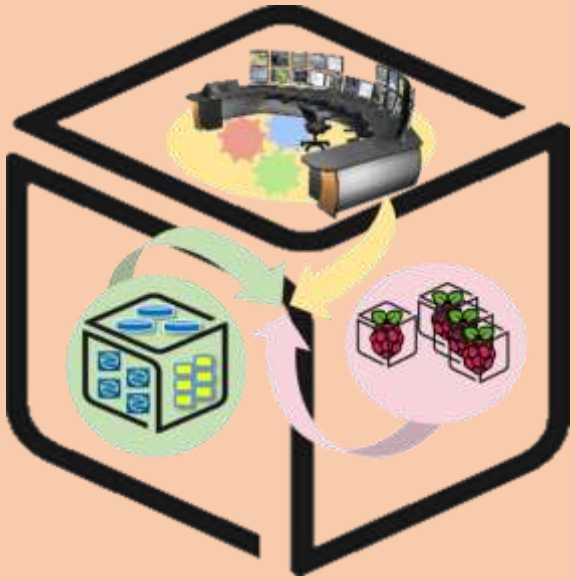
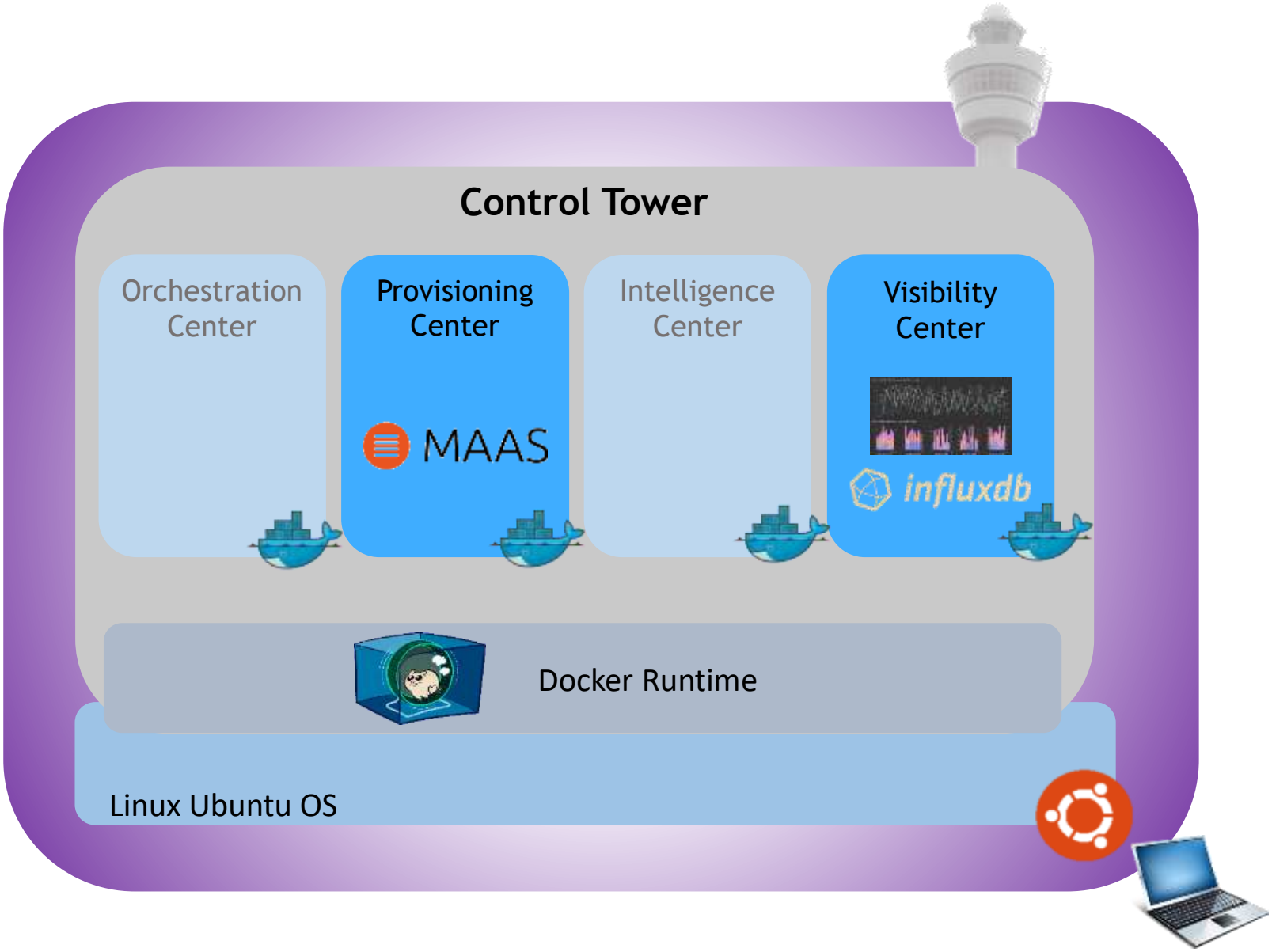
Computer Systems For AI-inspired Cloud Theory & Lab.

