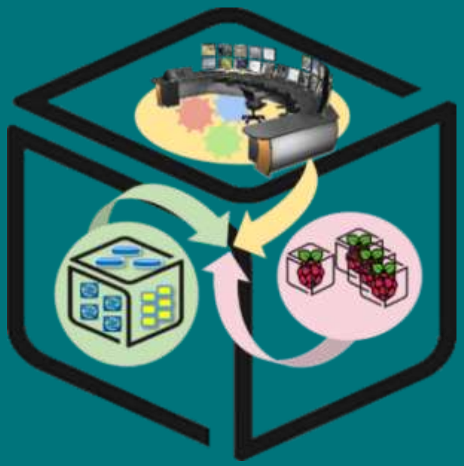
Tower (Lab#3)

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

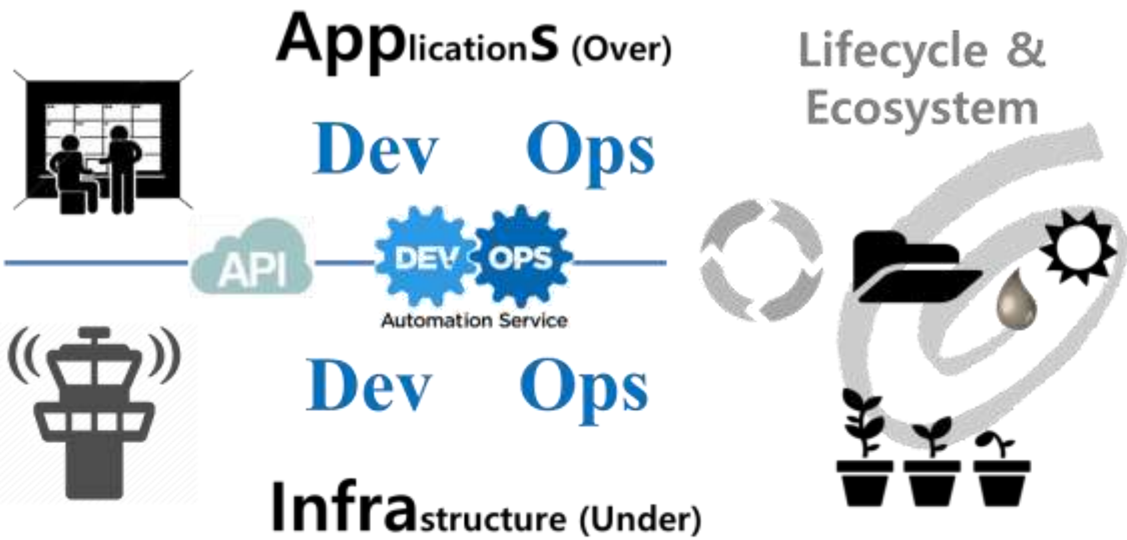
Tower Lab: Concept

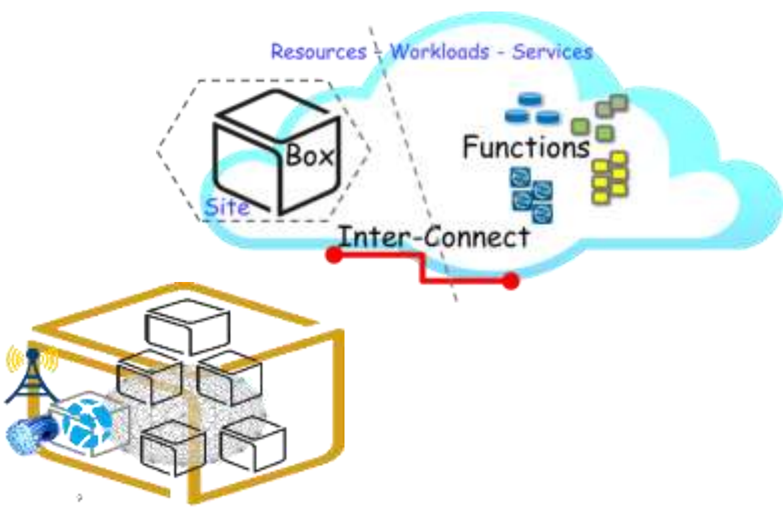


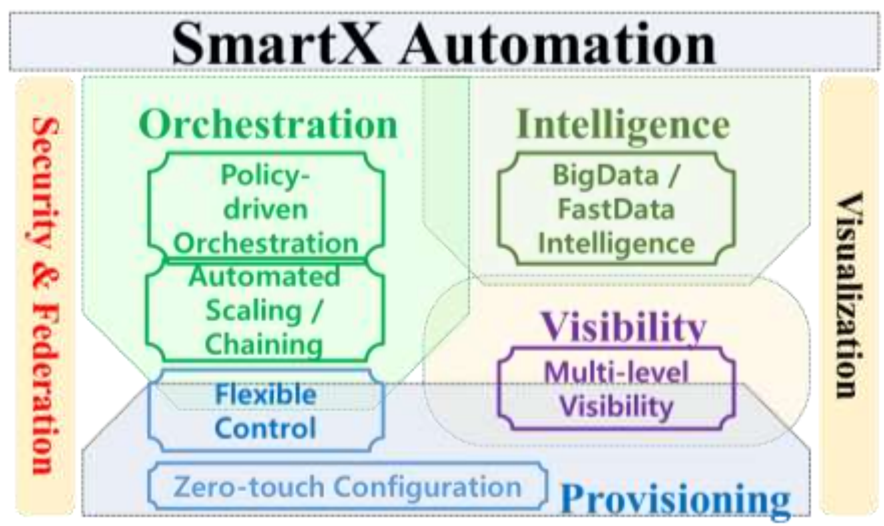

Tower Lab
Control Tower
Orchestration Center
Provisioning Center
MAAS
Intelligence Center
Visibility Center
influxdb
Docker Runtime
Linux Ubuntu OS
Box Lab
KVM
VM
Tap1
br0
em1
Open vSwitch
Kafka broker cluster
docker
Inter-Connect Lab
Net-SNMP
docker
Hypriot

