# SmartX OpenStack Cloud: Provisioning/Operation Automation and Orchestration

## Open Networking Korea 2015 & 정보과학회 단기강좌

신준식, 한정수, 배정주
(On behalf of SmartX Team)

Networked Computing Systems Laboratory
School of Information and Communications
Gwangju Institute of Science & Technology (GIST)

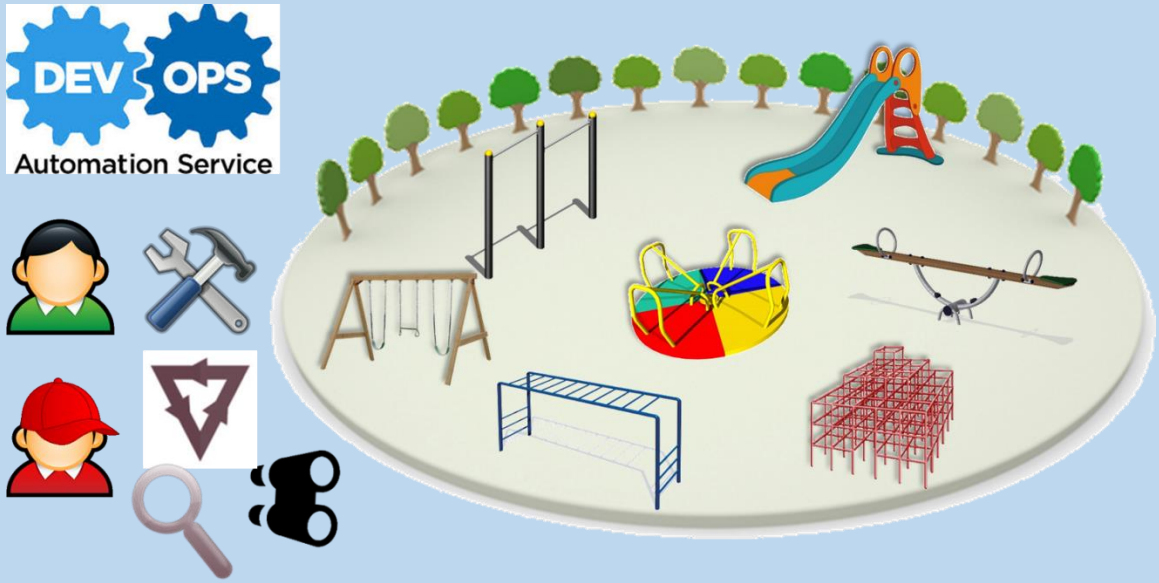
openstack
CLOUD SOFTWARE

# Open Networking KOREA

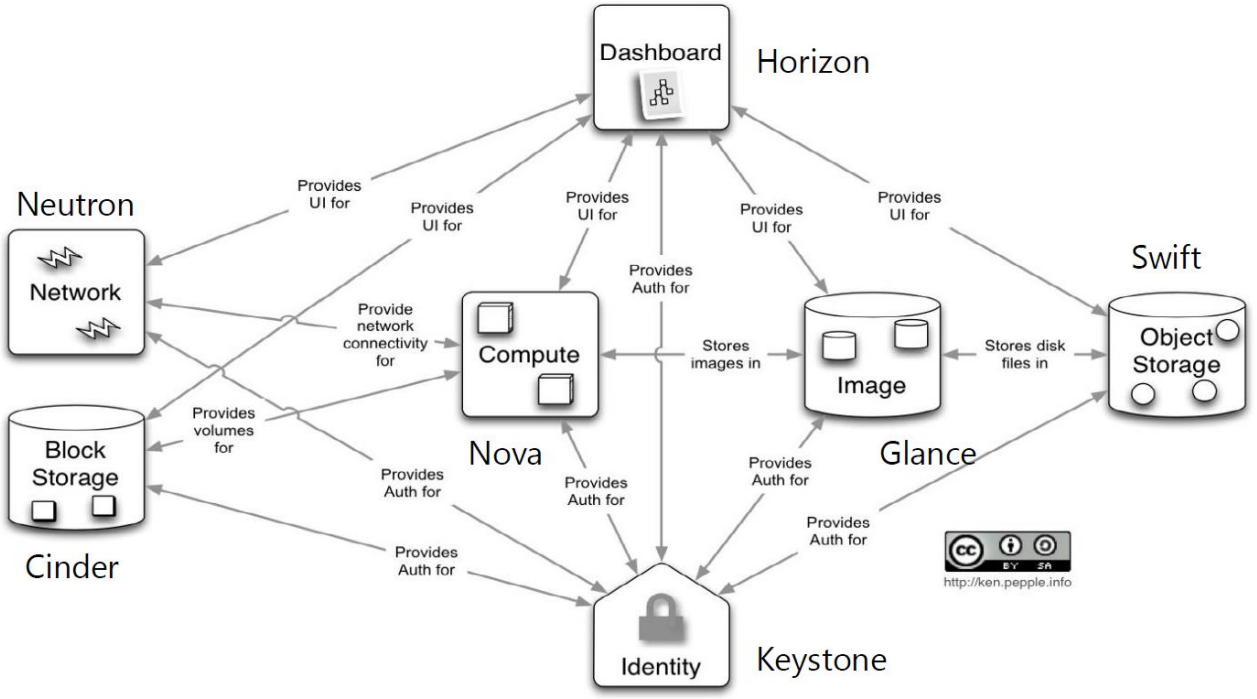


http://opennetworking.kr

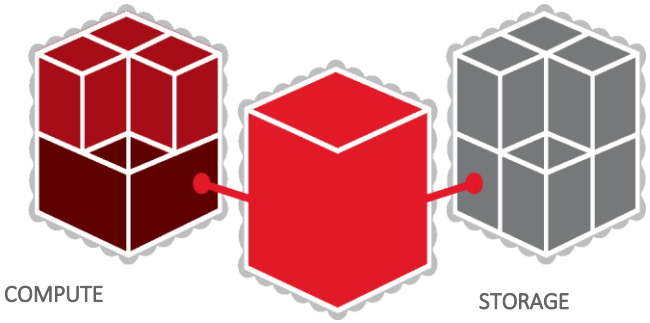
# OpenStack-leveraged SmartX Playground



# Open-Source Cloud OS: OpenStack (Infra+)




openstack™




CISCO openstack

vmware openstack

ORACLE

*Vendors will co-opt and fragment OpenStack*

HP Helion openstack

IBM openstack

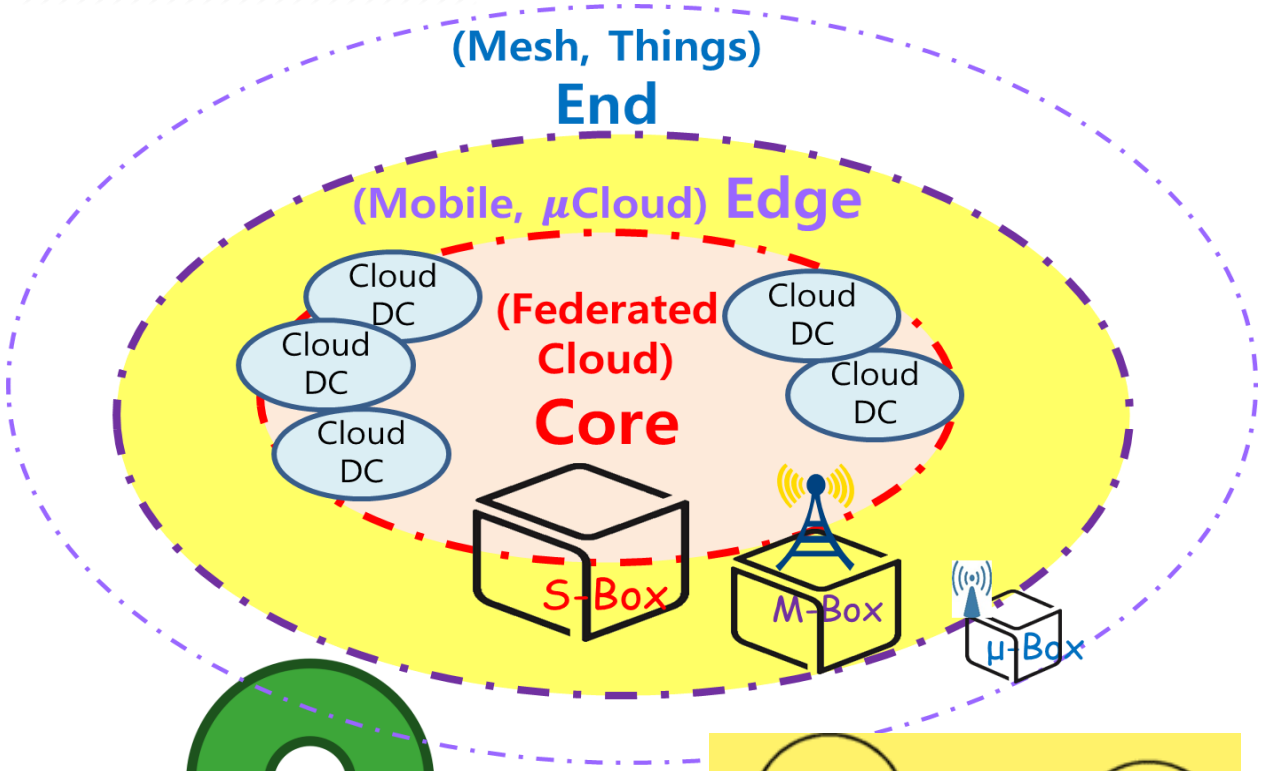
DELL & openstack

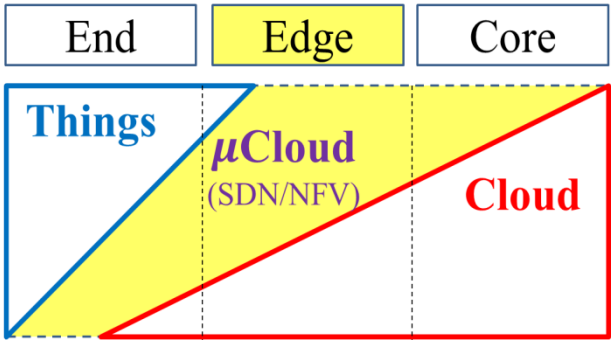
redhat. openstack

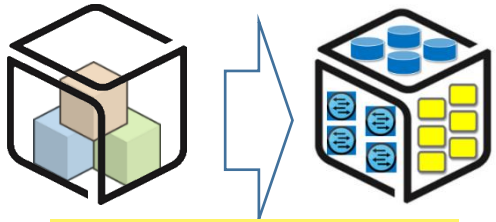
Azure openstack

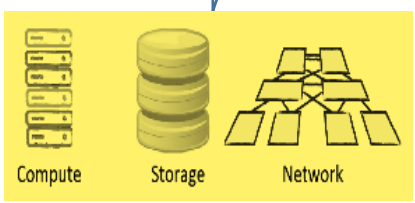
ALYPTUS openstack

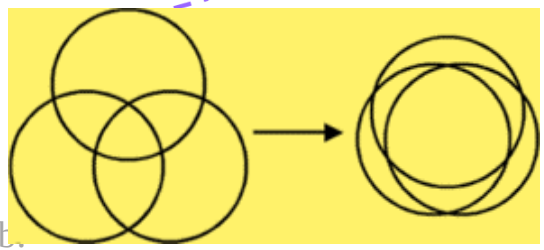
# Converged Software-Defined Infrastructure (SDN/NFV/Cloud Integrated)