Lab #3: Tower

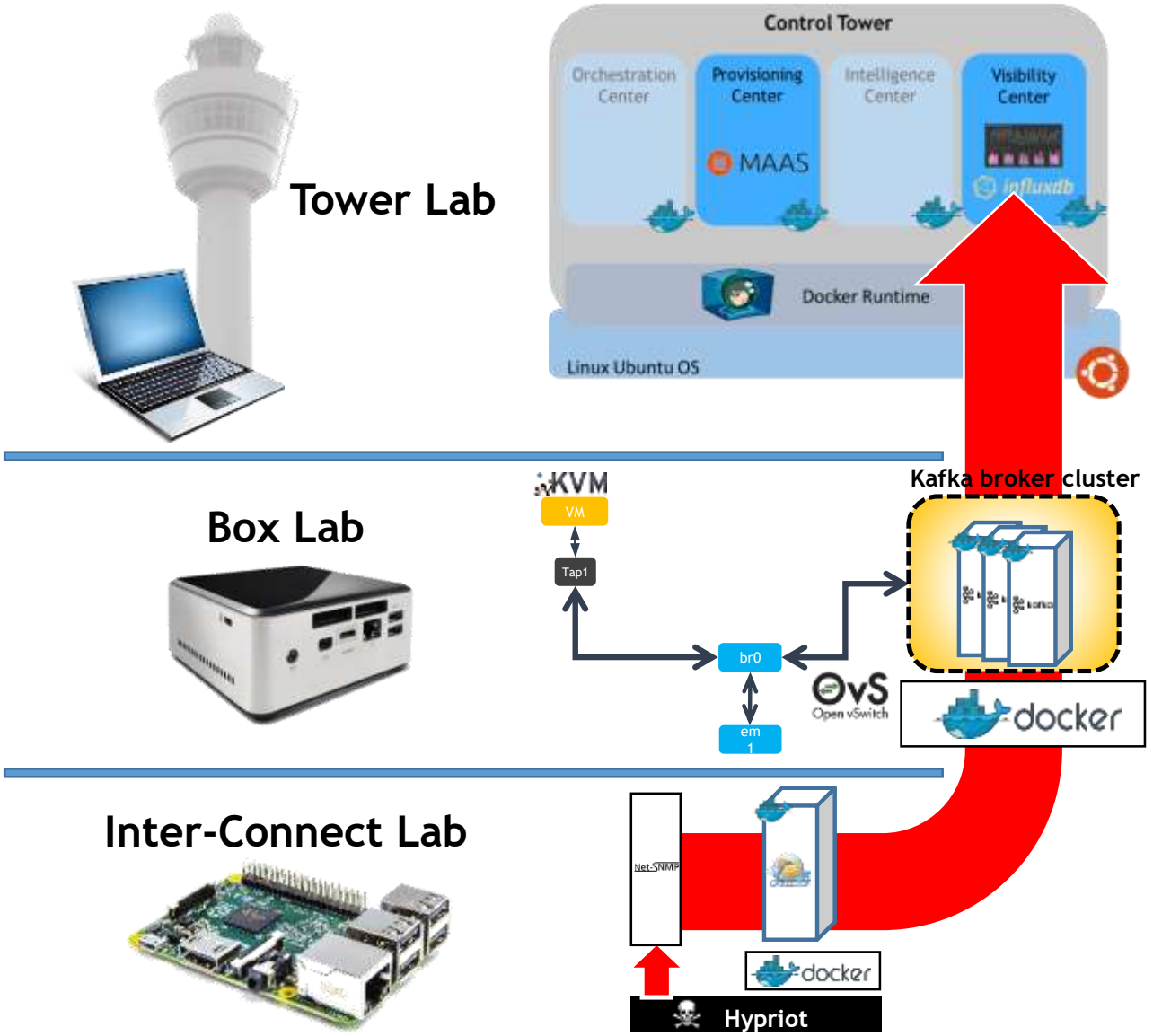


https://github.com/SmartX-Labs/SmartX-Mini-MOOC

Tower Lab: Concept

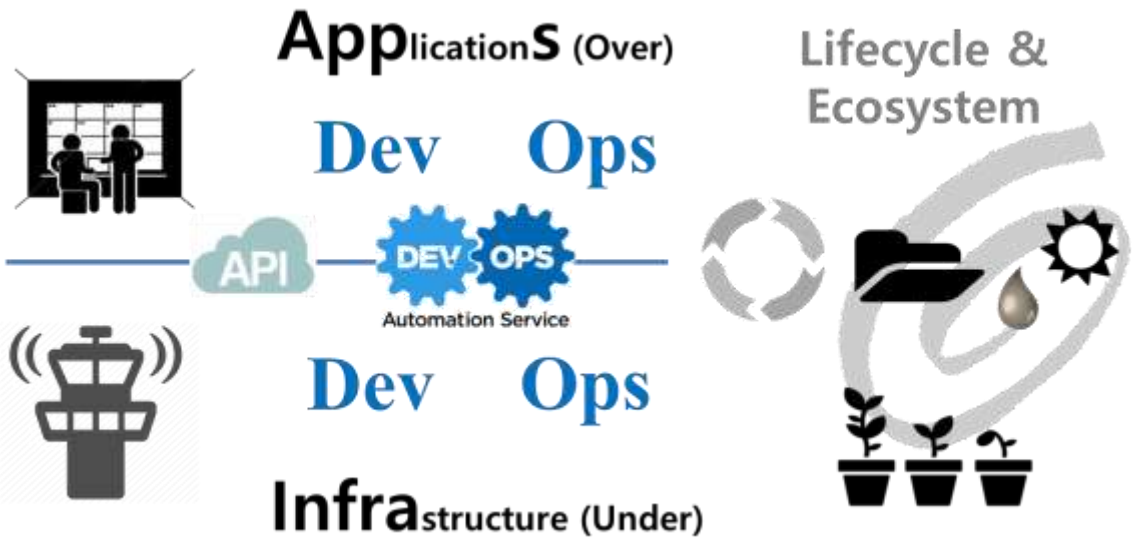


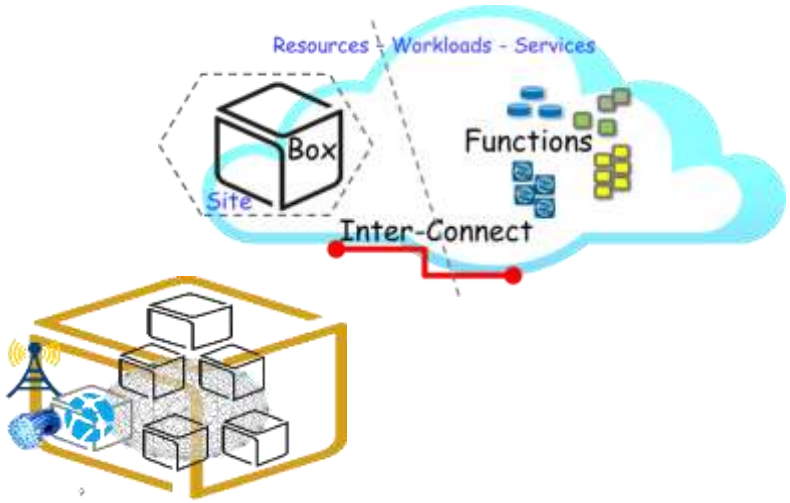
SmartX Labs #1~#3: Relationship

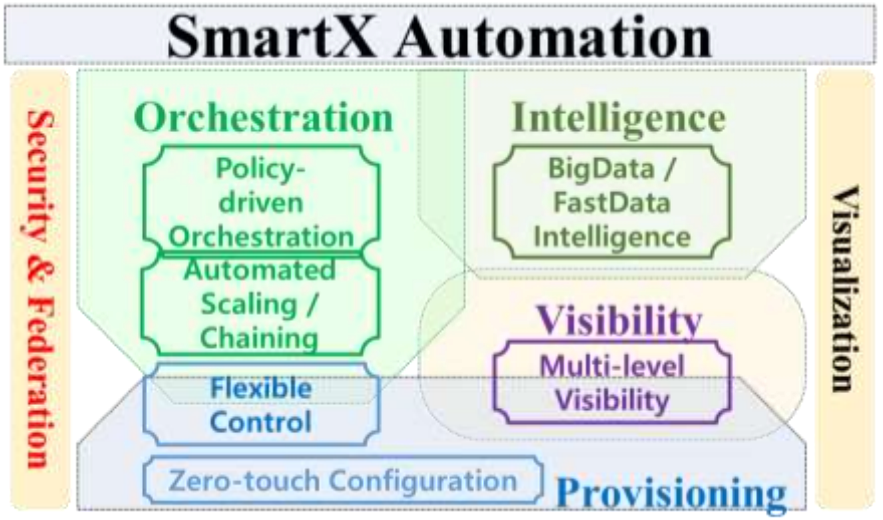


Theory

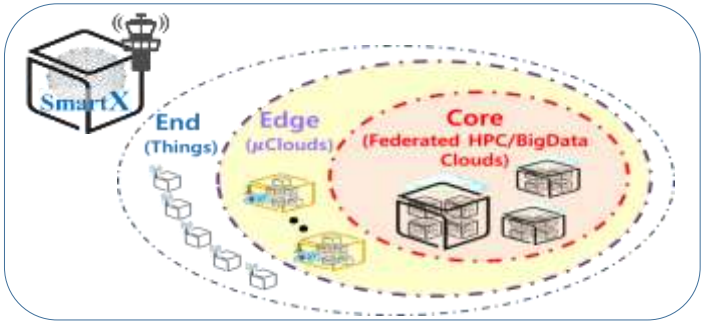
SmartX Automation Framework







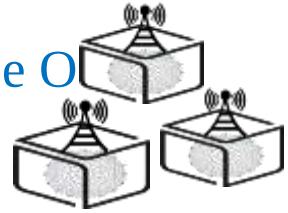
SmartX Composable Playground & Boxes



Type B/C

SmartX
Cloud Box

Type O

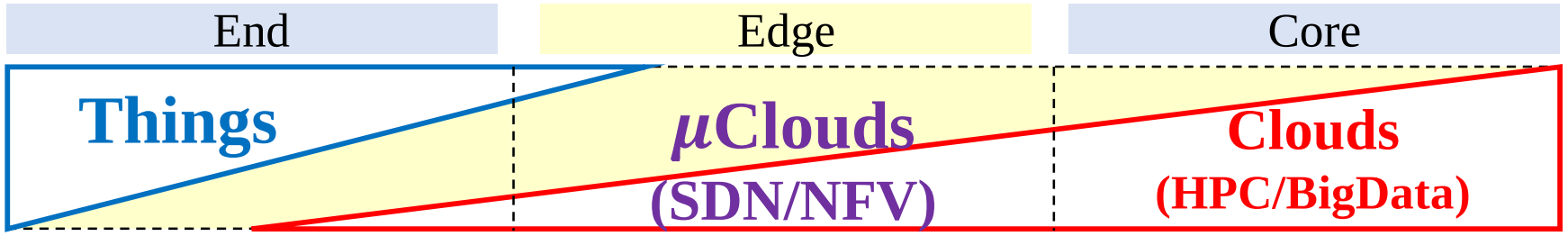


SmartX
Edge μBoxes

Type K/S

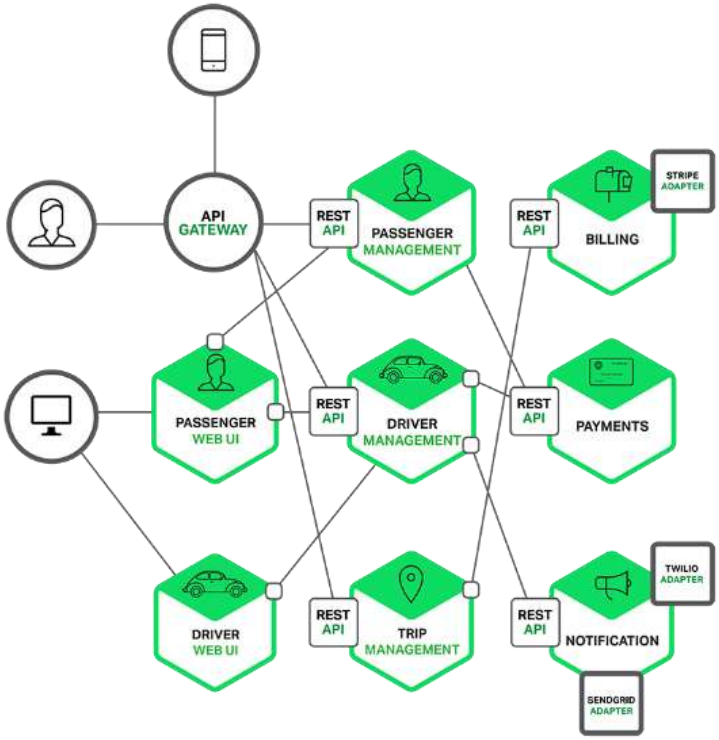
SmartX
Edge Cluster

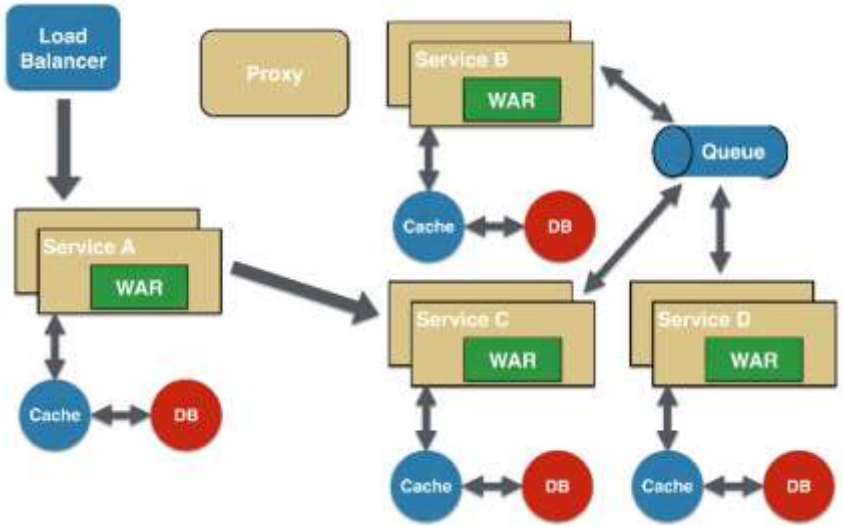
SmartX
DevOps Tower
Cloud
with DataLake



Container-based MSA (MicroServices Architecture)