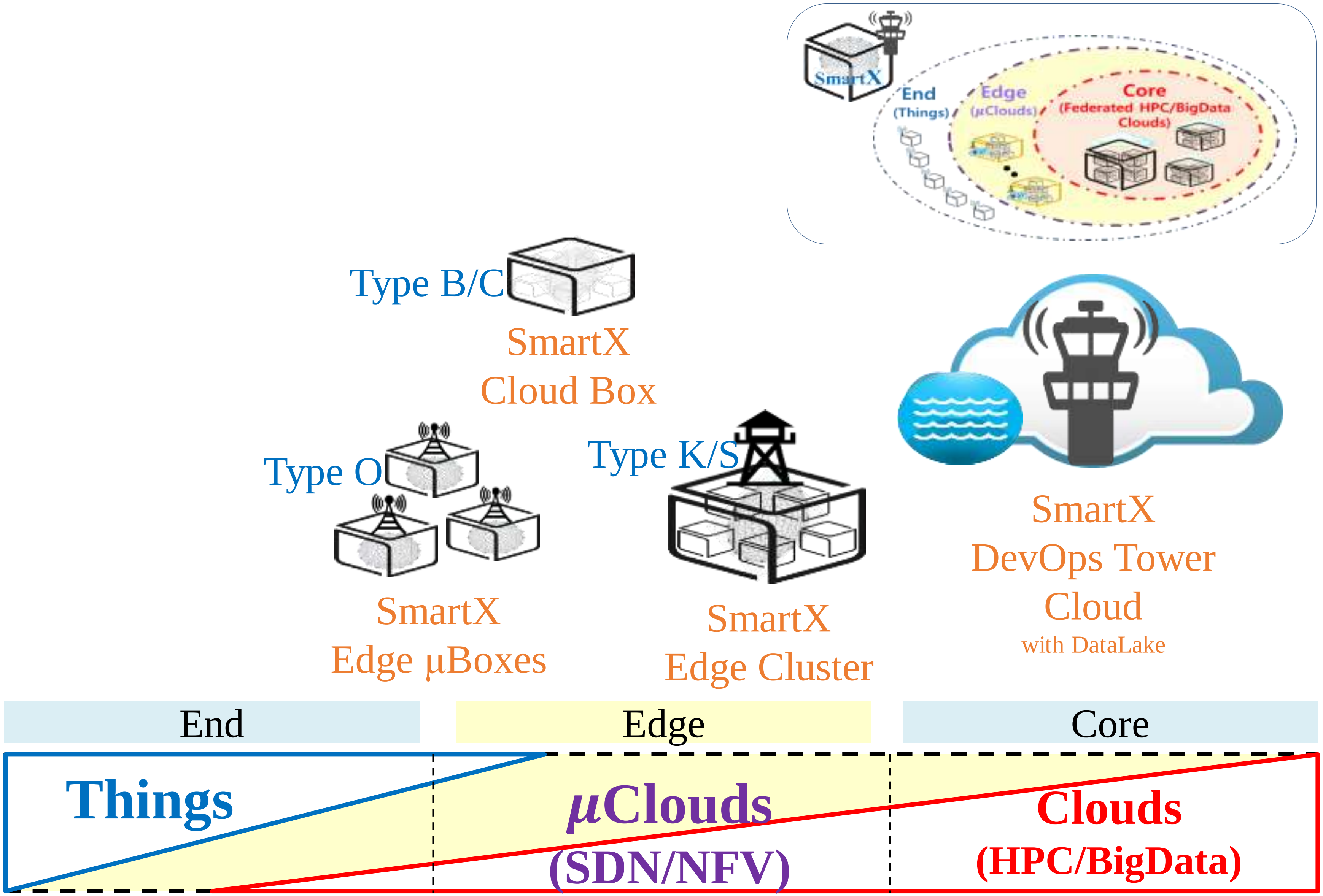


Lab **Theory**

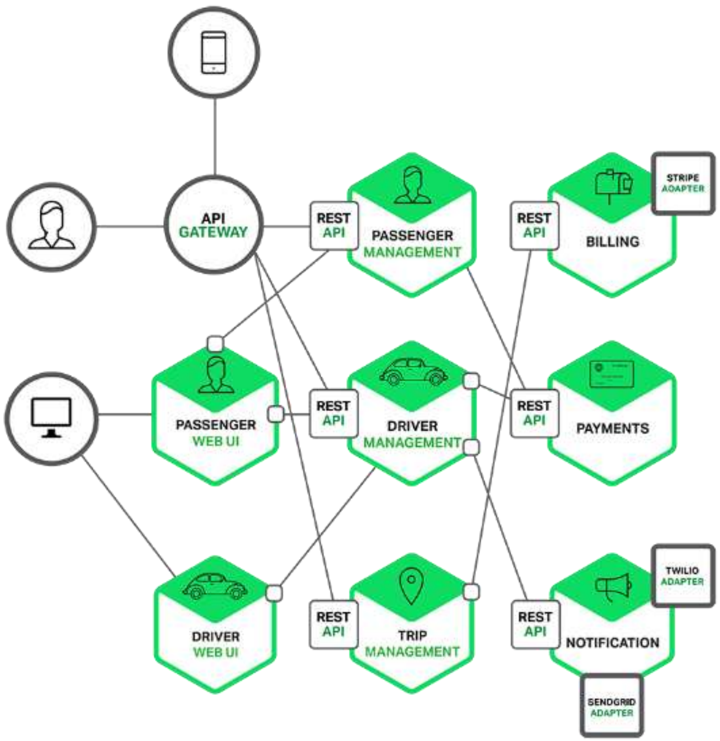

ApplicationS (Over)
Dev Ops
API
DEV OPS
Automation Service
Dev Ops
Infrastructure (Under)
Lifecycle &
Ecosystem

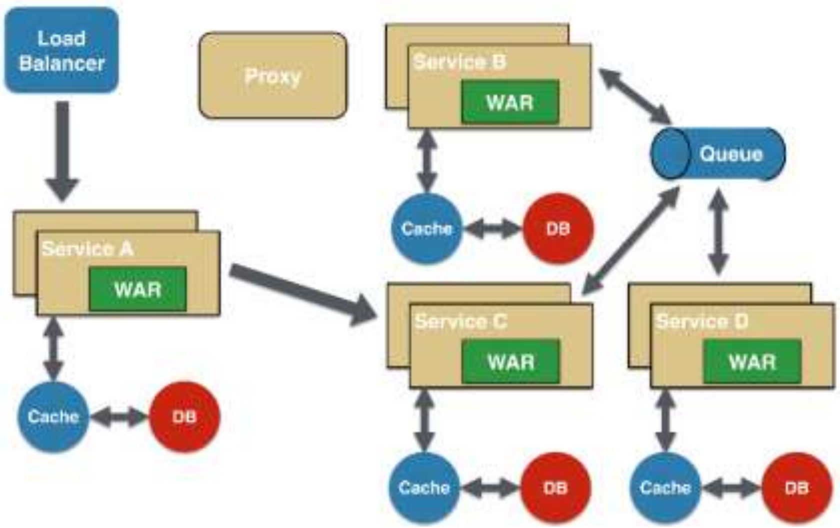

Resources - Workloads - Services
Box
Site
Functions
Inter-Connect


SmartX Automation
Security & Federation
Orchestration
Policy-driven Orchestration
Automated Scaling / Chaining
Flexible Control
Intelligence
BigData / FastData Intelligence
Visibility
Multi-level Visibility
Zero-touch Configuration
Provisioning
Visualization


SmartX
End
(Things)
Edge
(μClouds)
Core
(Federated HPC/BigData Clouds)
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
Things
μClouds
(SDN/NFV)
Clouds
(HPC/BigData)

- Software development technique based on **Collection of loosely coupled small-size services (i.e., functions)**
- Fine-grained services and lightweight protocols to improve modularity, create applications easier, and helps resiliency against architecture erosion