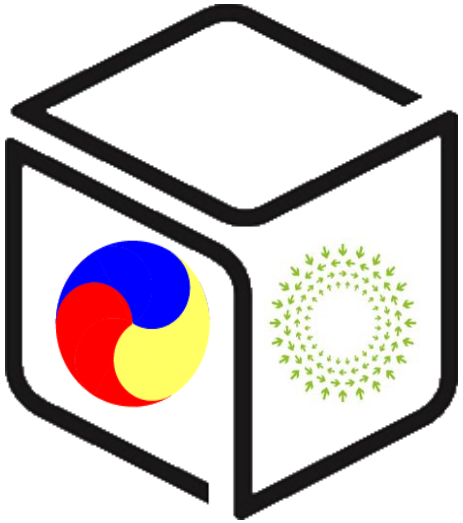





# Convergent SDI & Open-Source SW/HW




OPNFV

openstack™

OPEN DAYLIGHT

onos

IoT

IoT

IoT

µCloud (SDN/NFV/ FastData)

Federated Cloud DCs (BigData/HPC)

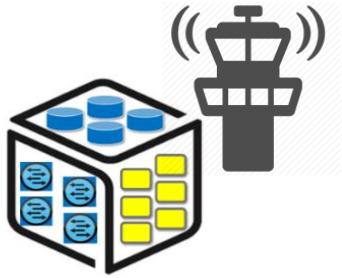
# Smart + X

Providing Intelligence

All services that are Flexible and Adaptable

⇒ Providing **User-defined**, **Intelligent, and Flexible/Adaptable SmartX Services**