- 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

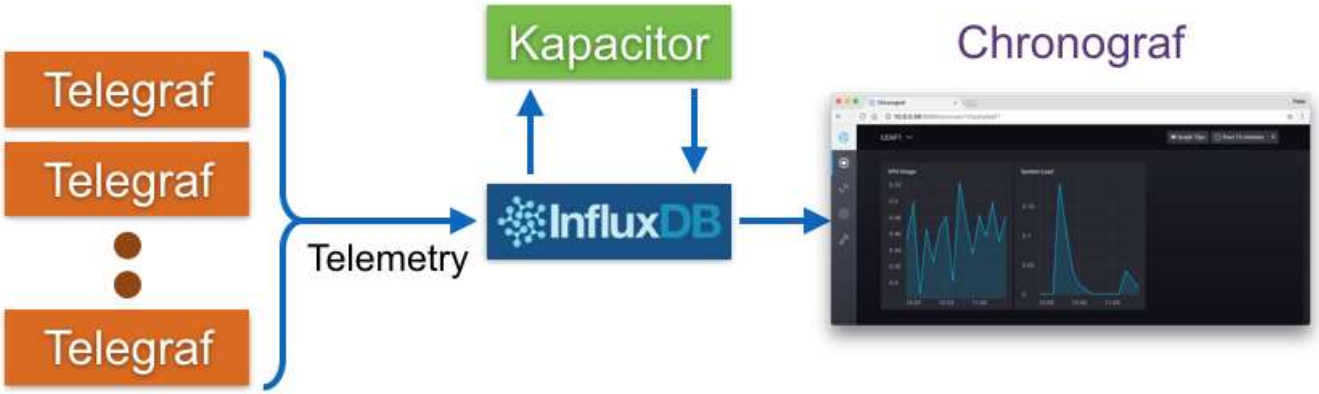




Visibility: TSDB (Time Series Database)

influxdb
cassandra
druid
riakTS
InfluxDB
elasticsearch

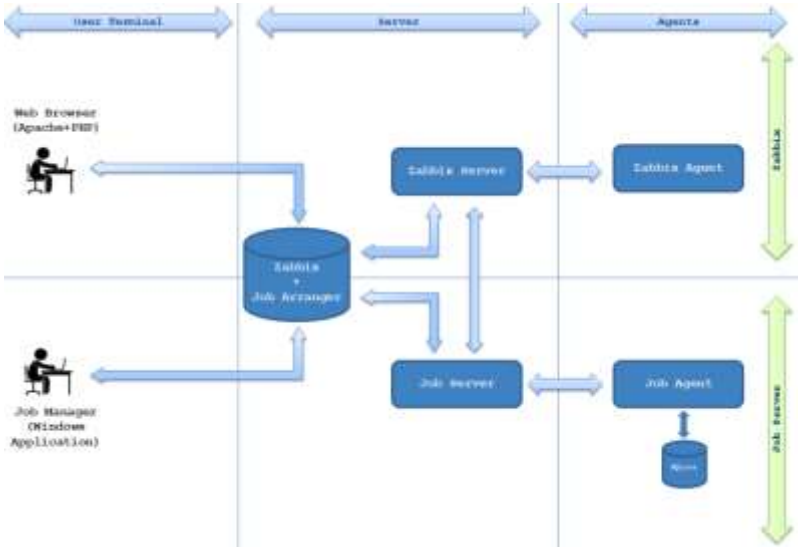
- Time series data is arrays of numbers indexed by time.
- In some fields these time series are called profiles, curves, or traces.



Visibility: Zabbix Distributed Monitoring

- 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

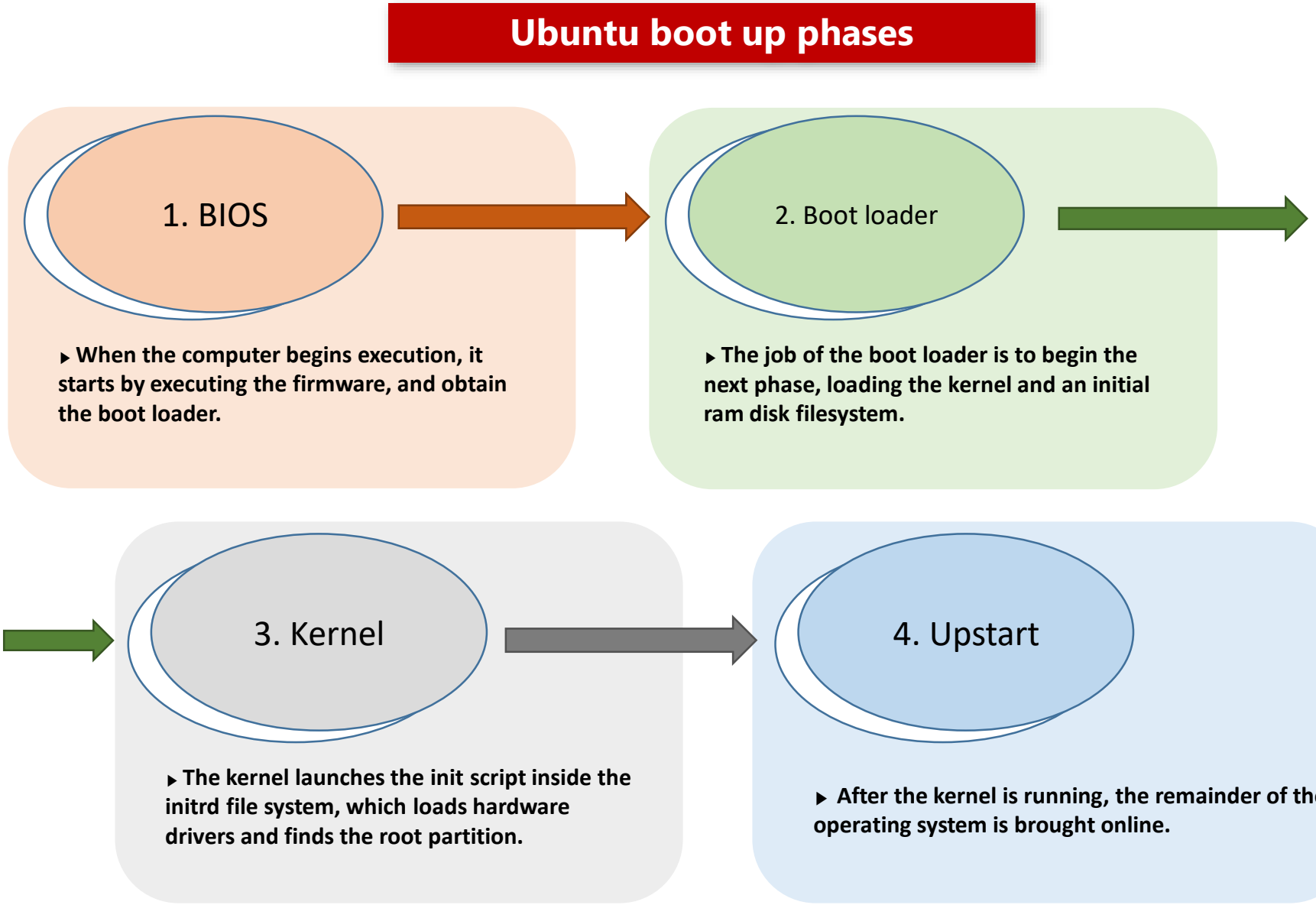
ZABBIX



Zabbix Server Agent structure

Ubuntu boot procedure

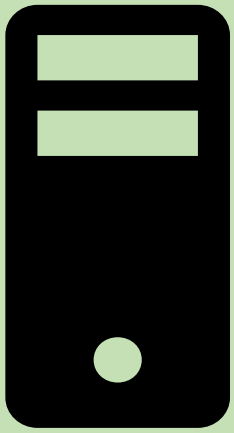
Ubuntu boot up phases



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

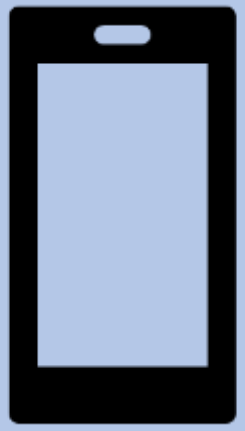
Remote OS installation on machines

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

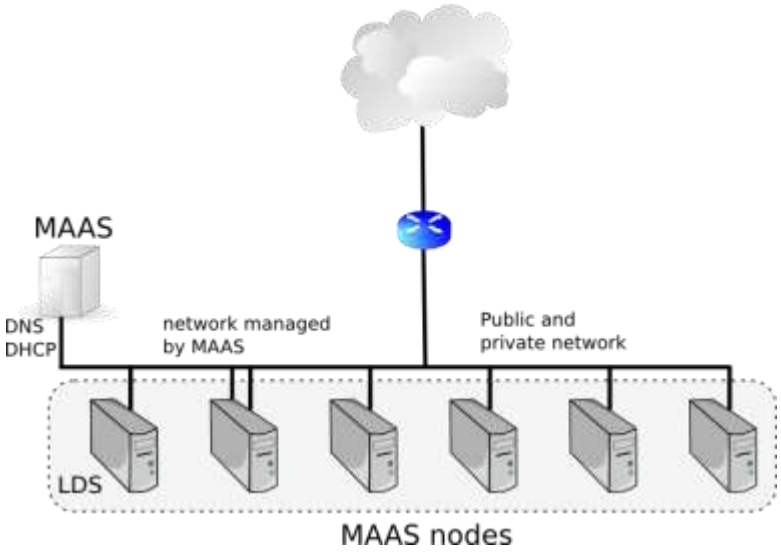
MAAS(Metal As A Service) is better suited to the bare metal.

puppet

cobbler

stacki

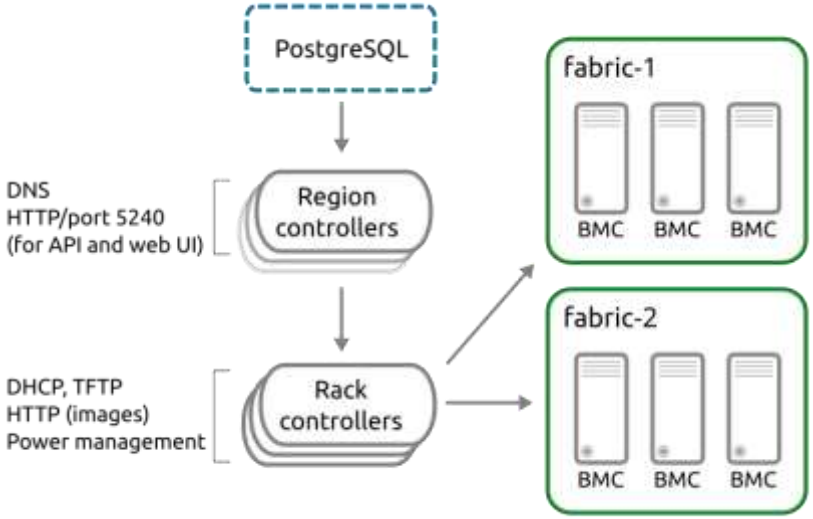
Provisioning: Ubuntu MAAS (Metal as a Service)