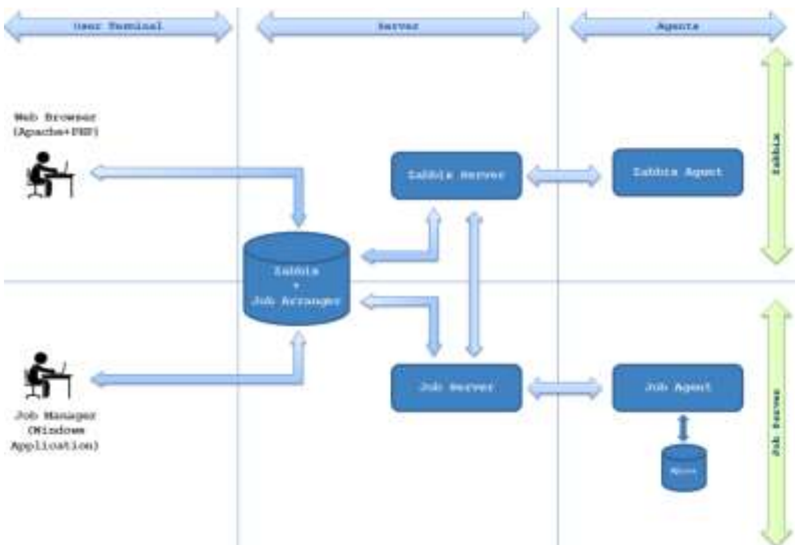
Visibility: TSDB (Time Series Database)

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

- Adopts a flexible notification mechanism
-
- User can configure & watch graph easily via Web GUI

- Consists of structured server and client
  - Client collects the monitoring data and send it to the Zabbix server
  - Server visualizes the data that is collected by the Zabbix Agent

ZABBIX



Zabbix Server Agent structure

Ubuntu boot procedure

Ubuntu boot up phases

1. BIOS

- When the computer begins execution, it starts by executing the firmware, and obtain the boot loader.

2. Boot loader

- The job of the boot loader is to begin the next phase, loading the kernel and an initial ram disk filesystem.

3. Kernel

- The kernel launches the init script inside the initrd file system, which loads hardware drivers and finds the root partition.

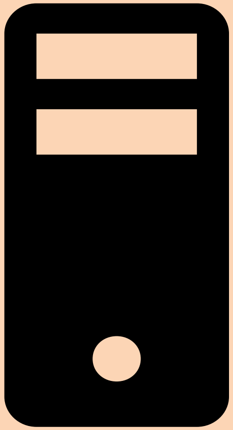
4. Upstart

- After the kernel is running, the remainder of the operating system is brought online.

Reference: https://wiki.ubuntu.com/Booting

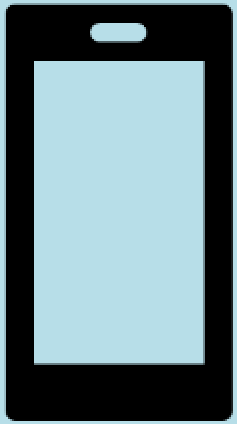
Remote OS installation on machines

Remote OS installation targets



Bare metal

- OS absent in the hardware
- For remote OS installation, the host doesn't have a decision-making power.



Mobile device

- OS already activated in the hardware
- From the standpoint of user, OS is installed automatically by host.

MAAS

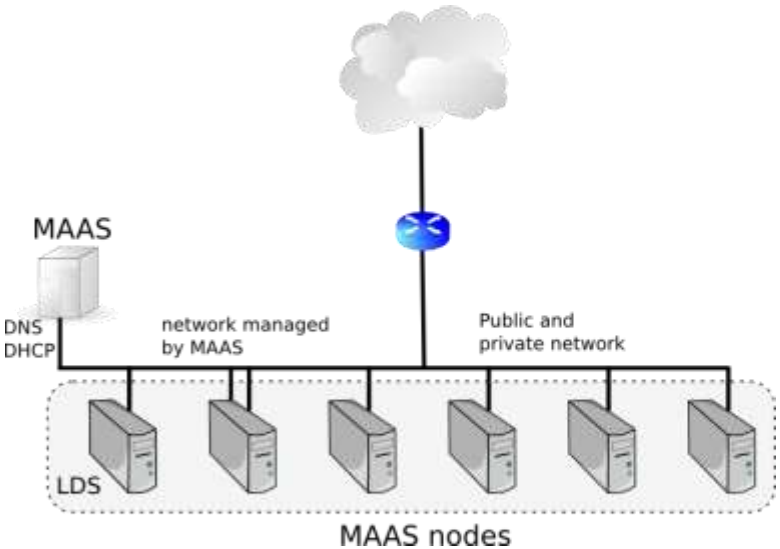
MAAS(<u>Metal</u> As A Service) is better suited to the <u>bare metal</u>.

puppet

cobbler

stacki

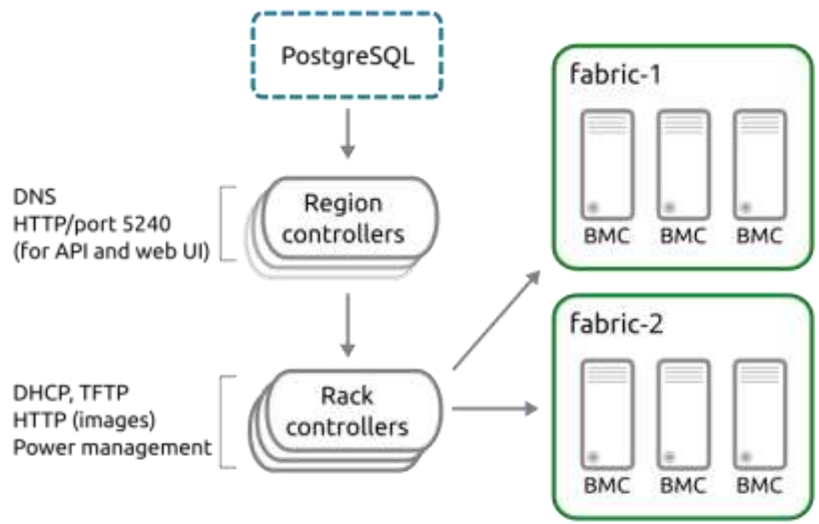
Provisioning: Ubuntu MAAS (Metal as a Service)