# SmartX Box: Inter-Connected Functions inside Boxes/Sites



Open Container Initiative

OPEN VIRTUALIZATION ALLIANCE

THE LINUX FOUNDATION



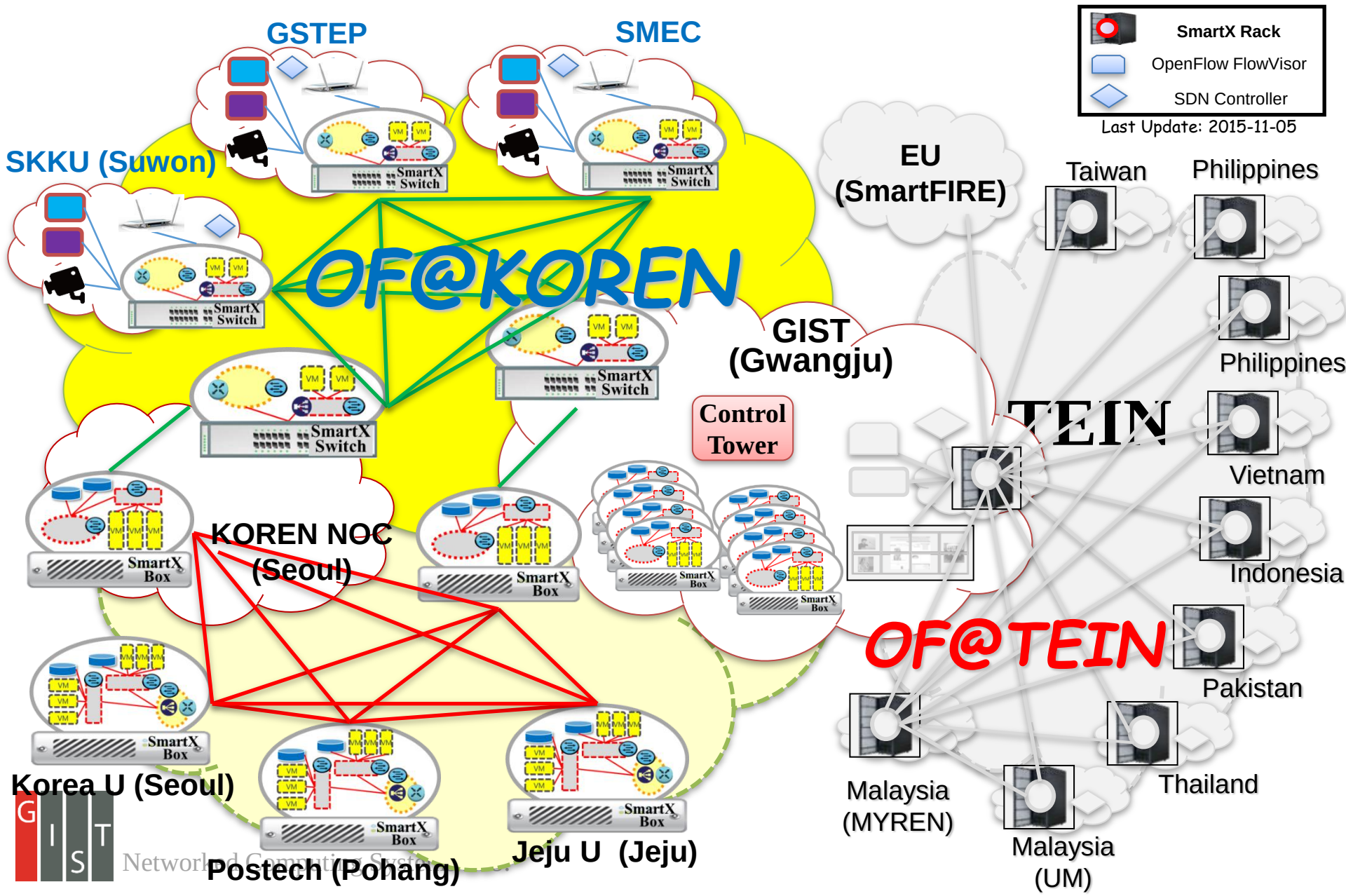

IOVISOR PROJECT

OpenDataPlane.org

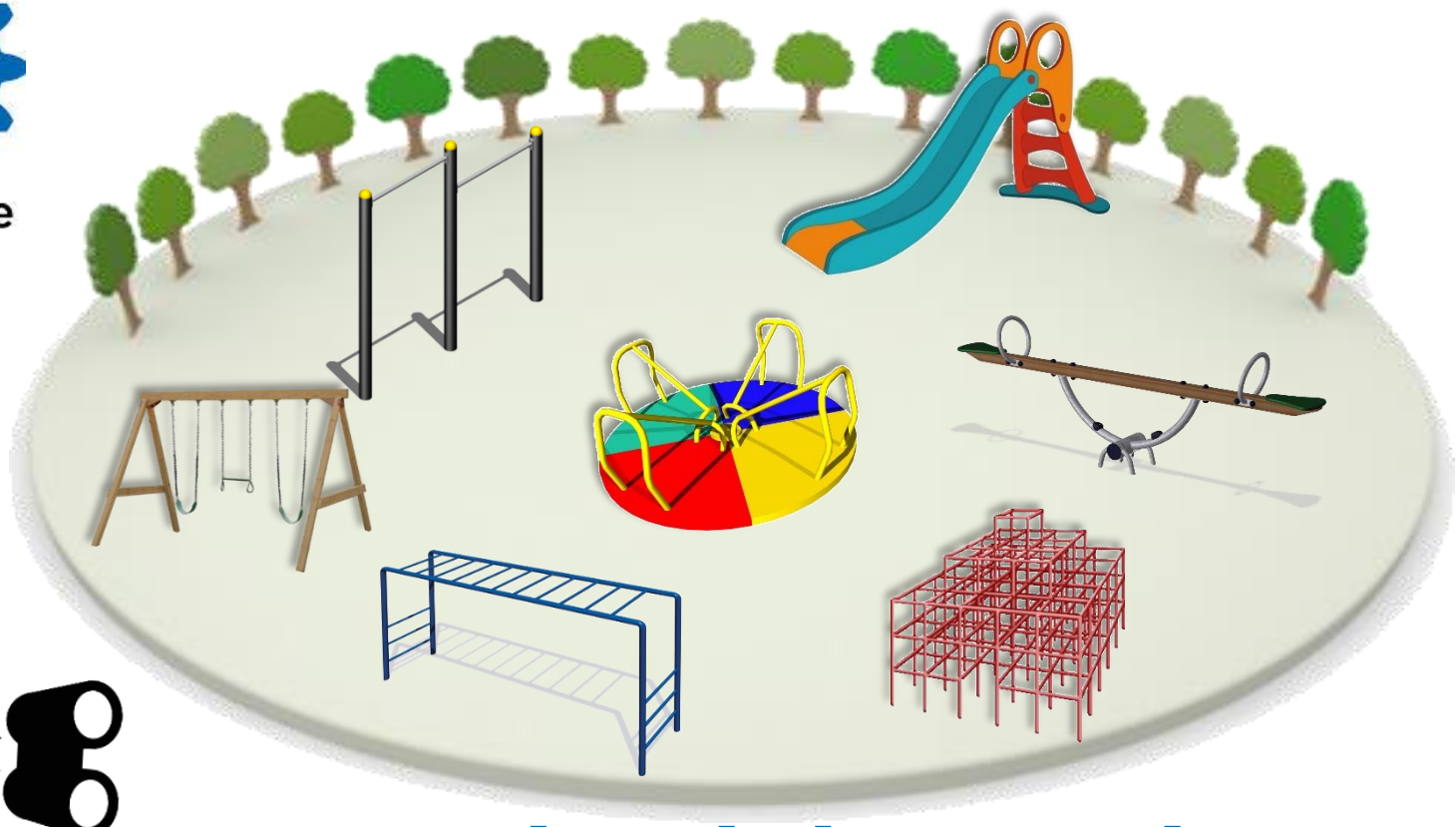
DPDK
DATA PLANE DEVELOPMENT KIT

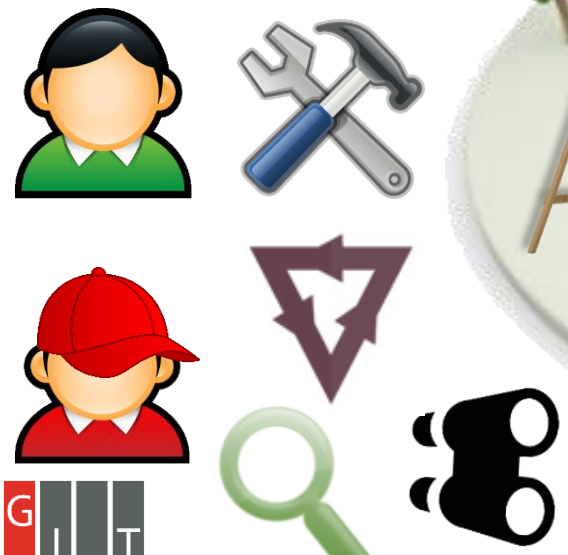
OPEN Compute Project

# SmartX Playgrounds: OF@KOREN & OF@TEIN



# Building/Operating & Playing with Open Federated (Shared) Playground



**Open & Shared Playground!!!**

# Open Federated Playground for SDI R&D (Planned)

K-ONE

K-ONE 공용개발환경

**K-ONE Consortium & Collaborators**

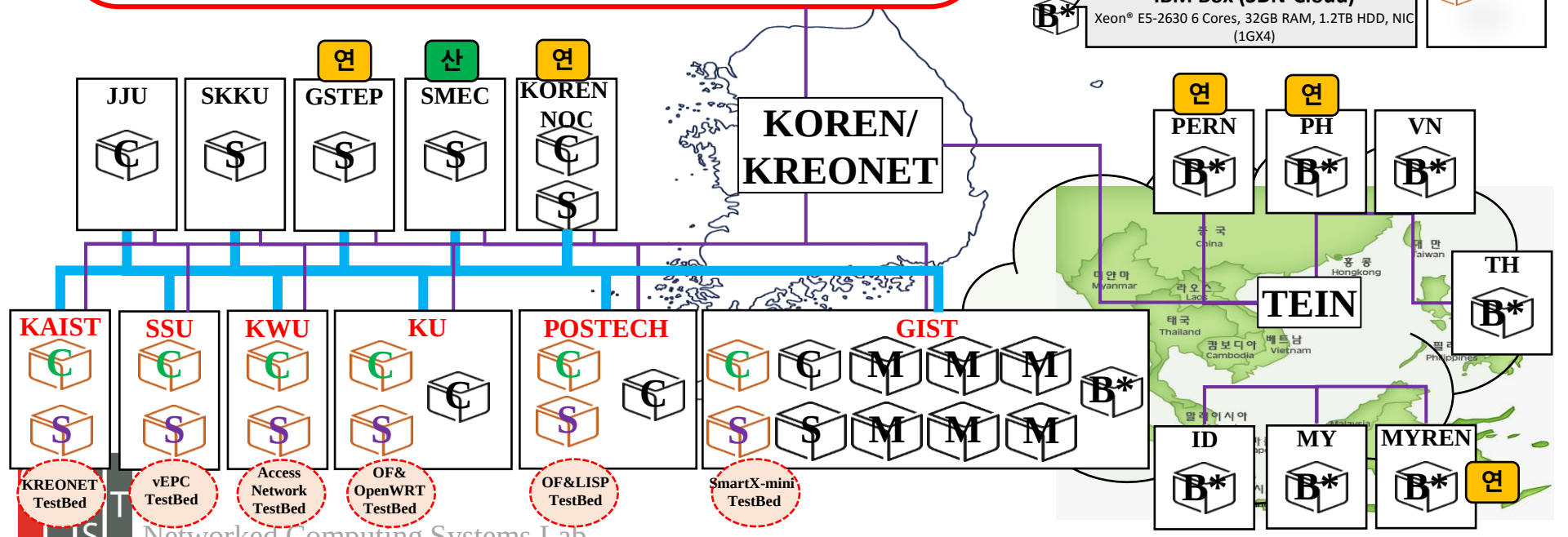

**SmartX Box HW**

- **ONP Box (High-Power Cloud)** — Xeon® E5-2690 v2 20 cores, 96GB RAM, 1.3TB SSD, 3TB HDD, NIC(10GX2, 1G X4)
- **ONP Box (Cloud Storage)** — Xeon® E5-2650 v3 20 cores, 128GB RAM, 120GB SSD, 8 TB HDD, NIC (10GX2, 1GX4)
- **Server-Switch Box (SDN/NFV)** — ATOM C2558 4 Cores(Sw)/Xeon® E5-2600 v2 16 Cores(Server), 40GB RAM, Ports(10G X 24, 40G X 4)
- **IBM Box (SDN-Cloud)** — Xeon® E5-2630 6 Cores, 32GB RAM, 1.2TB HDD, NIC (1GX4)

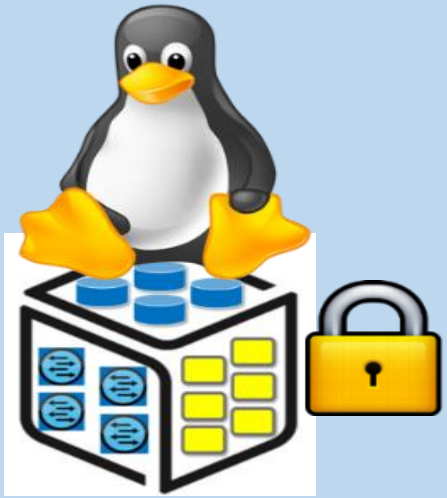
**Legend**

Network: 10G, 1G

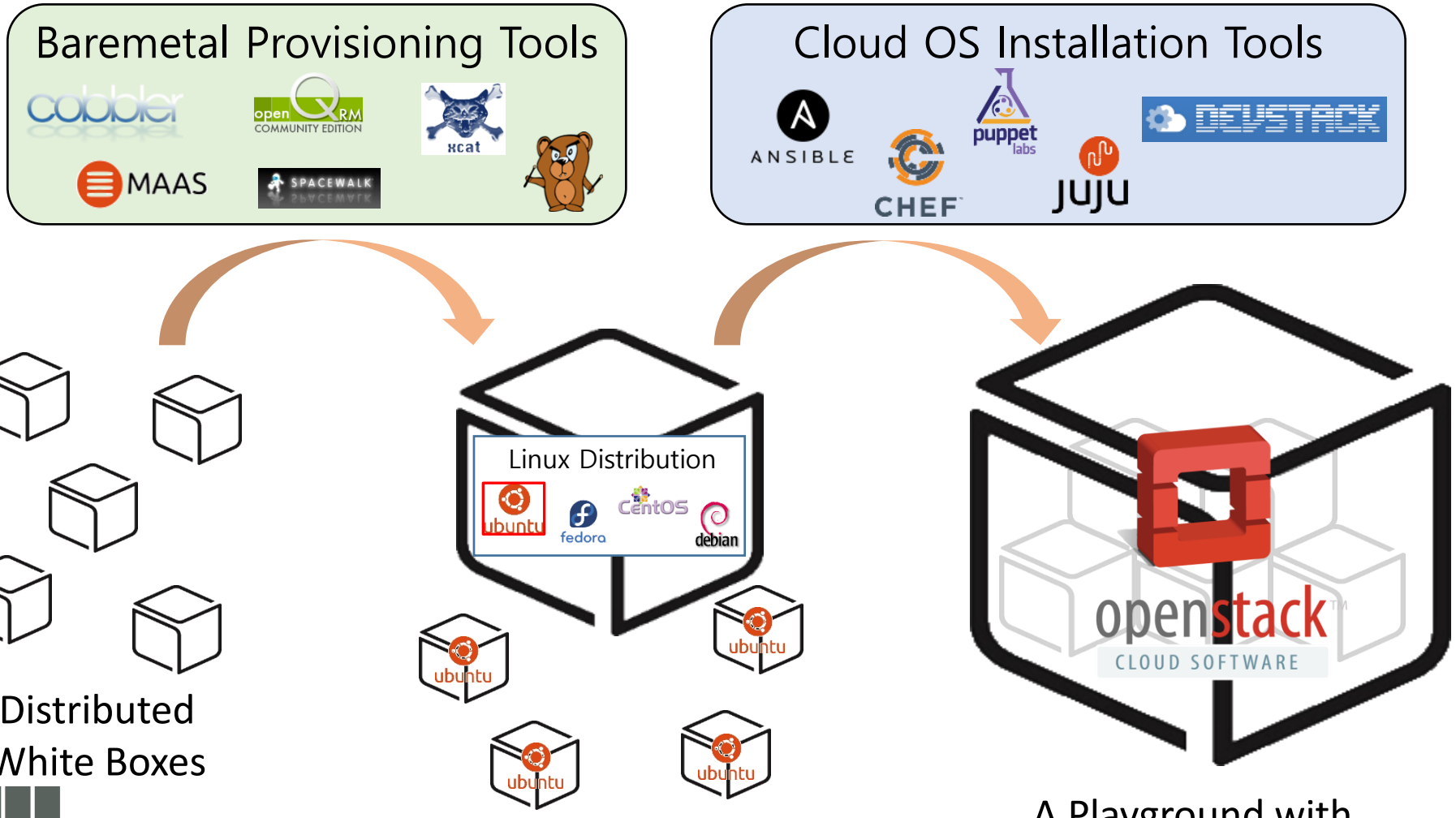
Box: Existing, New



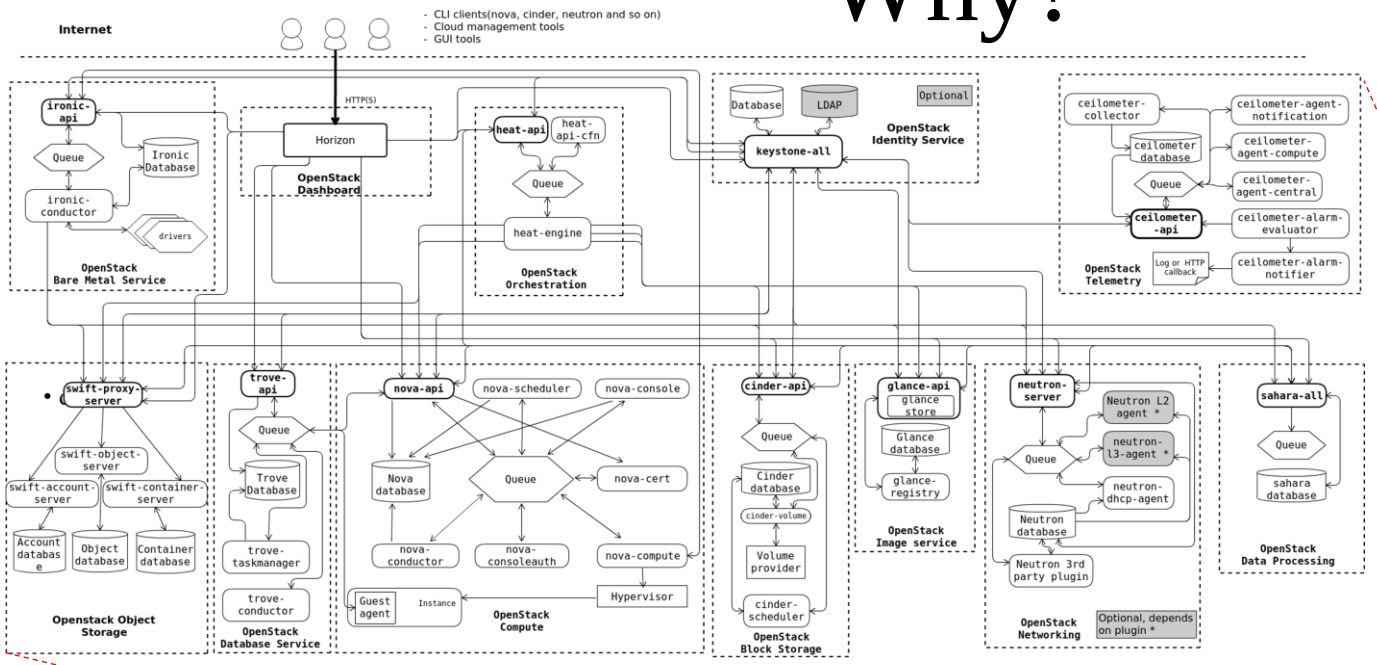
# OpenStack: Automated Provisioning (Installation & Configuration)