- 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, …

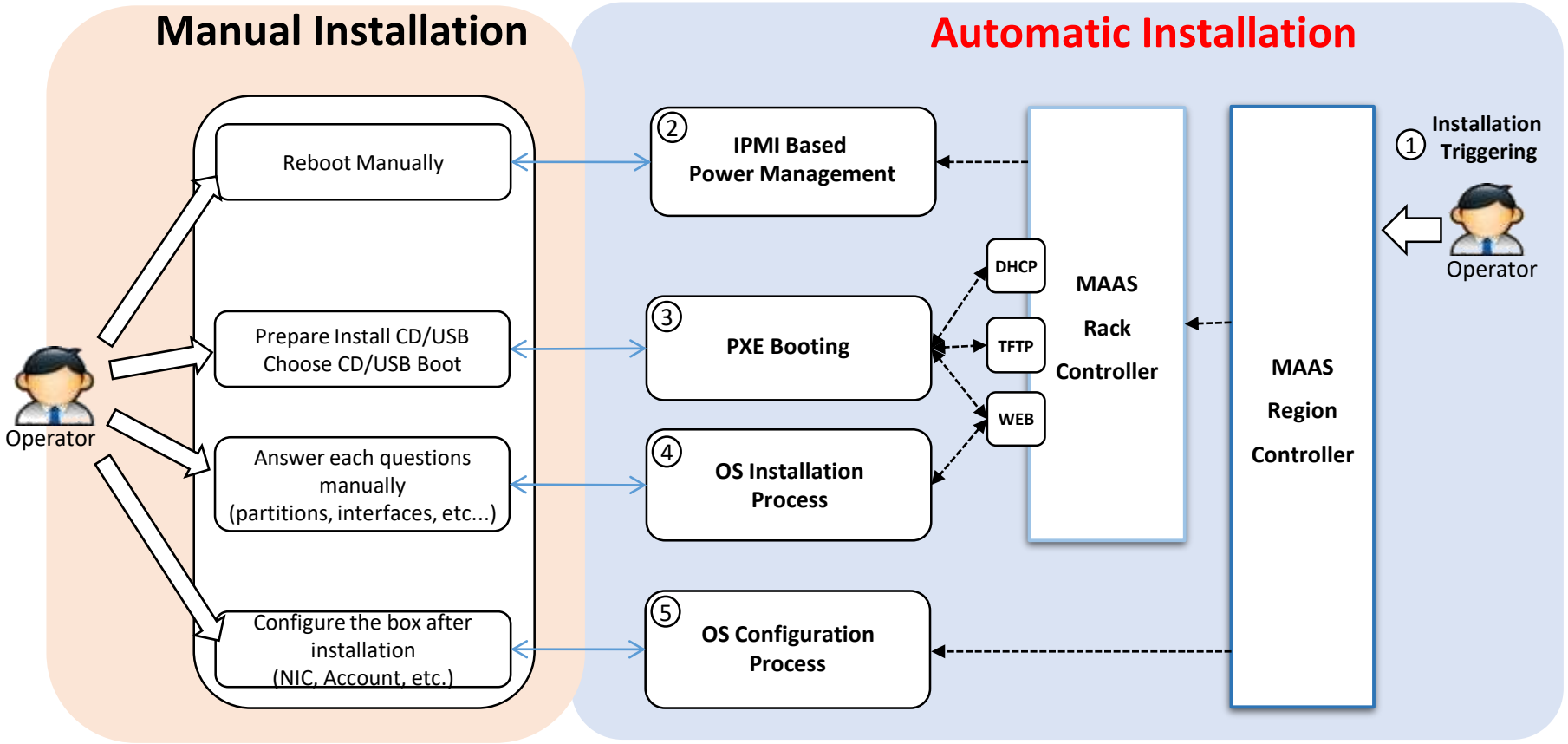
MAAS Controller Architecture

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



Reference: https://docs.maas.io/2.4/en/intro-concepts#controllers

MAAS: OS Automated Installation



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 TFTP
3. Machine boots
4. Initrd mounts a squashfs image over HTTP

Enlistment

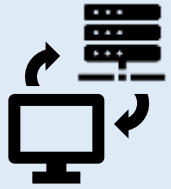
5. cloud-init runs enlistment scripts
6. Machine shuts down

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



Practice

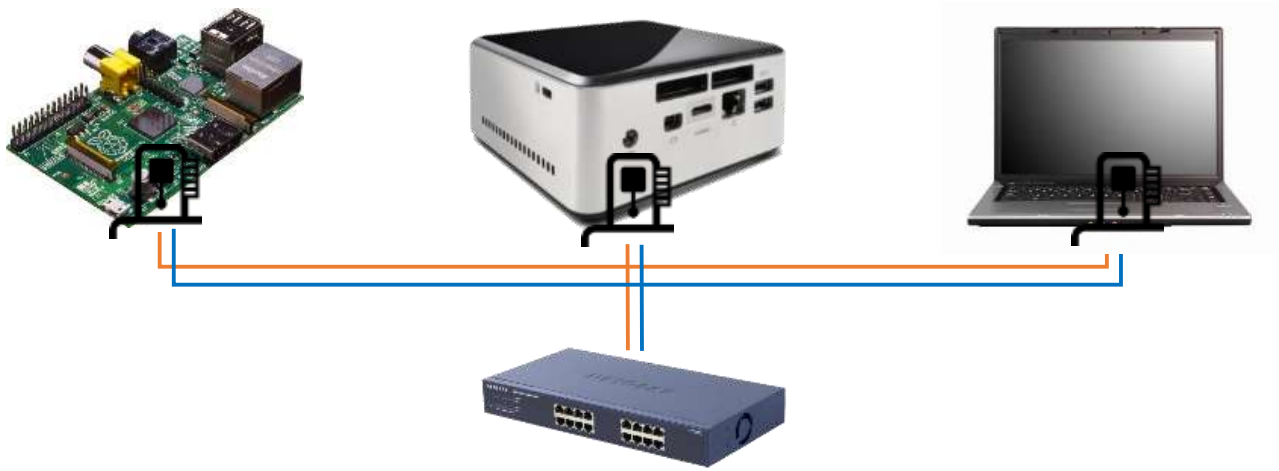
#0 Lab Preparation (1/2)

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

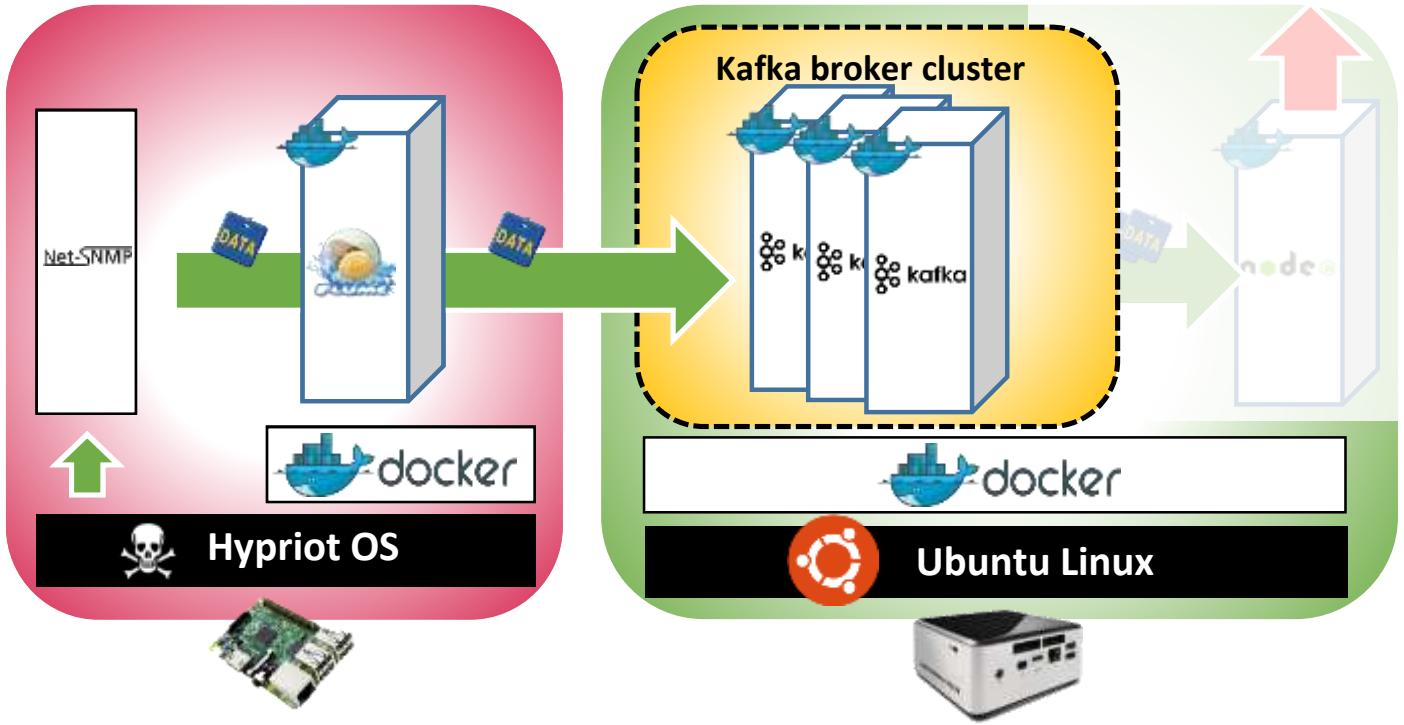
NAME: NT900X3A
CPU: i5-2537U @1.40GHz
CORE: 2
Memory: 4GB DDR3
HDD: 128GB



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

#0 Lab Preparation (2/2)