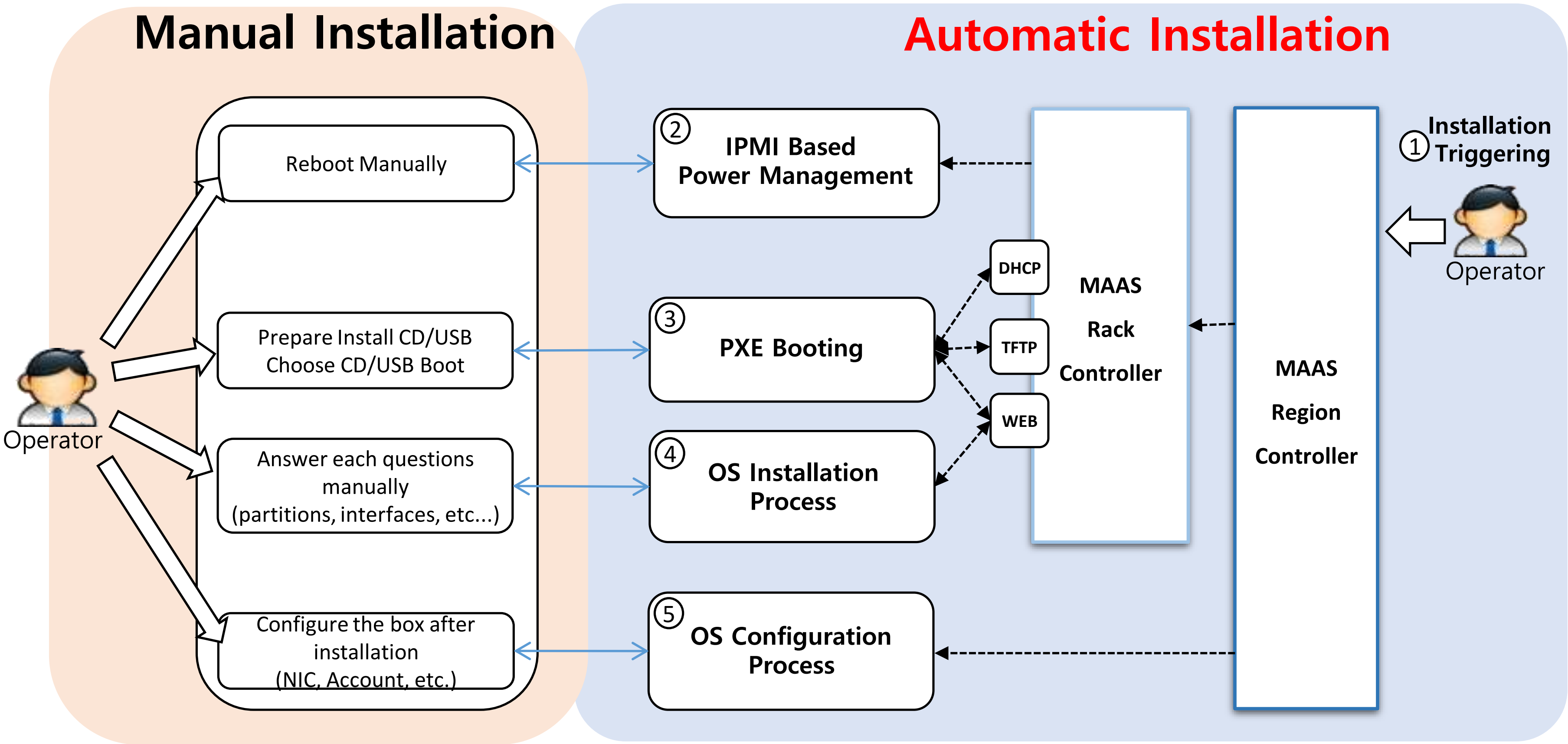


- Bear-metal machines can be quickly provisioned and destroyed; MAAS provides management of a large number of physical machines by creating a single resource pool
- MAAS can act as a standalone PXE services, provides Web GUI, supports various Linux distribution installation, …

MAAS Controller Architecture

- Region Controller: Deals with operator requests
- Rack Controller: Provide the high bandwidth services to multiple server racks + Cache OS install images





Warning!

Box Hardware Requirements for Automated Installation

- ***IPMI, Intel AMT, IBM HMC, ...**
- **PXE bootable with DHCP option**
- **Two Ethernet interfaces**

*IPMI: The intelligent Platform Management Interface. Remote hardware health monitoring and management system that defines the interfaces for use in monitoring

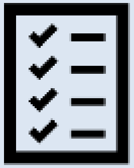
MAAS: Remote OS installation process

Remote OS installation Process

1. DHCP server contacted
2. Kernel, initrd received over TFPT
3. Machine boots
4. Initrd mounts a squashfs image over iSCSI

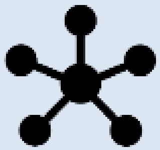
Enlistment

5. cloud-init runs enlistment scripts
6. Machine shuts down



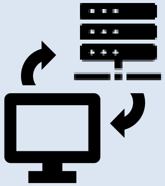
Commissioning

5. cloud-init runs commissioning scripts
6. Machine shuts down



Deployment

5. cloud-init triggers deployment
- Curtin installation script run
- Squashfs image placed on disk



Lab Practice

#0 - Lab Preparation (1/2)

Wired connection

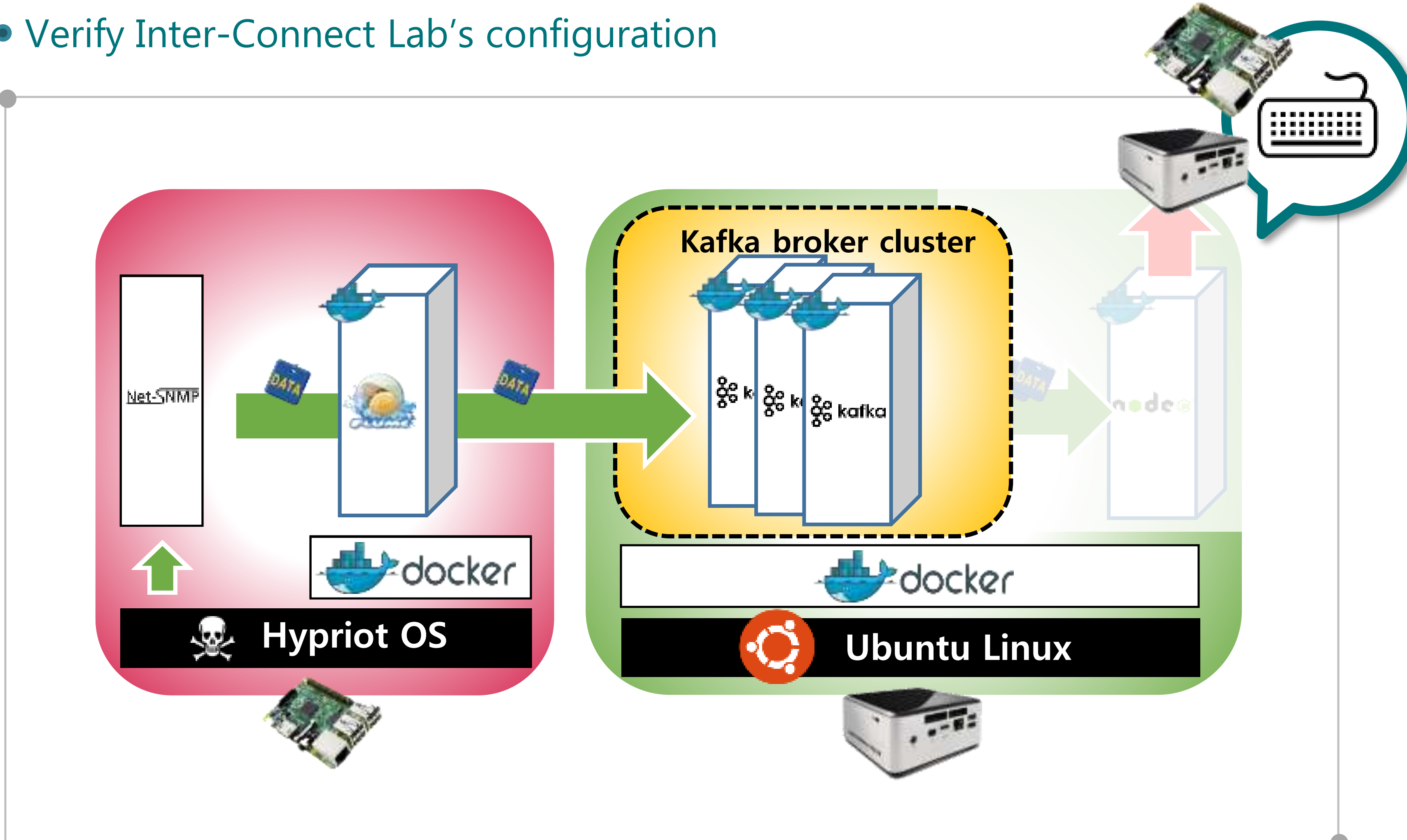
NAME: Raspberry Pi Model B (Pi)
CPU: ARM Cortex A7 @900MHz
CORE: 4
Memory: 1GB
SD Card: 32GB