# Automated Installation/Configuration of OpenStack-leveraged Playground?

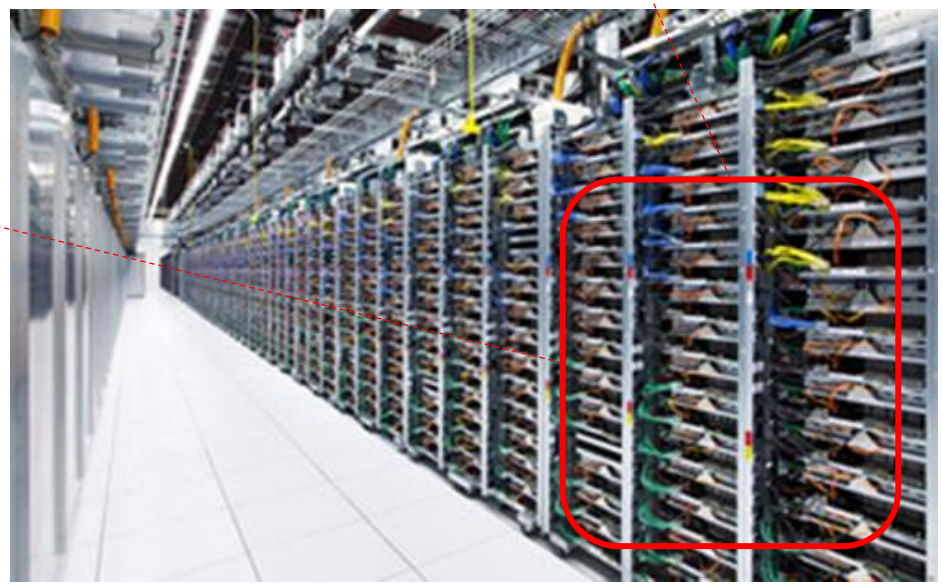


# Automated Installation/Configuration Tool: Why?




**OpenStack Architecture**

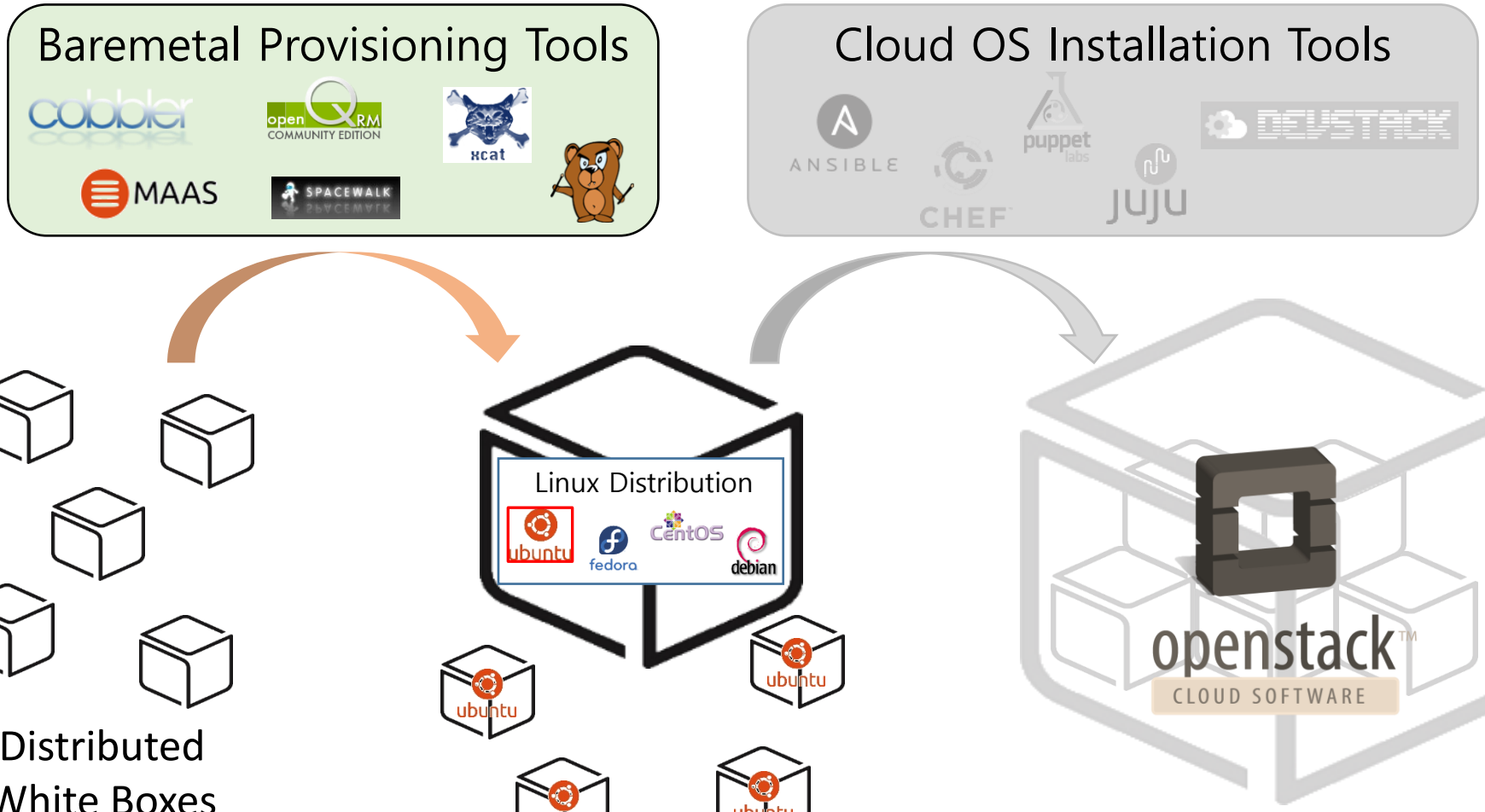
**Data Center with lots of boxes**



**Manual Installation/Configuration?**

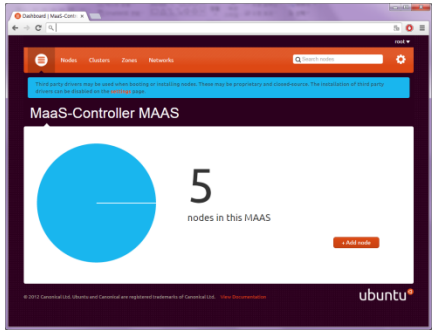
Is it really possible?

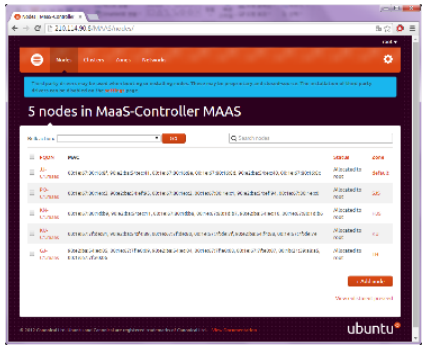
# Automated Installation/Configuration: Step #1 - Linux Installation