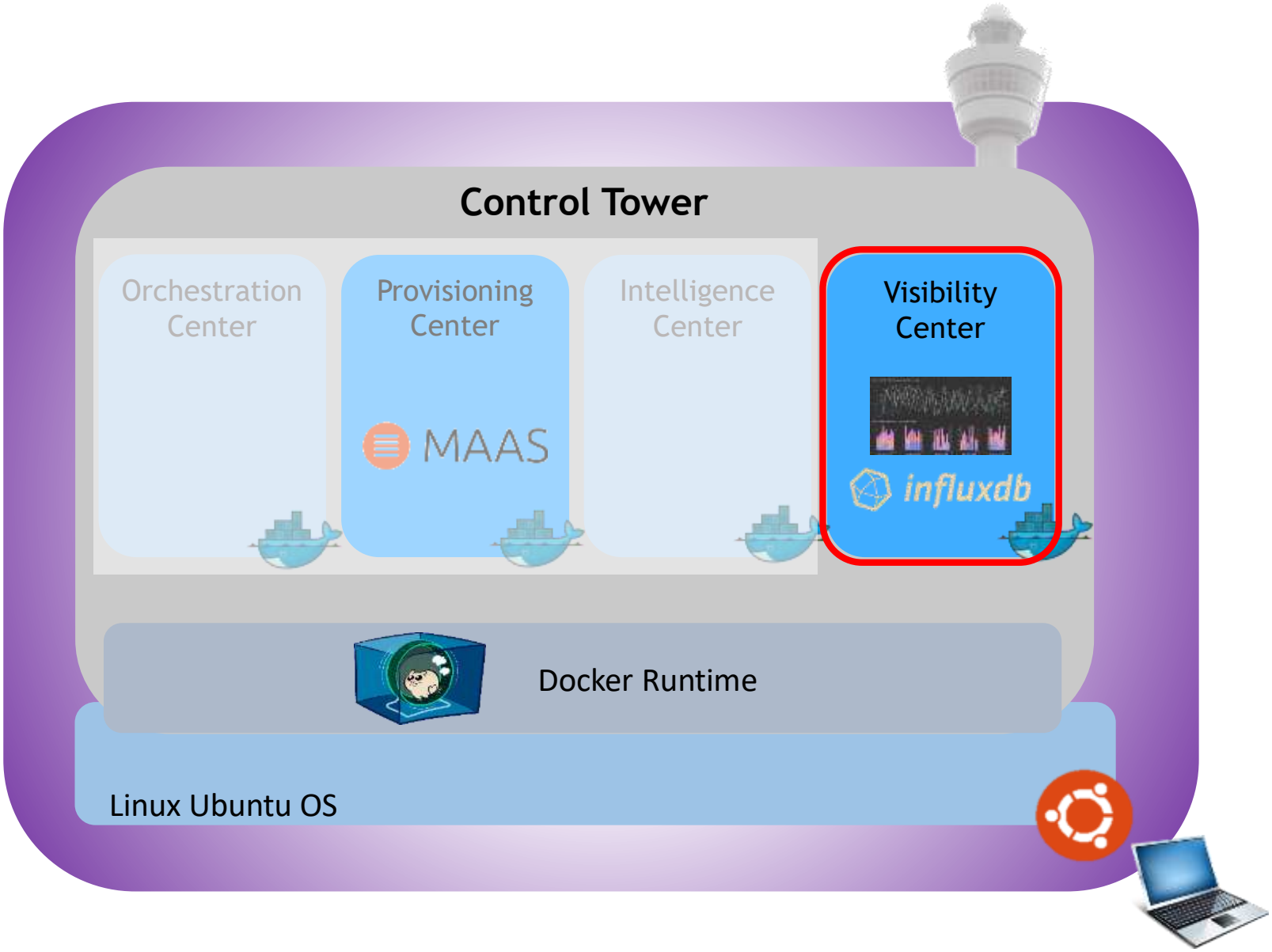
- Verify Inter-Connect Lab's configuration



Are they working?

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

Visualization of Resource Visibility



#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 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