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

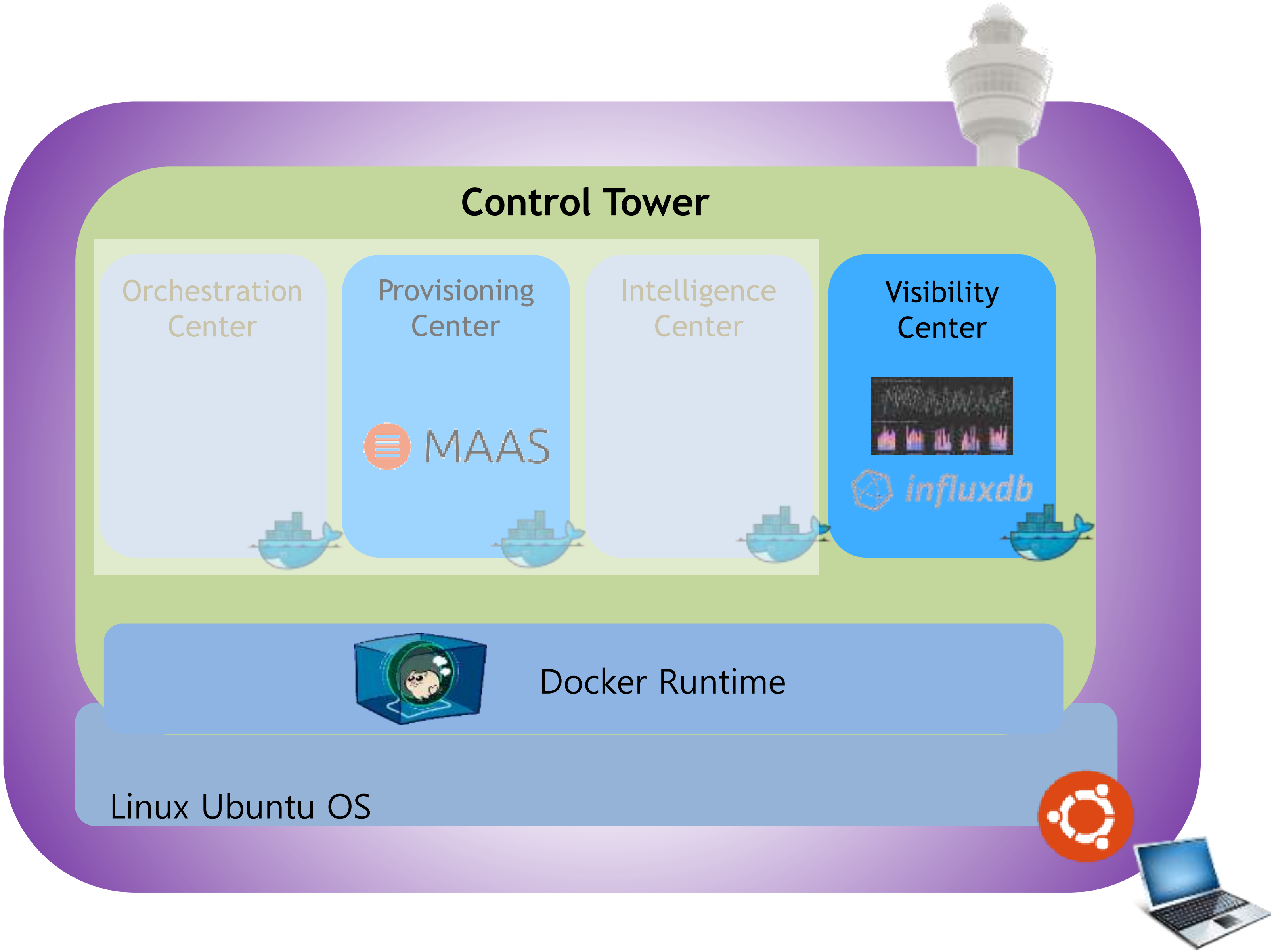
NAME: netgear prosafe 16 port gigabit switch(Switch)
Network Ports: 16 auto-sensing 10/100/1000 Mbps Ethernet ports

Verify Inter-Connect Lab's configuration

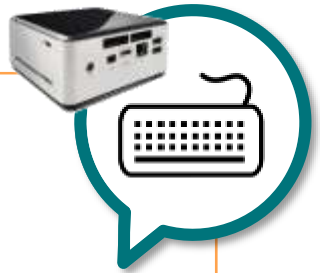


Are they working?

If you can see logs of resource status on console consumer, go ahead!


Control Tower
Orchestration Center
Provisioning Center
MAAS
Intelligence Center
Visibility Center
influxdb
Docker Runtime
Linux Ubuntu OS

#1 Run InfluxDB & Chronograf Containers on NUC



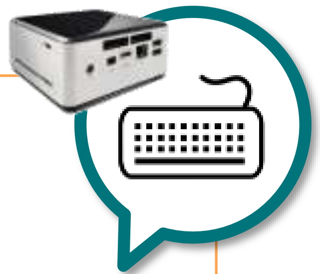
- Run InfluxDB Container

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

- Make and run Chronograf container

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

#2 Install python packages:for python Kafka consumer



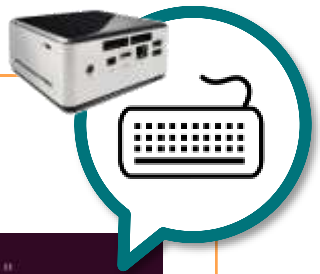
- Install python-pip

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

- Install python package

  $ sudo pip install requests
  $ sudo pip install kafka-python
  $ sudo pip install influxdb
  $ sudo pip install msgpack

#3 Broker to InfluxDB code:Modify & Run python code



- Open 'broker_to_influxdb.py' code

  $ vi ~/SmartX-mini/ubuntu-kafkatodb/broker_to_influxdb.py

```
cmd ="curl -XPOST 'http://localhost:8086/query' --data-urlencode 'q=CREATE DATABASE 'Labs''"
subprocess.call([cmd], shell=True)

timeout = 100
actual_data=[]

consumer = KafkaConsumer('resource',bootstrap_servers=['[redacted]:9091'])
partitions = consumer.poll(timeout)
while partitions == None or len(partitions) == 0:

        consumer = KafkaConsumer('resource', bootstrap_servers=['[redacted]:9091'])
        message = next(consumer)
        print(message.value)
```