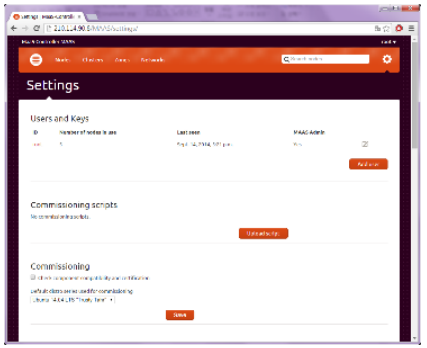


# Ubuntu MaaS (Metal As A Service)

- Baremetal Provisioning Tool
- Web UI

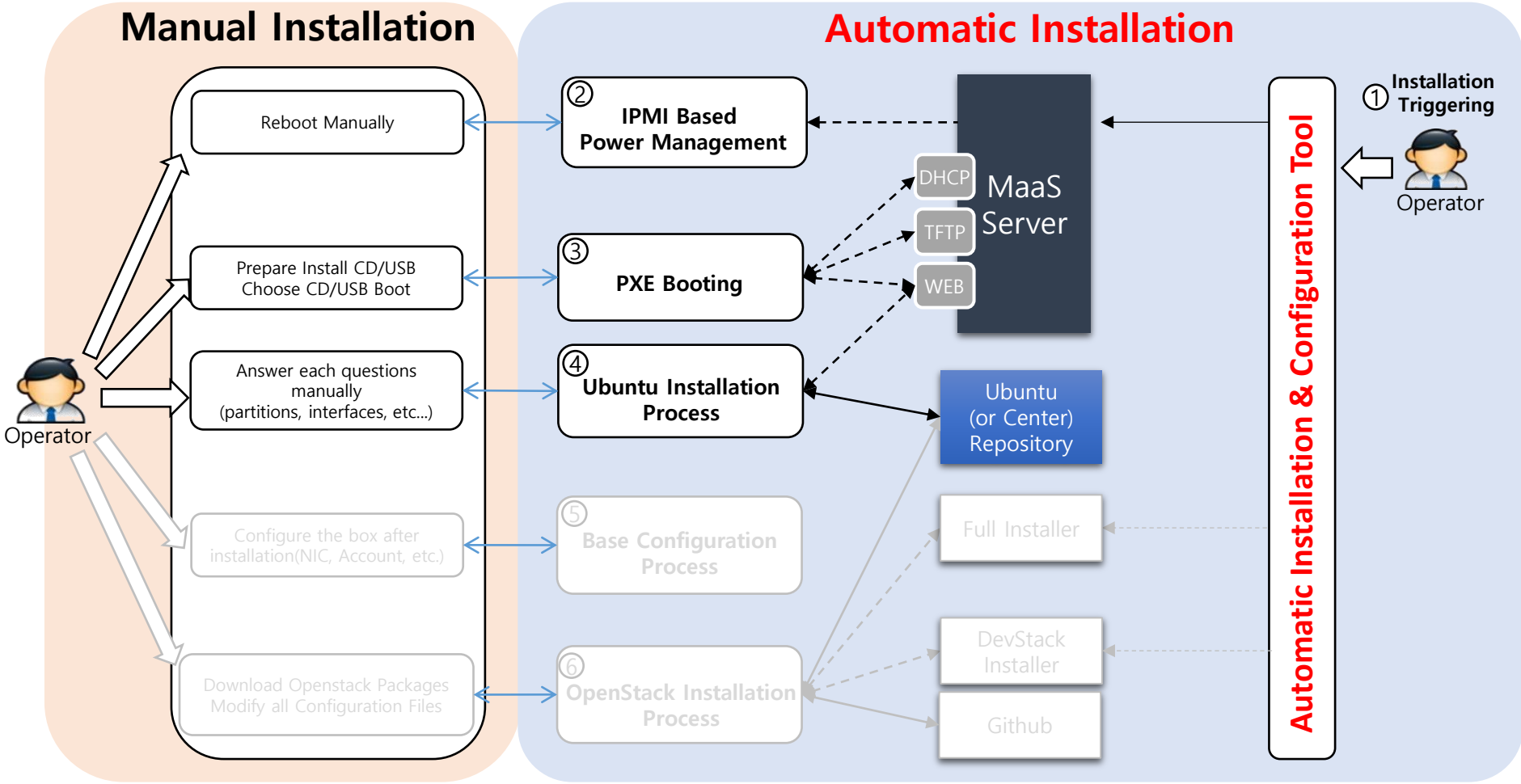


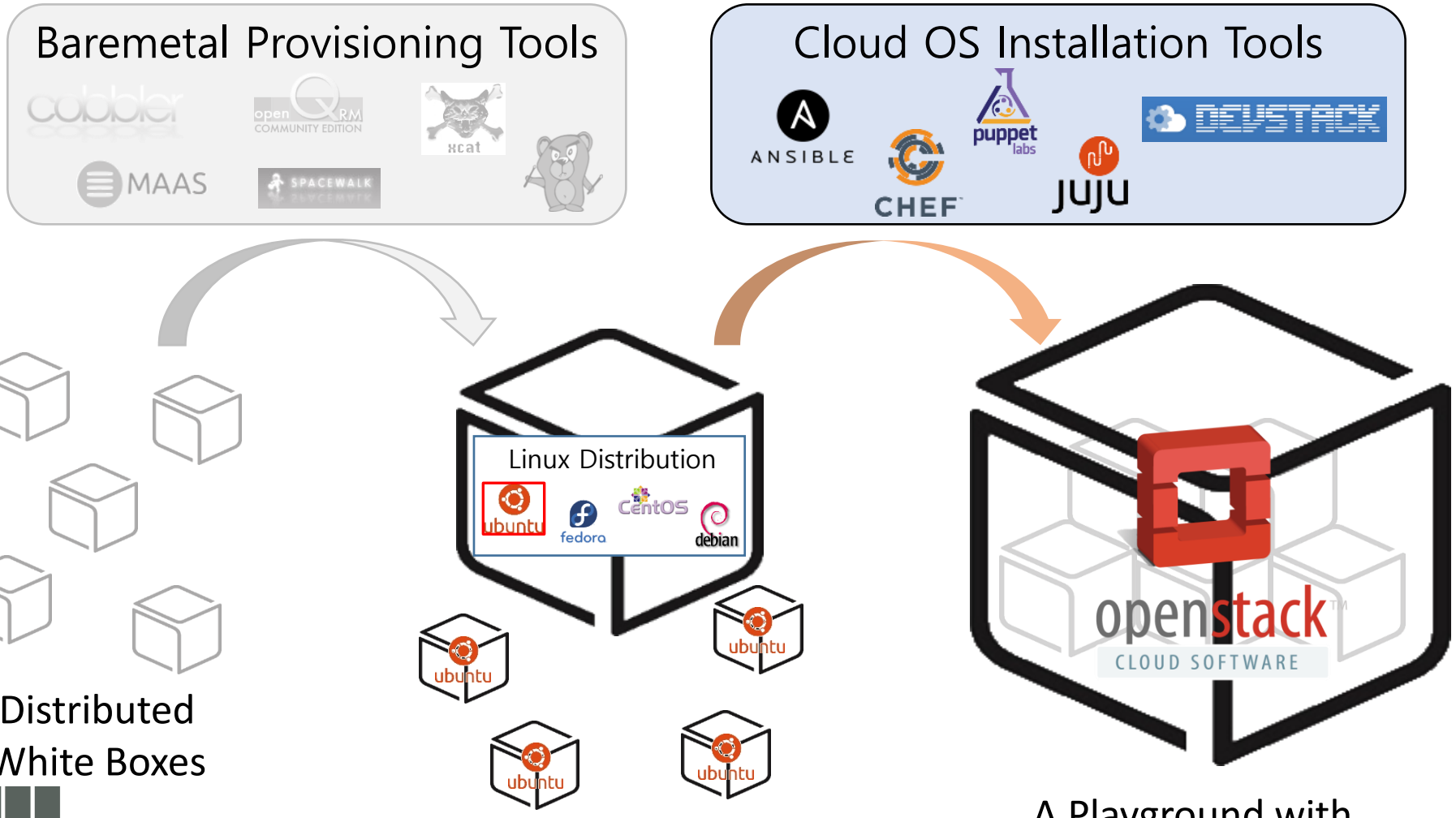




- Support ~~Only Ubuntu Distribution~~ Ubuntu, CentOS, Windows
- Automatically Manage installed Ubuntu Version
- Easy to Configure and to Use
- Provide fast installation methods
  - Curt installer (fast-path installer)
  - Package caching
- http://maas.ubuntu.com

# Automated Ubuntu I&C Procedure



# Automated Installation/Configuration: Step #2 - OpenStack Installation



# OpenStack Cloud: Devstack Installation Case

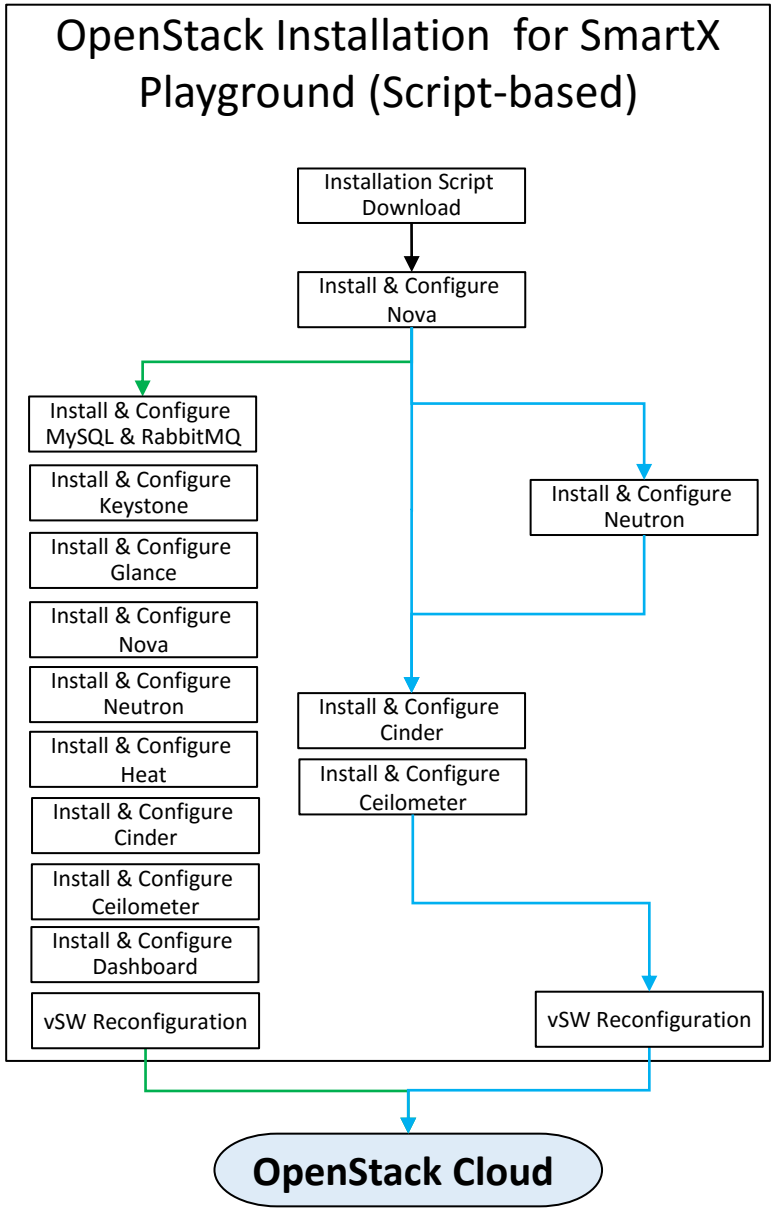

- Script-based OpenStack installation tool for developers (DevStack is only targeted for Developers, but not good for general operation
- Provide the easiest way to install OpenStack: For basic configuration, **only 30 minutes** to install
- Install based on a configuration file
  - The only one thing you should know is how to define devstack configuration file. (local.conf); Easy to configure various environments of OpenStack Cloud and to introduce new features into environment
- Other projects can also be easily installed by using Devstack
  - OpenStack – Opendaylight controller
  - OpenStack – Docker
  - OpenStack – DPDK accelerated OVS
  - OpenStack – Ceph
- It is suitable for creating the development environment, not for OpenStack Playground operation

```
[[local|localrc]]
FIXED_RANGE=10.0.0.0/16
FIXED_NETWORK_SIZE=65534
FLAT_INTERFACE=eth0

MULTI_HOST=True

HOST_IP=<TARGET_IP>
SERVICE_HOST=$HOST_IP

MYSQL_HOST=$SERVICE_HOST
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=$SERVICE_HOST

ADMIN_PASSWORD=secrete
MYSQL_PASSWORD=secrete
RABBIT_PASSWORD=secrete
SERVICE_PASSWORD=secrete
SERVICE_TOKEN=secrete

LOGFILE=../logs/stack.sh.log
#SCREEN_LOGDIR=../screen_log

disable_service n-net
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-l3
enable_service q-meta
enable_service neutron
enable_service tempest

Q_PLUGIN=ml2
ENABLE_TENANT_TUNNELS=True

NOVA_BRANCH=stable/icehouse
GLANCE_BRANCH=stable/icehouse
HORIZON_BRANCH=stable/icehouse
NEUTRON_BRANCH=stable/icehouse
CINDER_BRANCH=stable/icehouse
KEYSTONE_BRANCH=stable/icehouse
```

**http://devstack.org**

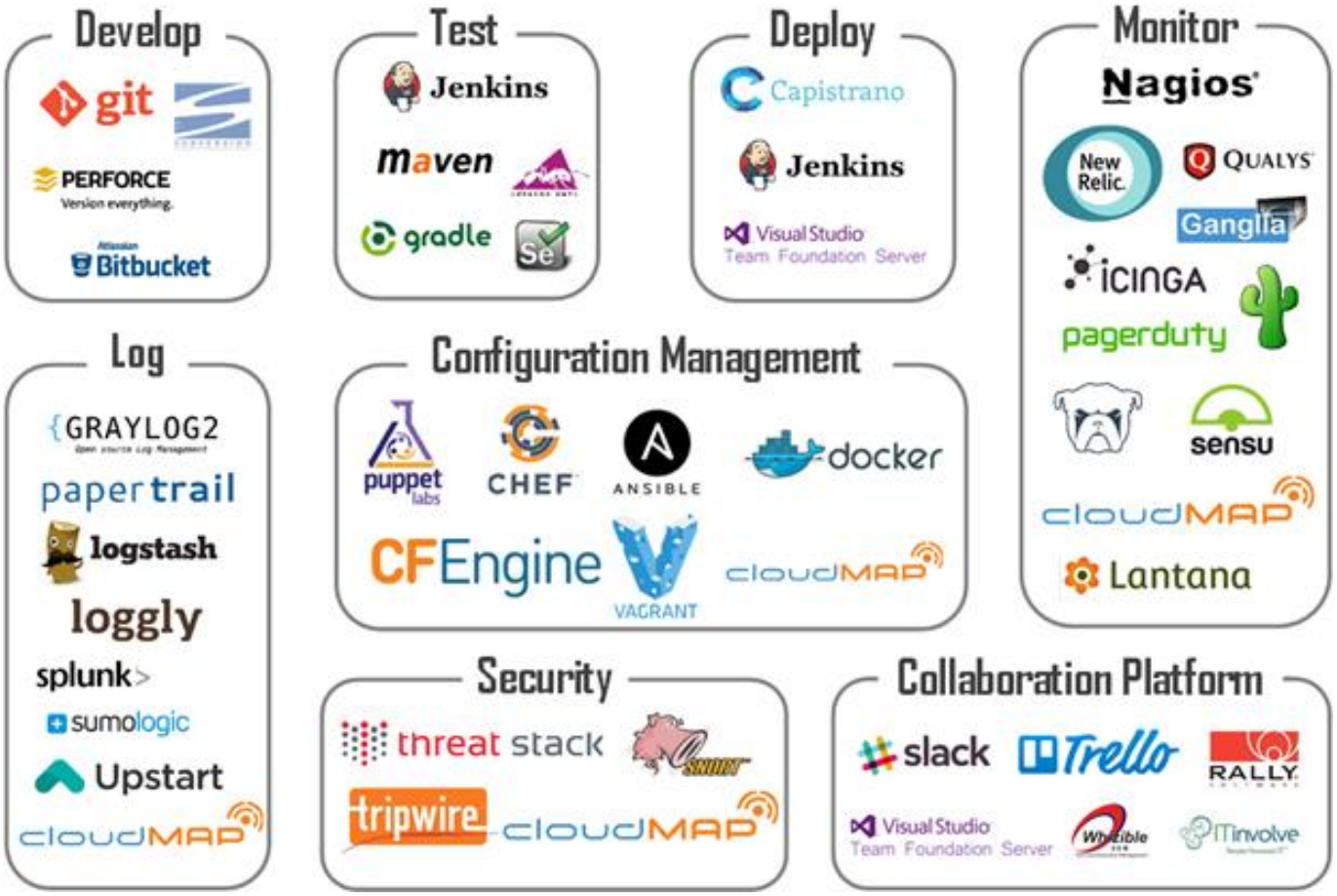
# OpenStack Cloud: Full Installation Case

- Based on the playground operation experience, we decided to install OpenStack by following Official OpenStack Installation Manual (It is commonly said that it may take 1 month to finish it manually)
- Suitable installation option for OF@KOREN **SmartX Multi-site Cloud Playground**
  - Nova, Keystone, Neutron, Heat, Cinder, Ceilometer, Horizon
- Currently we do not leverage any DevOps tools yet; Developed in-house a customized script-based full-version installer (Grizzly to Kilo iteration…).