#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=[]
                                                    Modify to your nuc IP Address
consumer = KafkaConsumer('resource',bootstrap_servers=[ [redacted]:9091'])
partitions = consumer.poll(timeout)
while partitions == None or len(partitions) == 0:
                                                    Modify to your nuc IP Address
        consumer = KafkaConsumer('resource', bootstrap_servers=[ [redacted]:9091'])
        message = next(consumer)
        print(message.value)
```

- Run python code

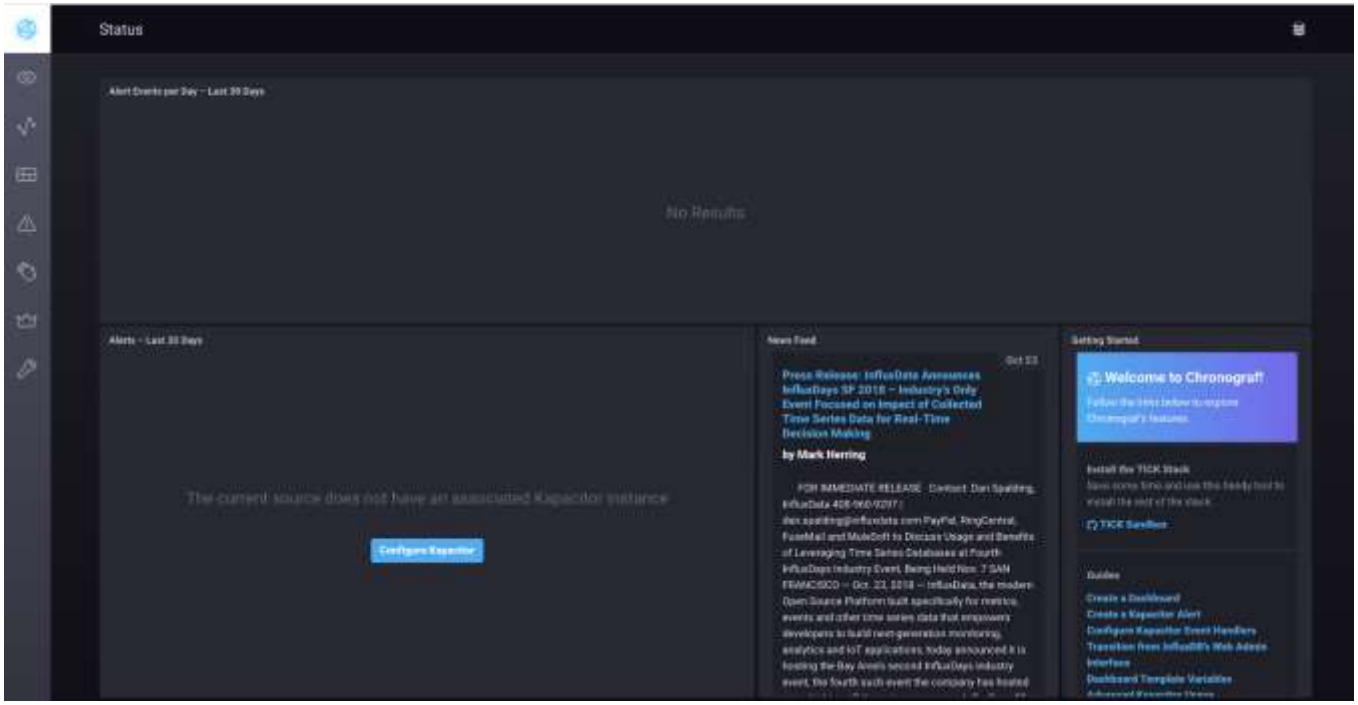
  $ sudo sysctl -w fs.file-max=100000

  $ ulimit -S -n 2048

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

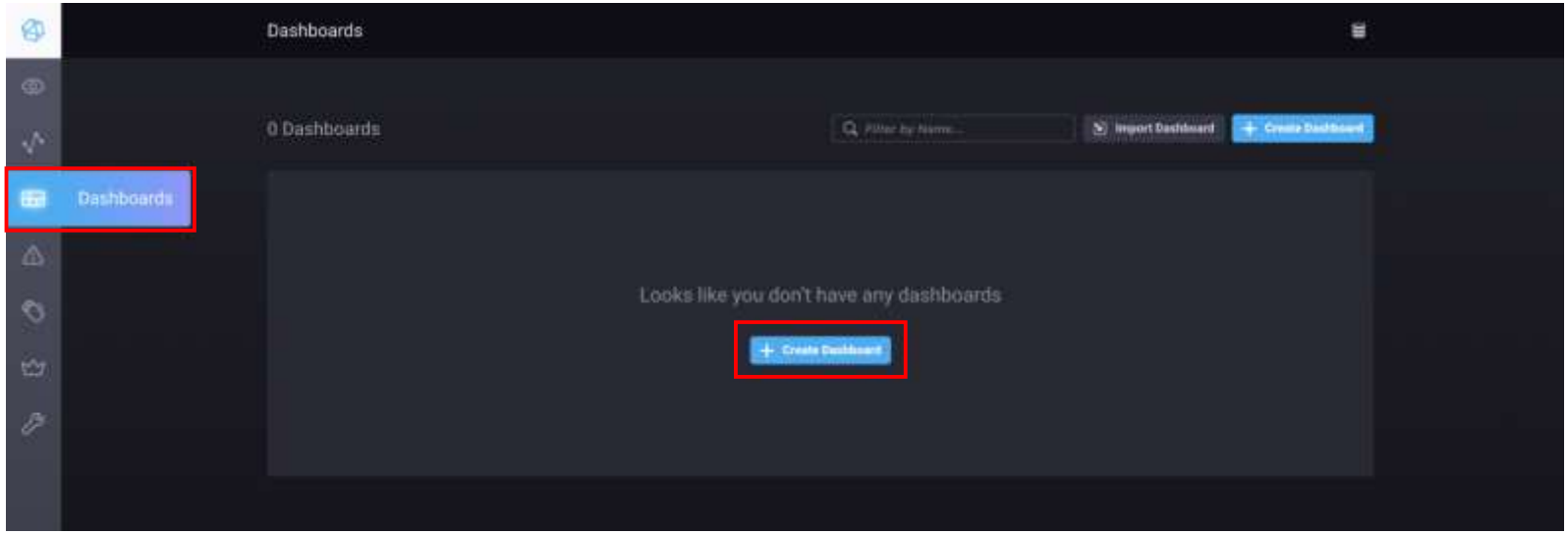
#4 Configure Chronograf Dashboard (1/4)

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



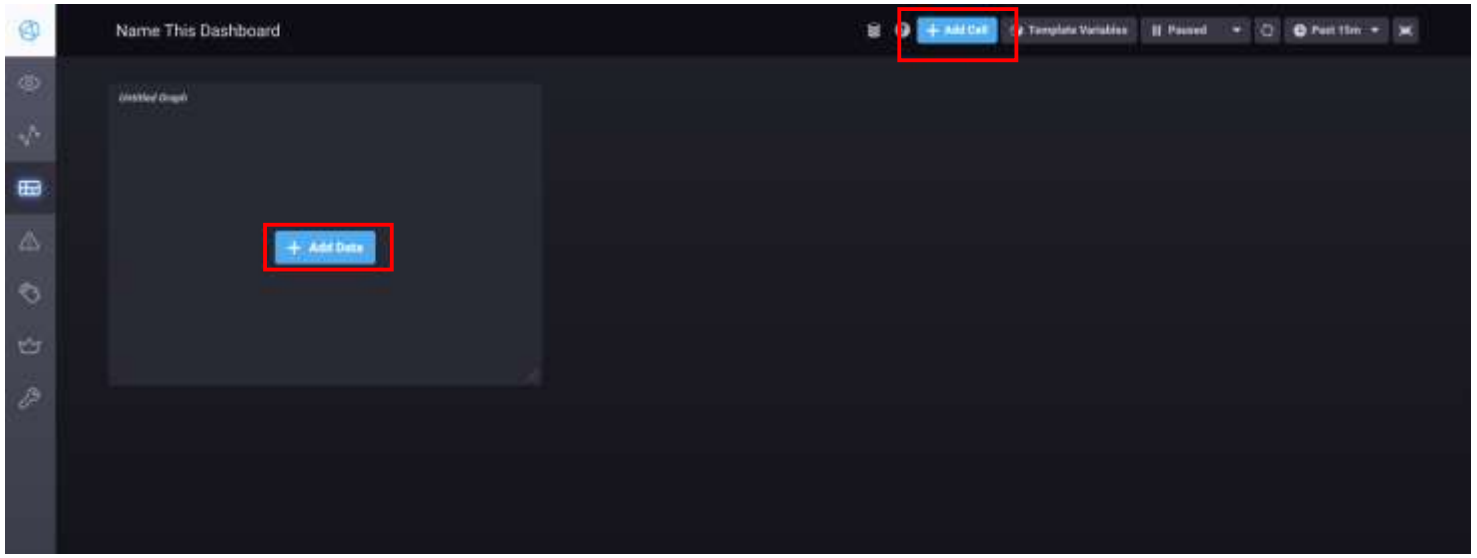
#4 Configure Chronograf Dashboard (2/4)

- Create Dash board



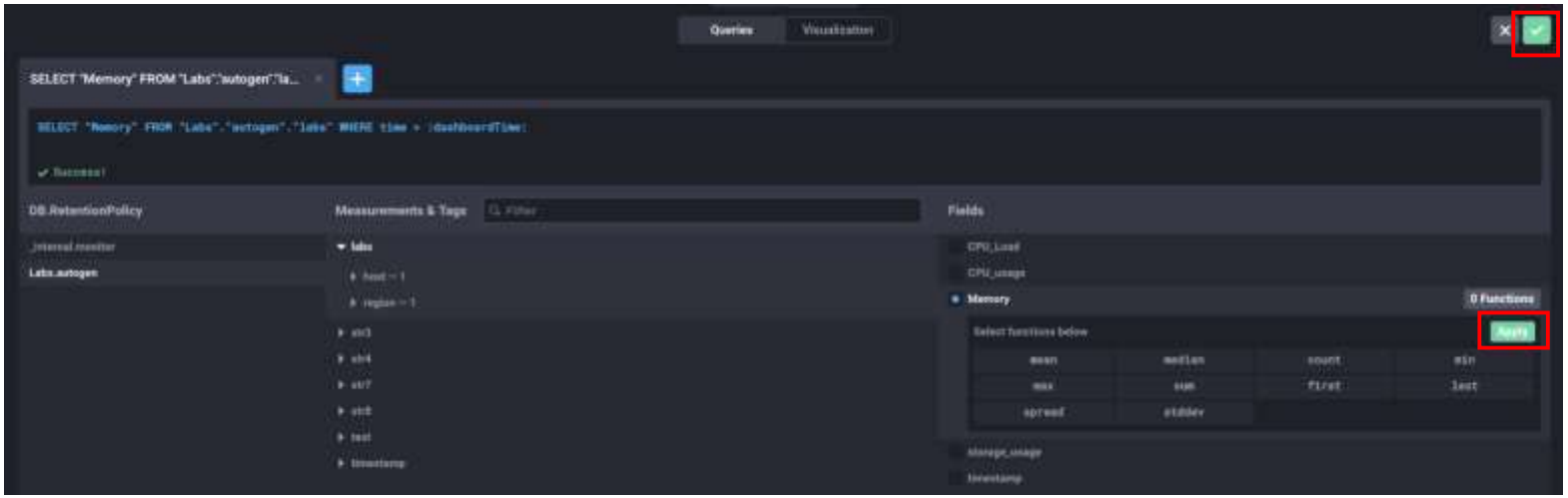
#4 Configure Chronograf Dashboard (3/4)

- Add Cell & Data



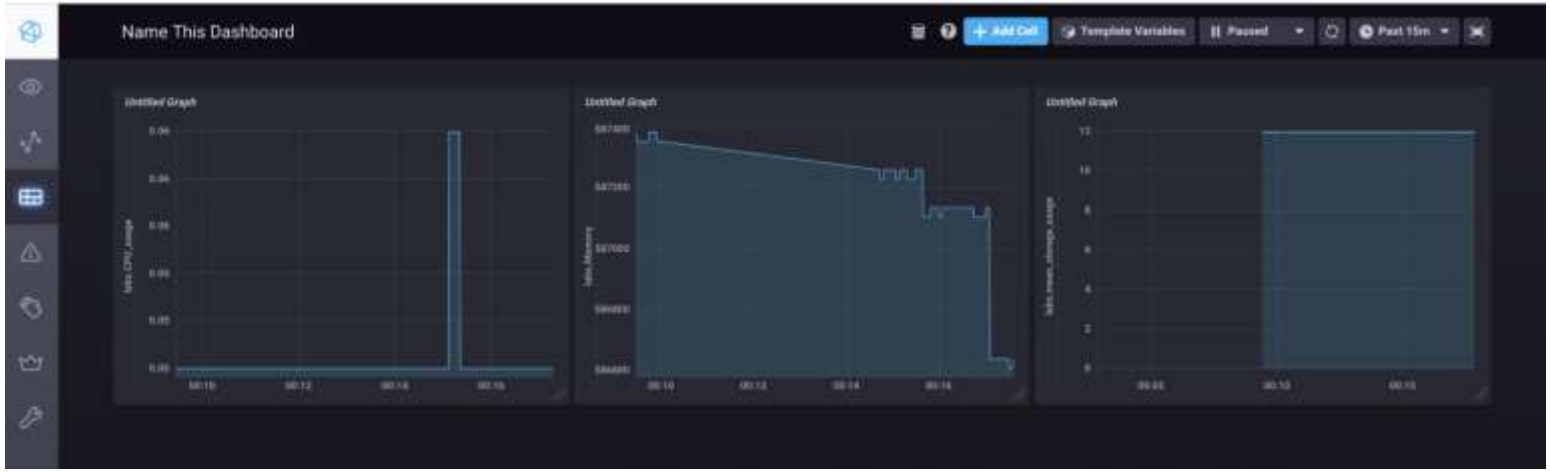
#4 Configure Chronograf Dashboard (4/4)

- Add Data

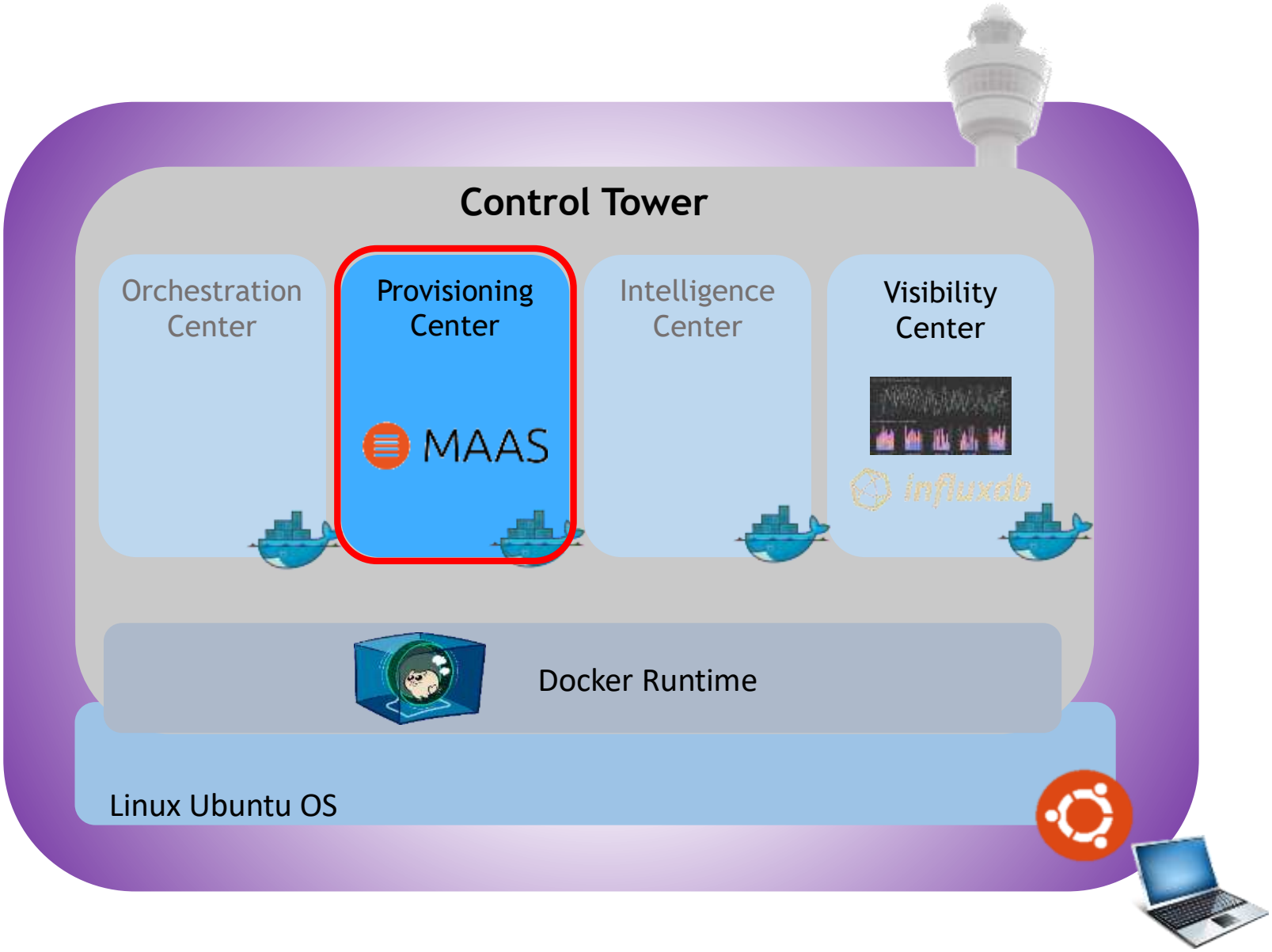


#5 – Check Chronograf Dashboard