Modify to your nuc IP Address

Modify to your nuc IP Address

- Run python code

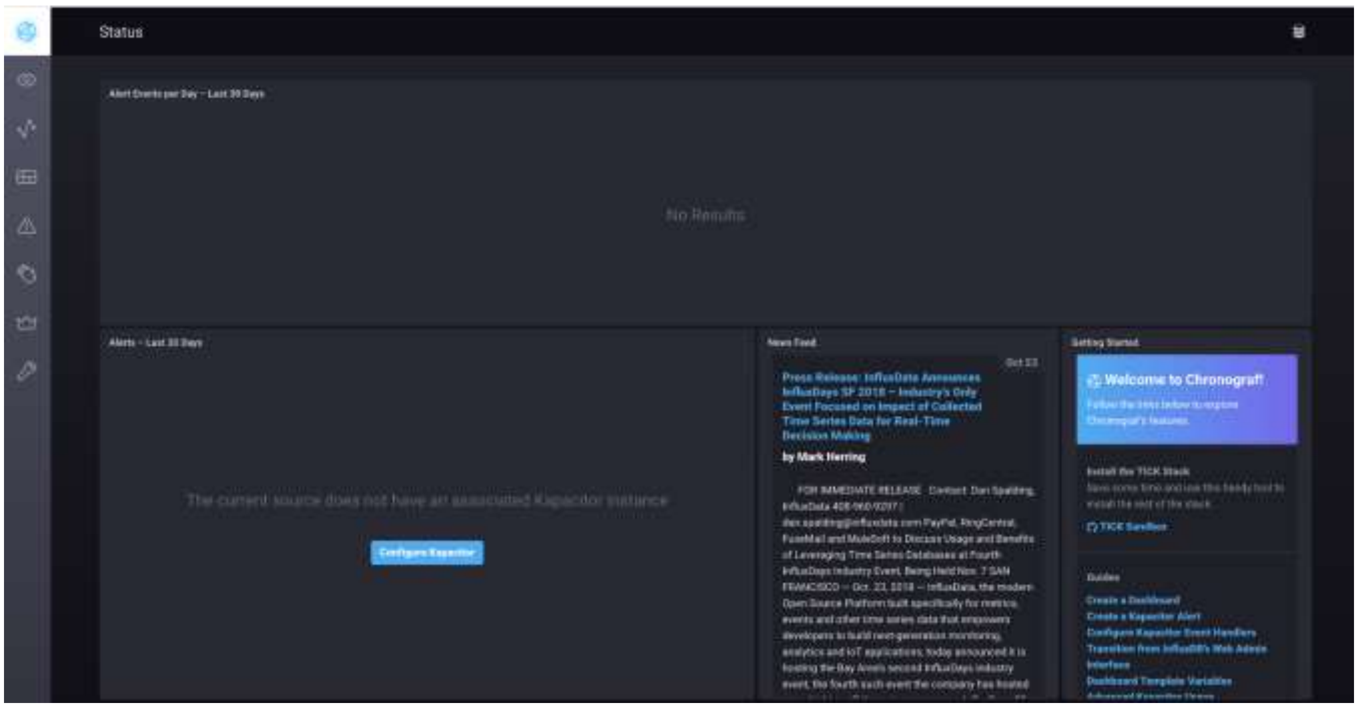
  $ sudo sysctl –w fs.file-max=100000

  $ ulimit –S –n 2048

  $ python ~/SmartX-mini/ubuntu-kafkatodb/broker_to_influxdb.py

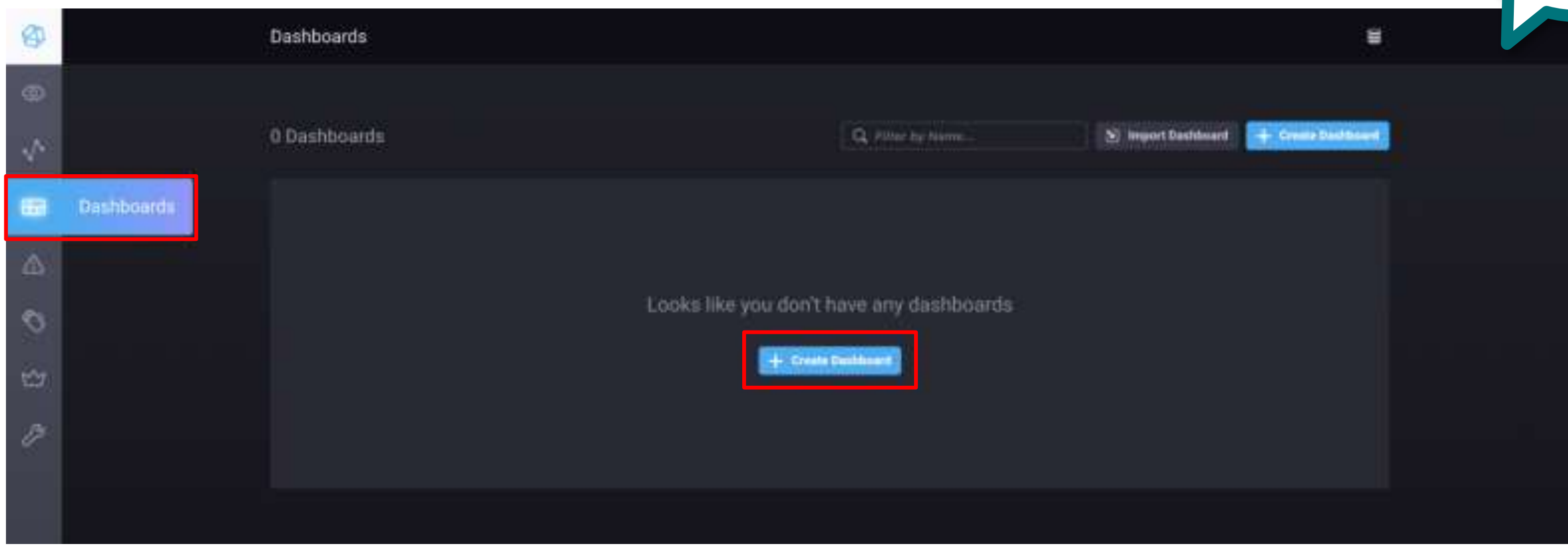
#4 Configure Chronograf Dashboard (1/4)