# OpenStack Installation: DevOps Tool-based Installation

Develop → Build → Deploy → Test → Release




http://www.e-zest.com/the-devops-philosophy/

# OpenStack Installation: DevOps Tools - SaltStack, Ansible, Puppet, Chef (1/2) -

| | Salt | Ansible | Puppet | Chef |
|---|---|---|---|---|
| **Motivation** | Creators found existing solutions to be lacking, and wanted a very low latency, highly scalable remote execution and data collection framework | Disappointment that existing tools required an agent and made it difficult to accomplish tasks like rolling deployments | Created "… out of fear and desperation, with the goal of producing better operations tools and changing how we manage systems" | Chef began as an internal tool for Opscode, to build end-to-end server/ deployment tools. Soon, its creators realized its broader use |
| **Users** | PayPal, Verizon, HP, Rackspace | Blue Box, Red Hat | Paypal NYSE, ADP, Symantec, Sony | Bloomberg, Ancestry.com, GE Capital, Digital Science, Nordstrom |
| **Enterprise offering** | Yes | Hosting/Consulting/ Training | Yes | Yes |
| **License** | Apache License v2 | GNU Public License v3 | Apache License v2 | Apache License v2 |
| **GitHub activity** | | | | |
| Contributors | 1,041 | 1,003 | 355 | 369 |
| Commits | 49,193 | 13,527 | 19,595 | 12,089 |
| Branches | 11 | 33 | 9 | 177 |
| Releases | 82 | 57 | 291 | 231 |

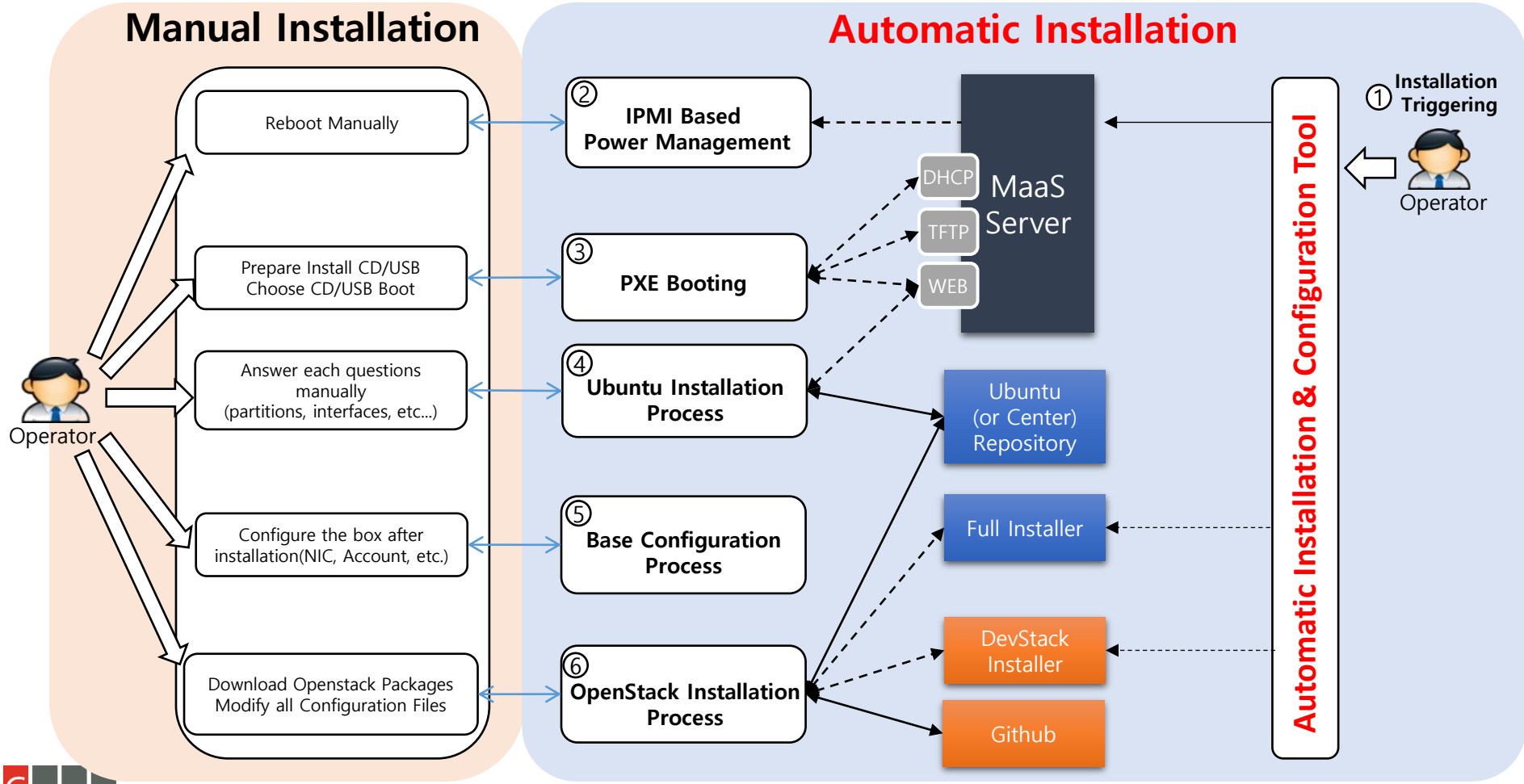
Ref: "CAPS: What's best for deploying and managing OpenStack?", OpenStack Summit 2015 Tokyo

# OpenStack Installation: DevOps Tools - SaltStack, Ansible, Puppet, Chef (2/2) -

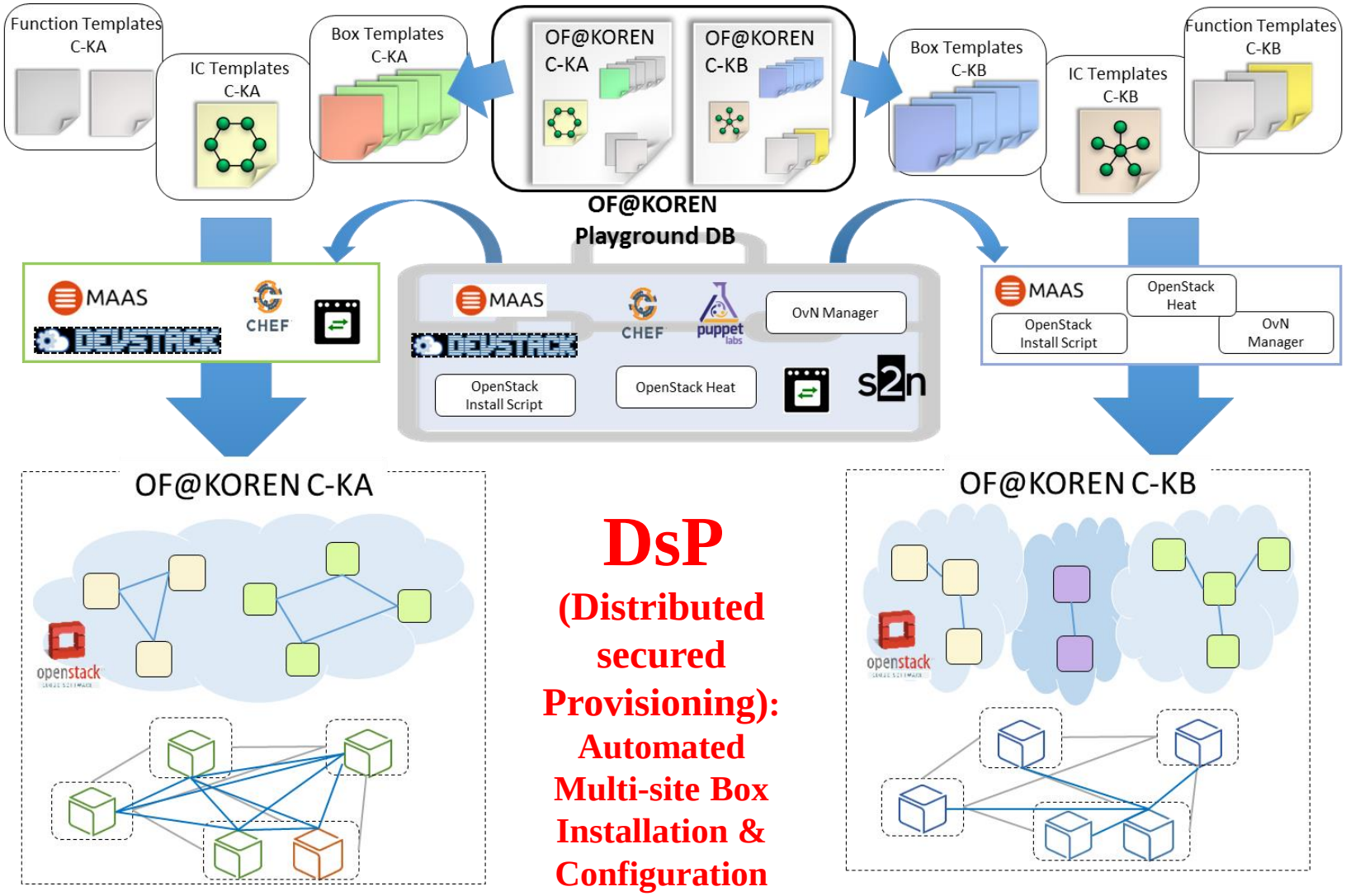
| | Salt | Ansible | Puppet | Chef |
|---|---|---|---|---|
| **Operator** | Not as mature as the other options for production OpenStack deployments. | Ursula/OSAD are the most straightforward and consistent approach to installing the OpenStack. | Oldest method to deploy OpenStack. Managed through the community process in the Big Tent. | Mature support for OpenStack. Managed through the community process in the Big Tent. |
| **Innovator** | Salt is gaining in market share and is easy to set up, but not effective at absorbing the upstream changes. | Lowest barrier to entry. Fastest growing community. | Fairly difficult to set up. Skills not as transferrable to other cloud projects. | Most difficult to set up, given the additional workstation components. Documentation from older versions conflicts with new |
| **Contributor** | Not integrated with the OpenStack development process (i.e., not a Big Tent project). | In the OpenStack Big Tent. | In the OpenStack Big Tent. | In the OpenStack Big Tent. |