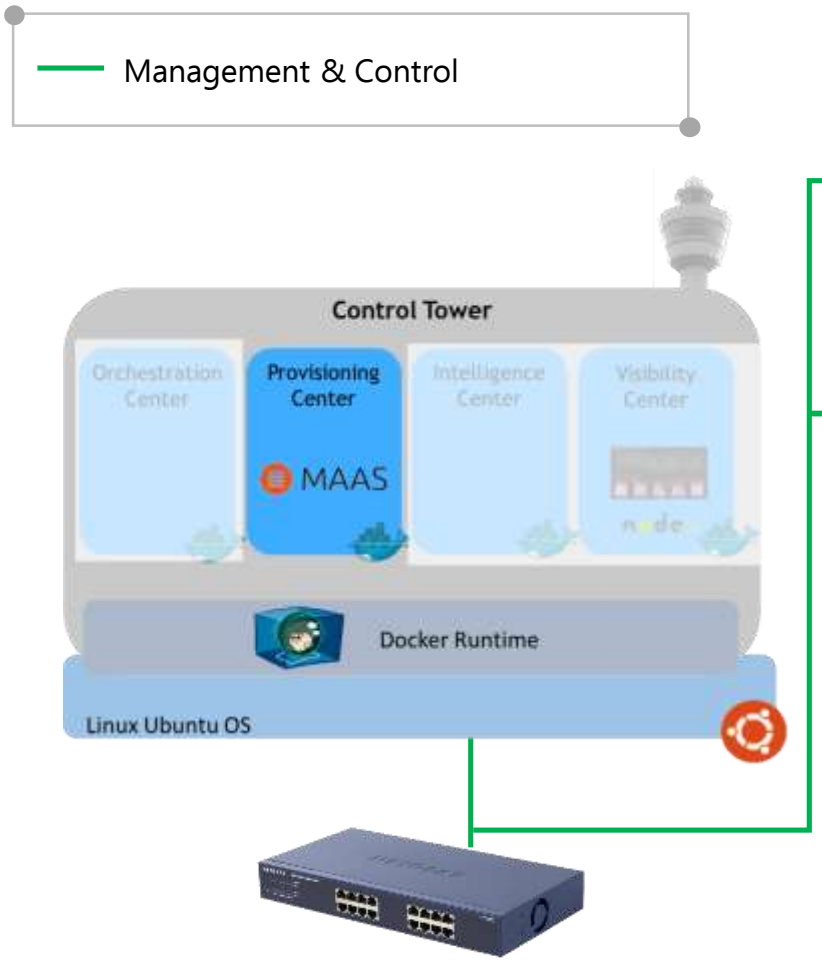
- We can see the changes of values from database



Automated Provisioning & Visibility



#6–1 OS manual Installation: Establish physical interconnect



Any Ethernet port that support PXE

Requirements for manual Installation

- DHCP PXE bootable
- USB to ethernet connector

Note: Typical NUC does not have a IPMI port.

#6-2 Install MAAS server

- 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

#6-3 Enlist, Commission and deploy the box

• 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”

#6-4 Deploying the box

- 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

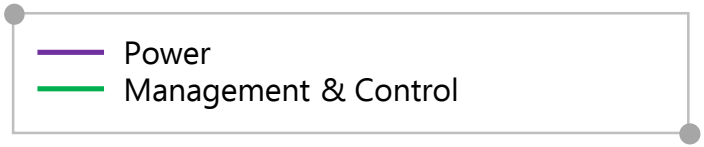
- Deploy

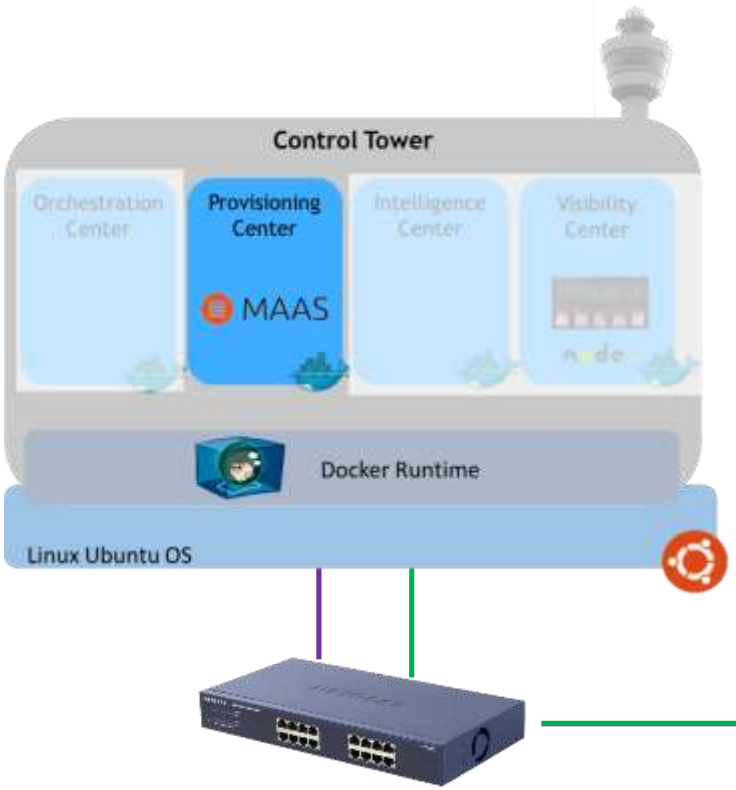
- From “take action”, choose “deploy”
- OS remote Install complete

Appendix

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

#7–1 OS Automated Installation: Establish physical interconnect



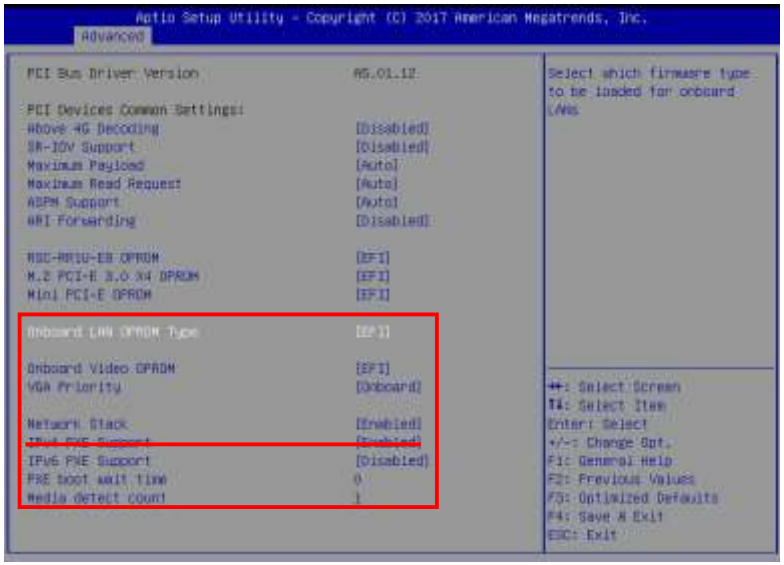


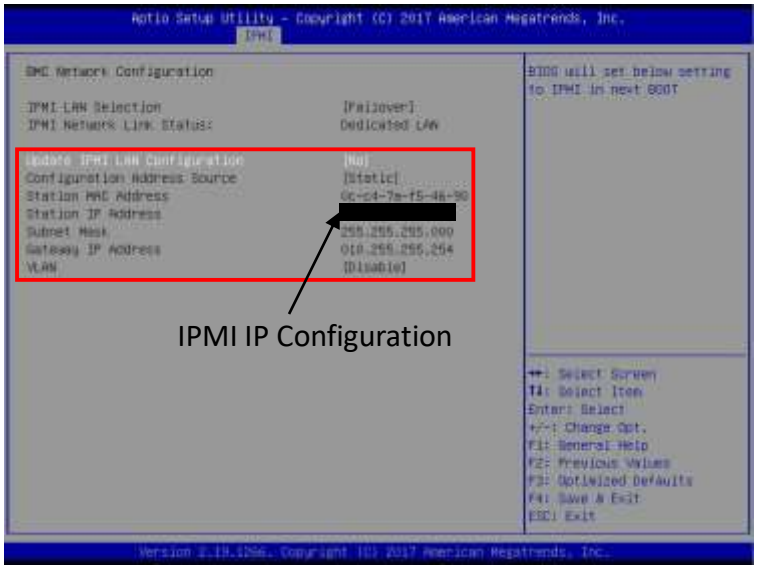
Requirements for Automated Installation

- IPMI, Intel AMT, IBM HMC … and so on.
- DHCP PXE bootable

Note: Typical NUC does not satisfy the above requirement!

#7–2 OS Automatic Installation: BIOS Configuration