- Open Web browser and connect to Chronograf Dashboard
  http://<NUC IP>:8888



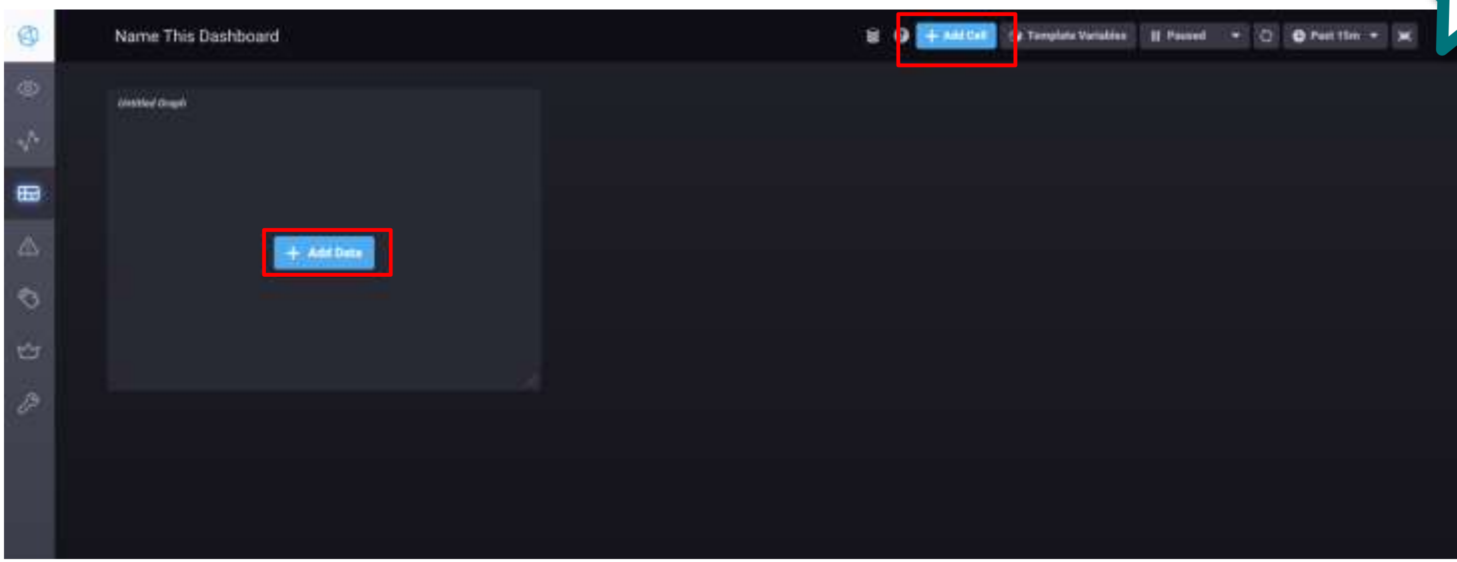
#4 Configure Chronograf Dashboard (2/4)

- Create Dash board

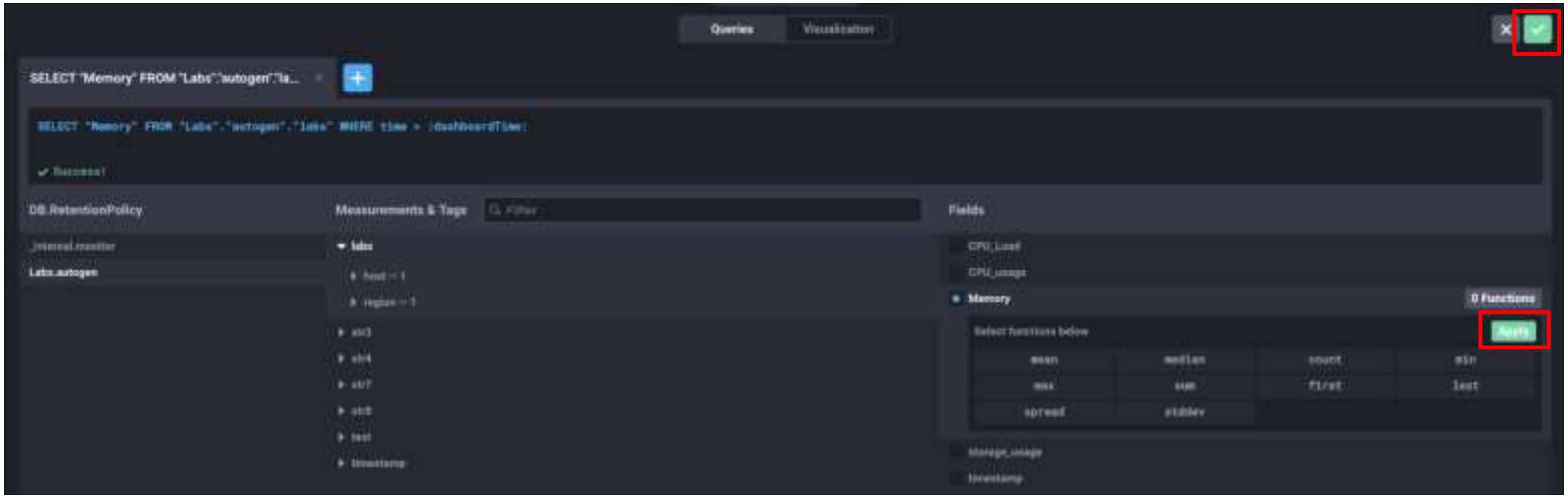


#4 Configure Chronograf Dashboard (3/4)