Ref: "CAPS: What's best for deploying and managing OpenStack?", OpenStack Summit 2015 Tokyo

# Automated I&C for SmartX Playground: Manual vs Automation



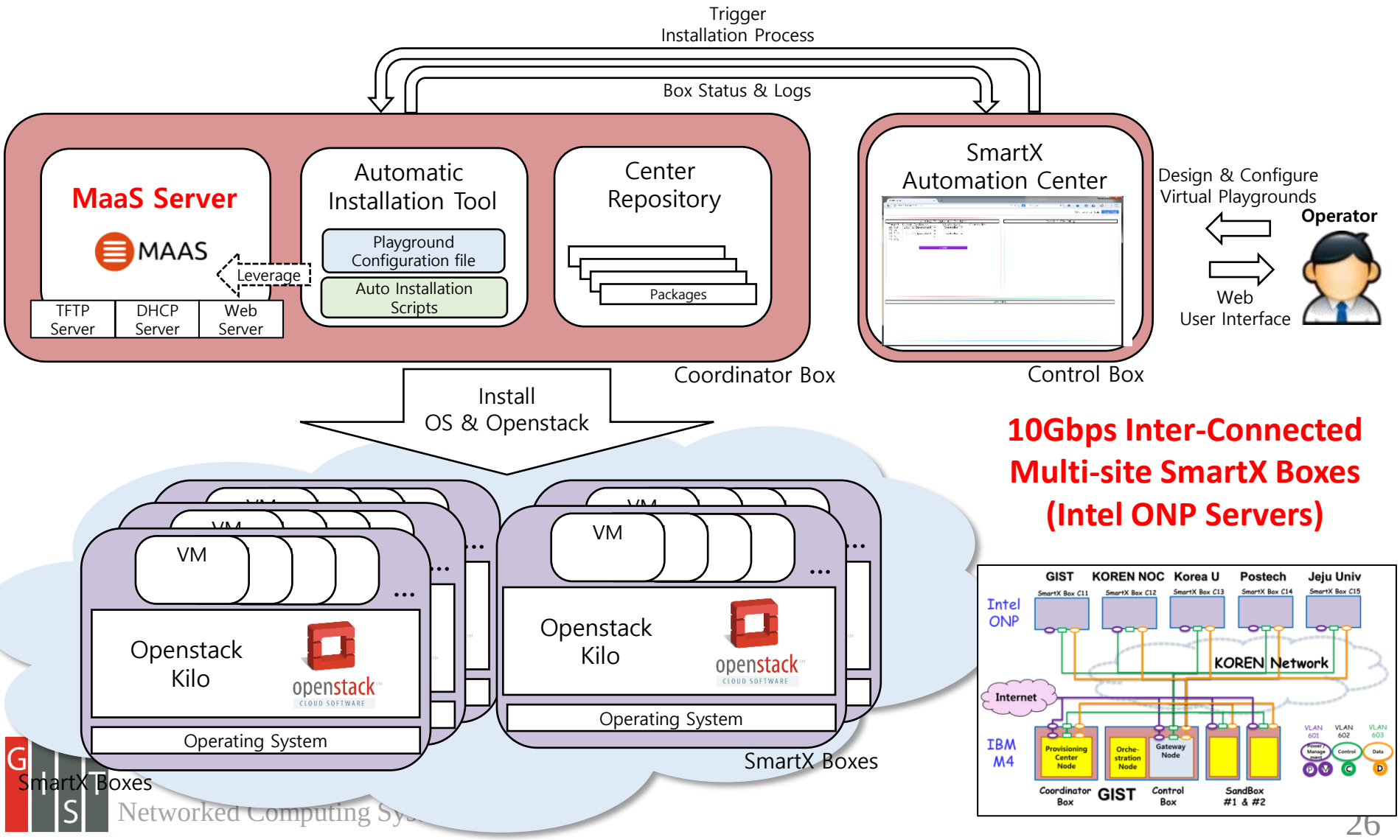
# Automated I&C for SmartX Playground: DsP Template-based I&C Concept




**DsP**

**(Distributed secured Provisioning): Automated Multi-site Box Installation & Configuration**

# Automated I&C for SmartX Playground: DsP Installer - Distributed SmartX Boxes



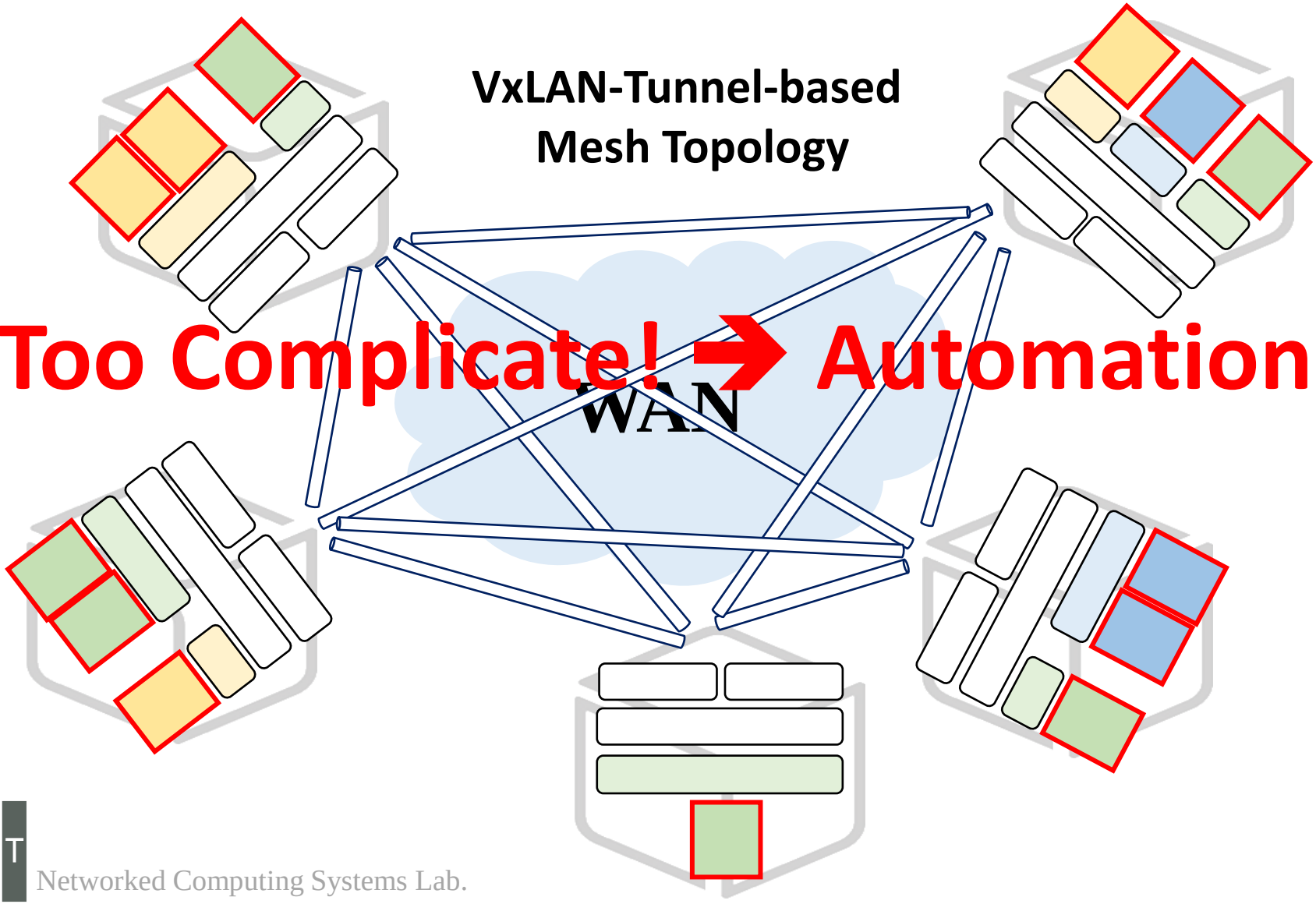
# Automated I&C for SmartX Playground: DsP Installer I&C Time for Multi-Site Boxes

| Condition | Ubuntu 14.04.1 | Openstack Icehouse(Stable) | Total Spend Time |
|---|---|---|---|
| **OpenStack Controller** | 11min 24sec | 9min 22sec | 20min 46sec |
| **OpenStack Compute** | | 4min 23sec | 15min 47sec |

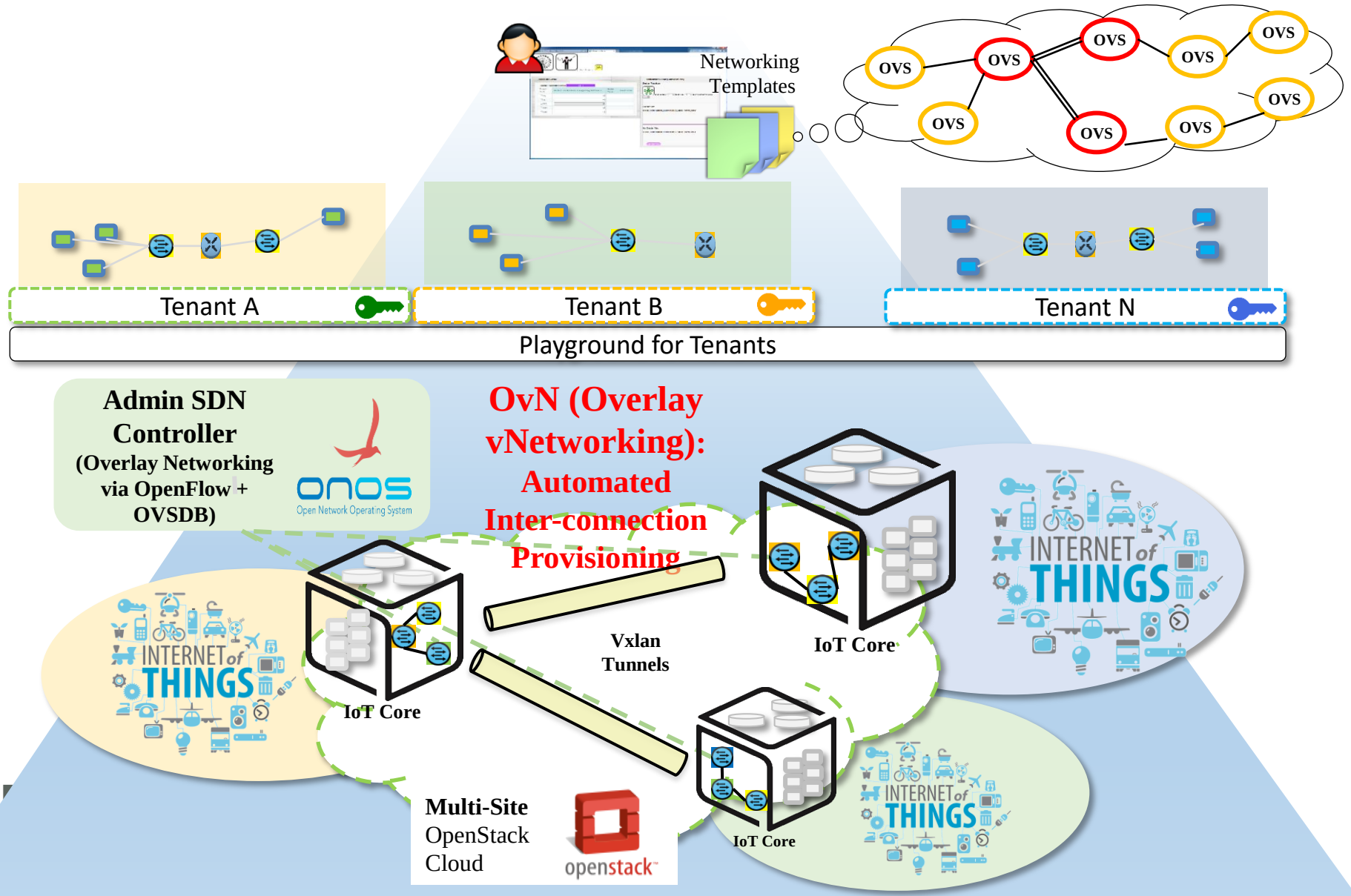
- Ubuntu Installer + DevStack Installer
- MaaS Server (from GIST Coordinator Box): IBM x3650 M4 Server
  - H/W Spec: Intel® Xeon E5-2630 6-cores, 32GB RAM, 646GB HDD
  - Xen VM: 8 core vCPU, 8GB RAM, 80GB HDD
- SmartX Boxes: Intel ONP Server
  - H/W Spec: 2x Intel® Xeon E5-2690 10-cores, 96GB RAM, 1.3TB SSD