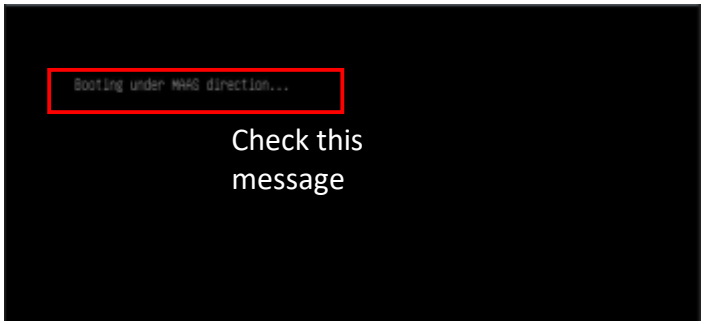


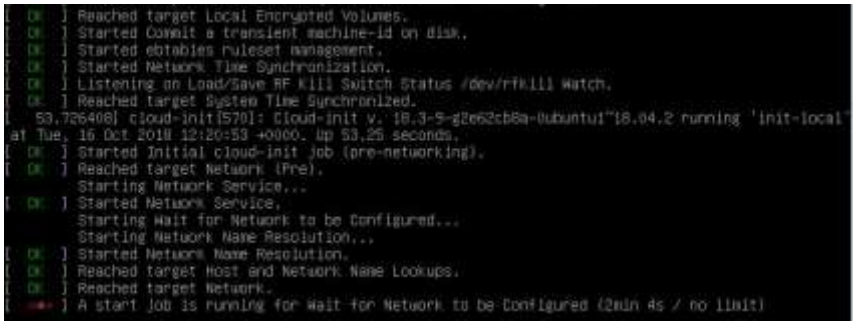
<BIOS PXE Configuration>

<BIOS IPMI Configuration>

- After then Save Configuration and Exit
- And then the PXE booting sequence is stated

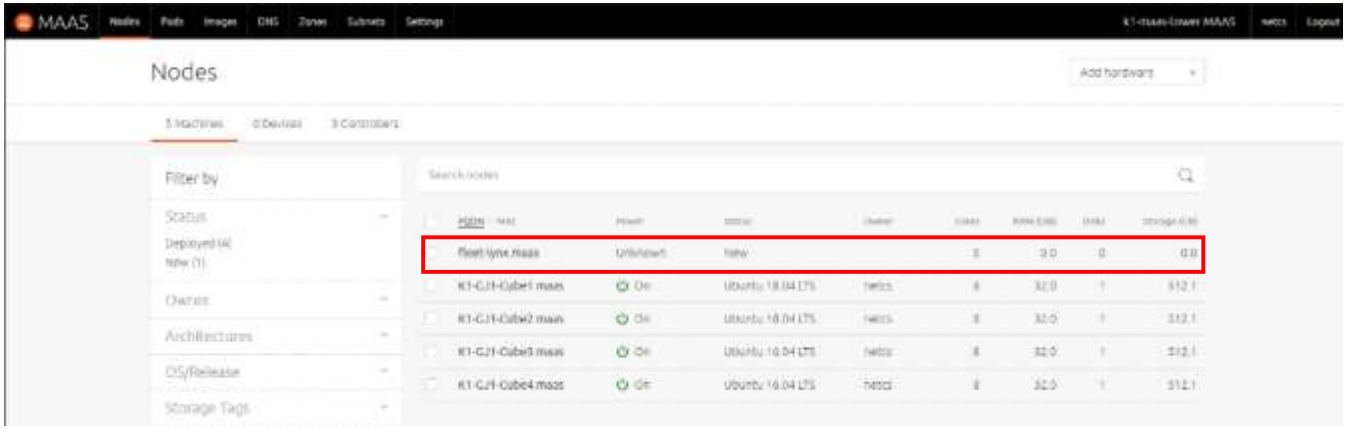
#8–1 OS Automatic Installation:Enlistment





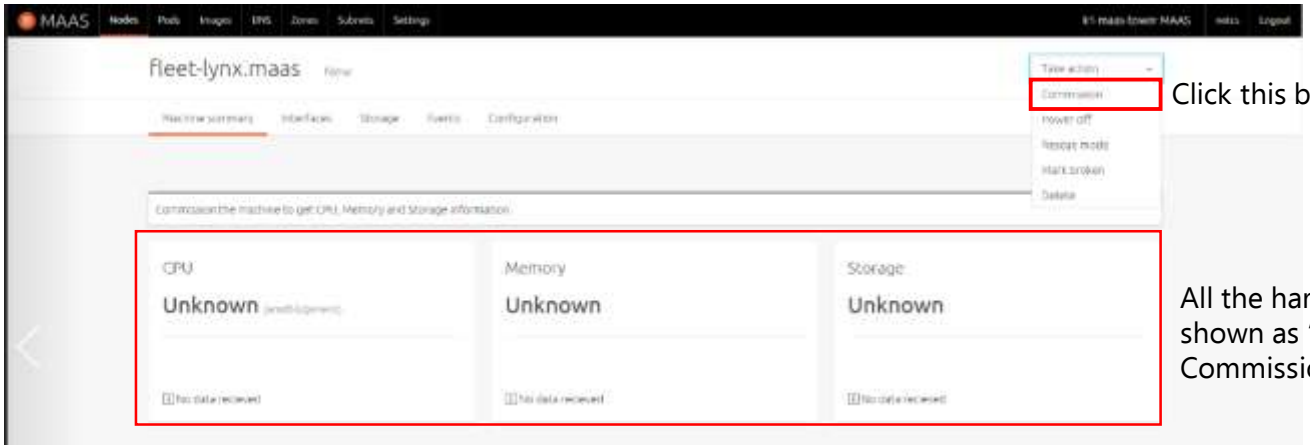
<Succeed PXE booting>

<Enlistment after PXE booting>



<After Enlistment (you can check on Web GUI) >

#8–2 OS Automatic Installation:Commissioning



Click this button to commission

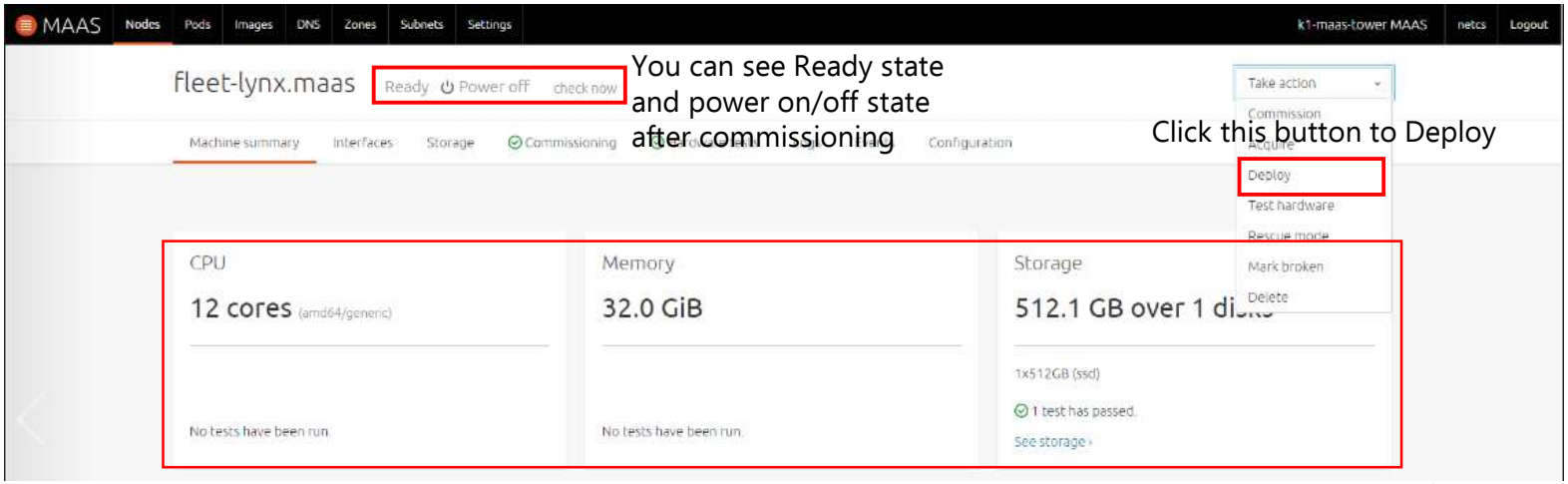
All the hardware information shown as 'Unknown' before Commissioning

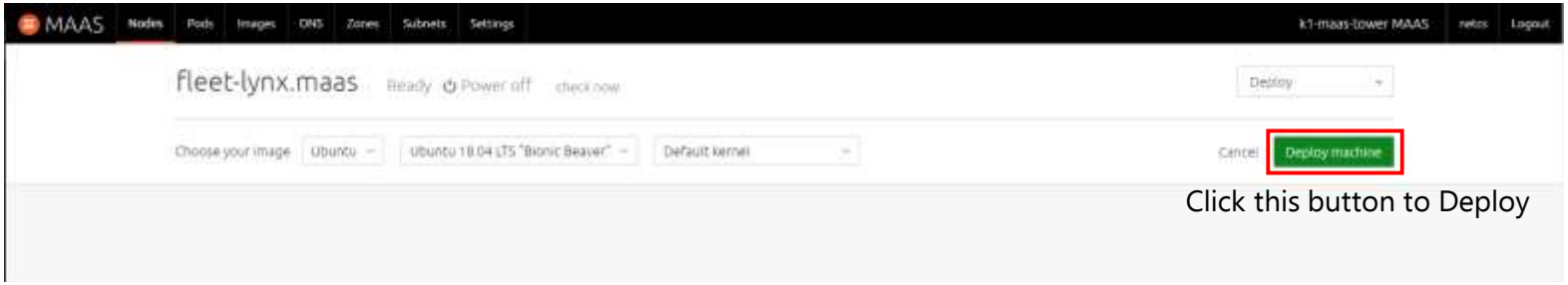
Do not check anything

Click this button to commission

- It takes about 10 minutes

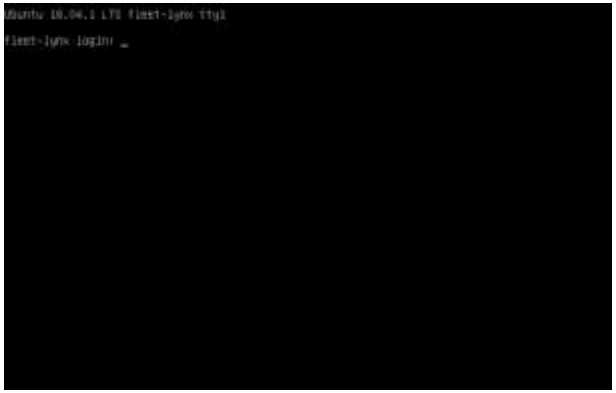
#8–3 OS Automatic Installation:Deployment (1/2)





- It is a OS Installation procedure
- It takes about 15 minutes

#8–3 OS Automatic Installation:Deployment (2/2)



<Complete Deploying (OS booted)>

Machine state is changed to 'Deployed'

<Complete Deploying (On Web GUI)>

Review

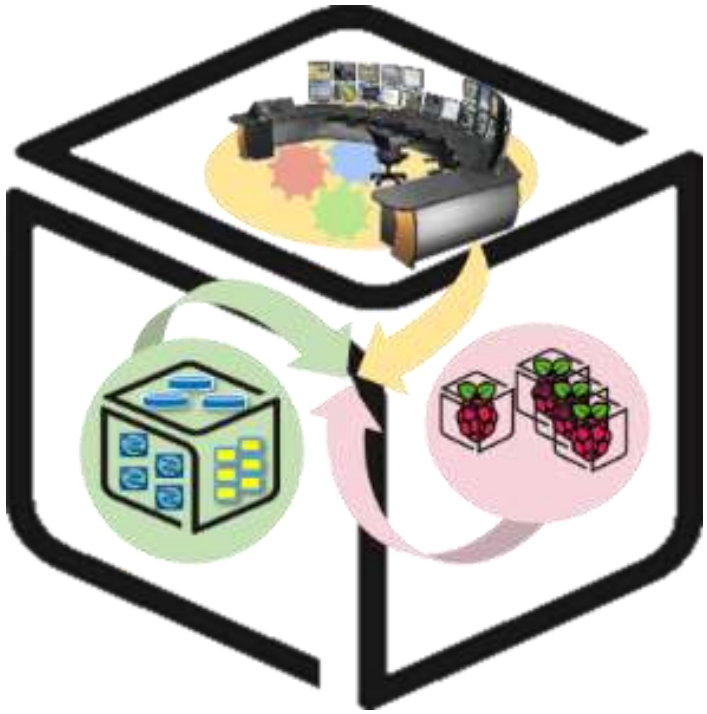
Lab Summary

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