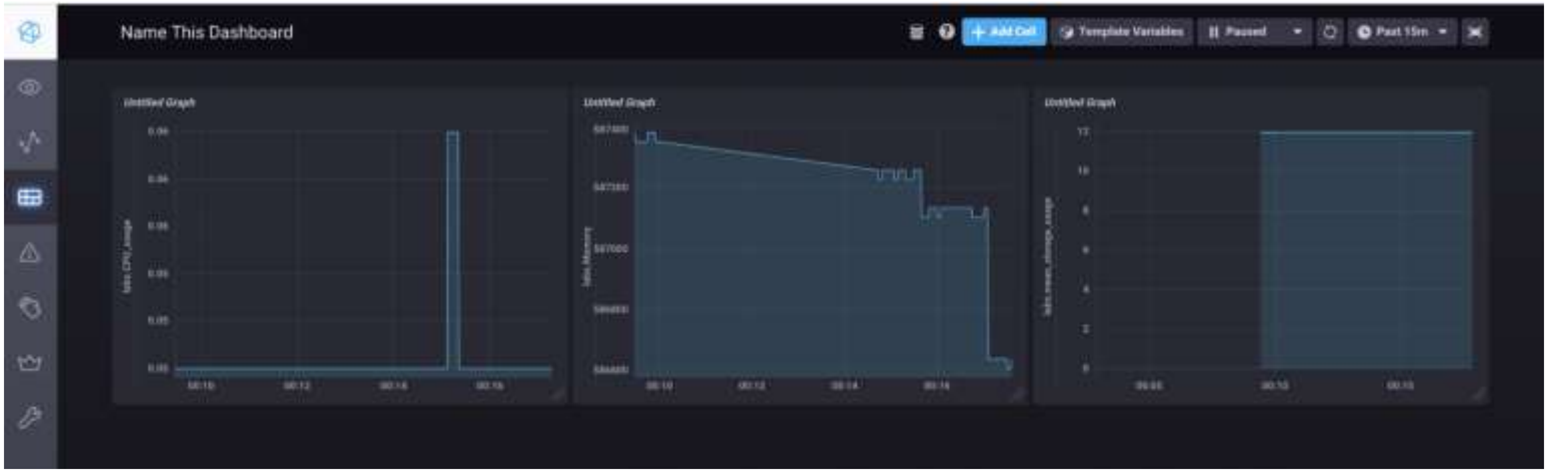
- Add Cell & Data



- Add Data

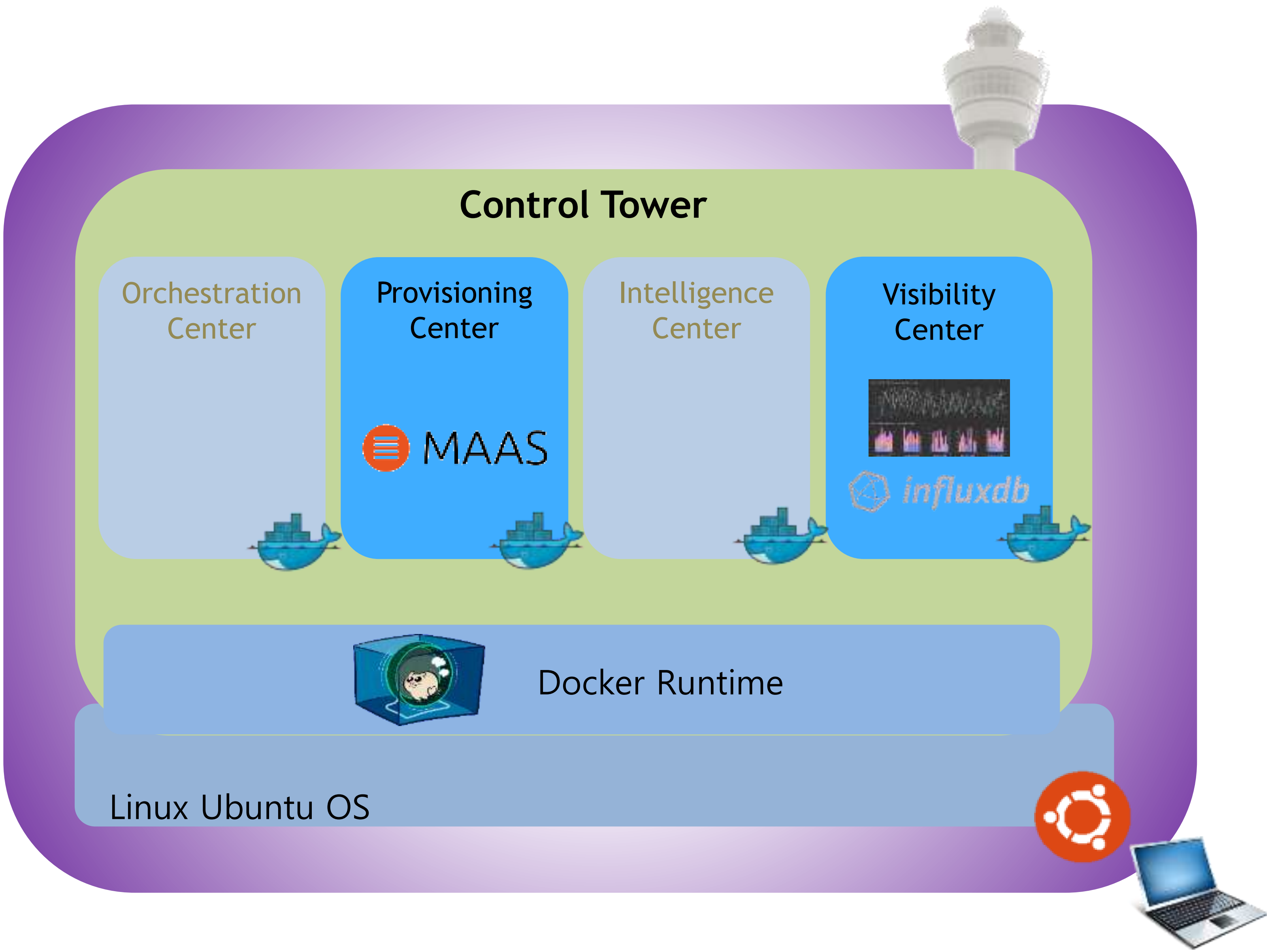

Queries
Visualization
SELECT "Memory" FROM "Labs"."autogen"."la...
DB.RetentionPolicy
Measurements & Tags
Fields
CPU_Load
CPU_usage
Memory
0 Functions
mean
median
count
min
max
sum
first
last
spread
stddev

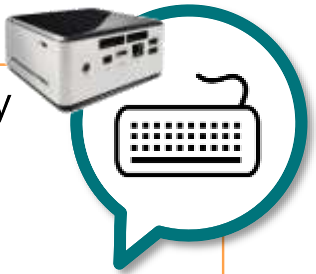
- We can see the changes of values from database



Note:

Remaining Lab practice requires special Box resources with **IPMI or similar Remote Power Management, PXE Boot support**


Control Tower
Orchestration Center
Provisioning Center
MAAS
Intelligence Center
Visibility Center
influxdb
Docker Runtime
Linux Ubuntu OS



- From target NUC, go to BIOS, turn on PXE and set network boot priority
- NUC reboot (to apply BIOS changes)
- Install MAAS server

  $ sudo apt update

  $ sudo apt install maas

- Initiate MAAS server

  $ sudo maas init

- Login to the MAAS UI at:

  http://<your.maas.ip>:5240/MAAS

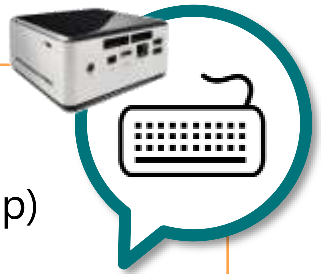
- From the MAAS UI, you need to make user configurations

- *Region name
- Ubuntu images

- Turn on DHCP

- Go to the "Subnets" tab, select the VLAN for which you want to enable DHCP
- From the "Take action" button select "Provide DHCP"

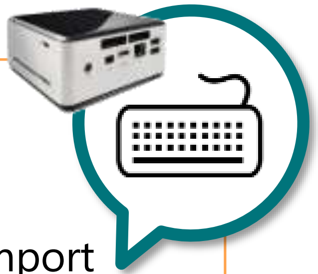
*region : Organizational unit. Contains all information about all machines running



- Enlist the NUC
  - Set all the servers to PXE boot
  - NUC reboot (When hardware is initialized, all software operations stop)
  - Check the NUC appear in MAAS
  - If the NUC does not support IPMI based BMC, edit them and enter their BMC details

- Commission the NUC
  - Go to “machine interface”, “configuration” and set “power type” to “manual”
  - From the “take action” button, select “commission”
  - NUC reboot (When a new kernel is installed, the box must be rebooted as to prevent the removed kernel from being loaded which will halt the NUC operation)

- Deploy the NUC
  - From the “take action” button, choose “deploy” and click “view this page”
  - From SSH keys, choose source and click upload
  - Set “id_rsa.pub” as a public key and click “import”



- Create SSH key

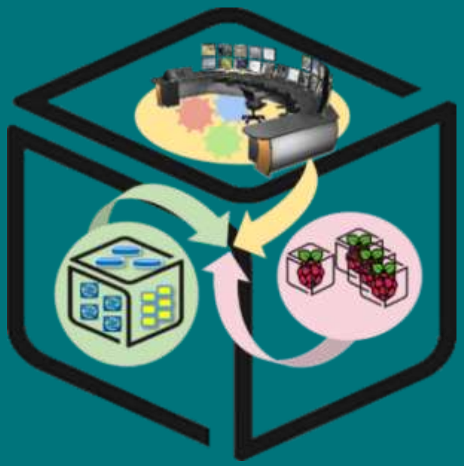
  $ sudo ssh-keygen

  $ sudo cat ~/.ssh/id_rsa.pub

  Copy the outcome , press “upload” and paste it on “User ID” and import

- Deploy

- From “take action”, choose “deploy”
- OS remote Install complete



Lab Review

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

01

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

02

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

Thank You for Your Attention
Any Question?

Mini@smartx.kr