⇒ **Selected Virtual Playgrounds are automatically installed & configured within 21 minutes**

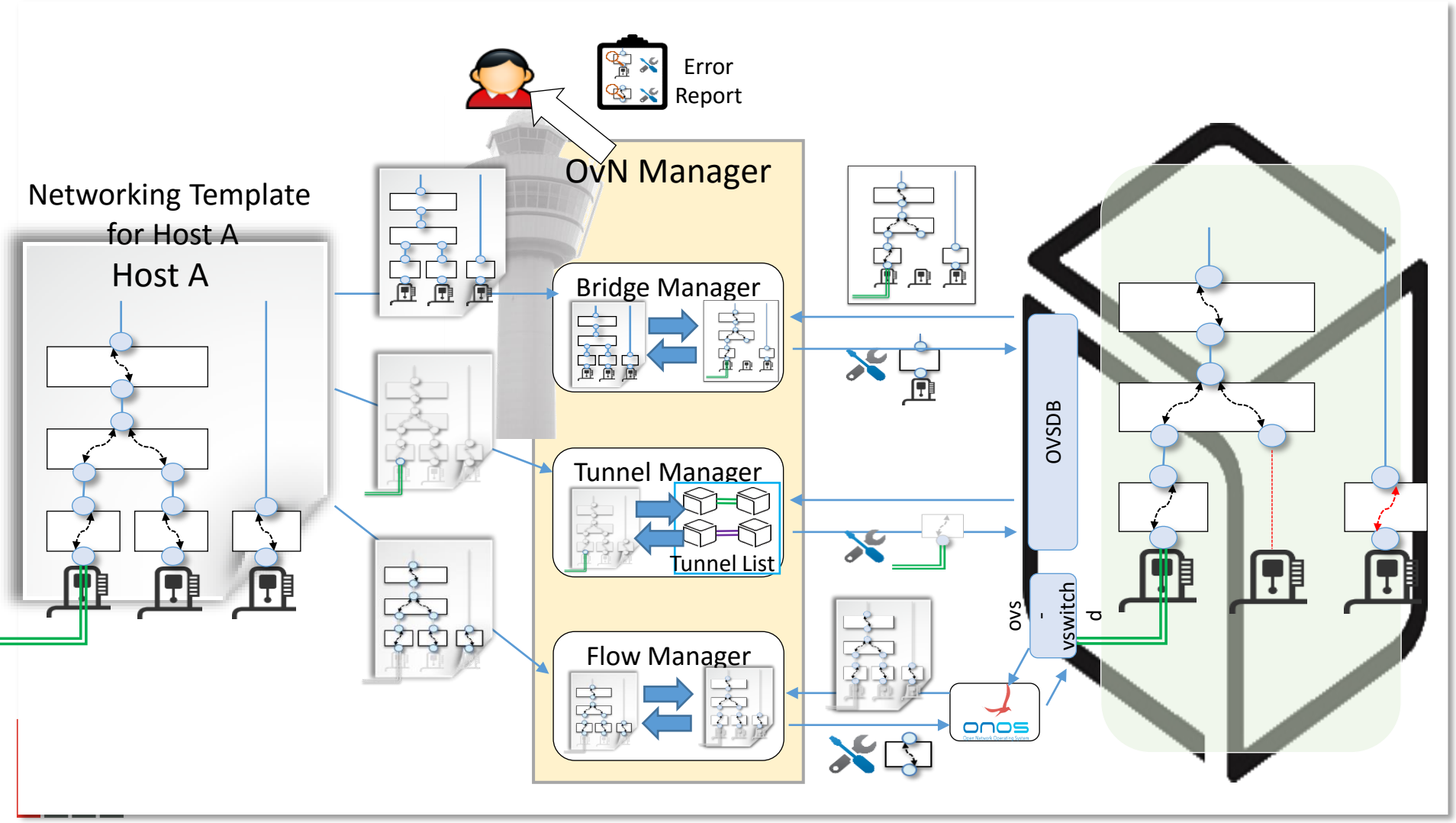
# Automated I&C for SmartX Playground: Inter-connections for Multi-Site Boxes



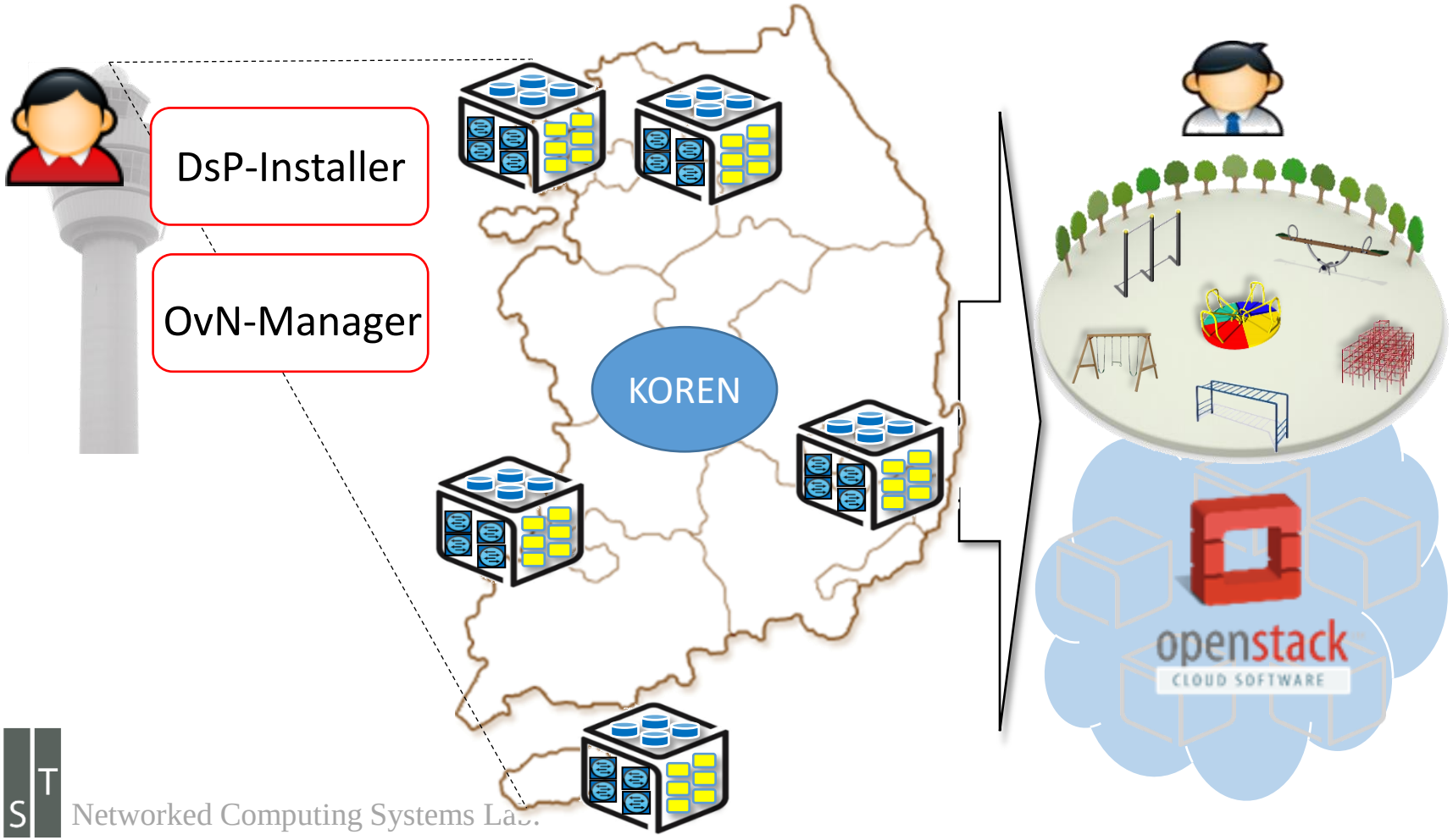
# Automated I&C for SmartX Playground: **OvN** Inter-connection for Multi-site Boxes



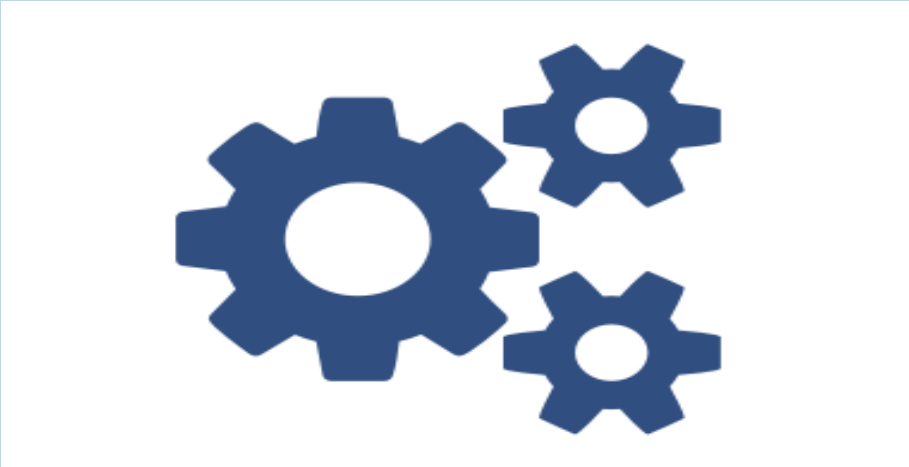
# Automated I&C for SmartX Playground: **OvN Manager** Template-based Inter-connection for Multi-site Boxes



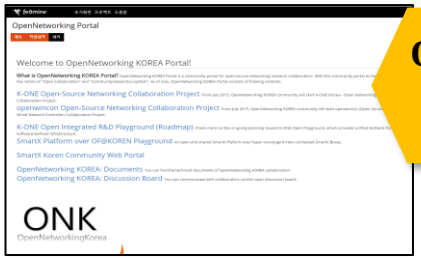
# Automated I&C for SmartX Playground: DevOps-based Automation for Multi-site Boxes (DsP) & Inter-Connections (OvN)



# OpenStack Operation & Visibility



# Operating OpenStack-leveraged SmartX Playground



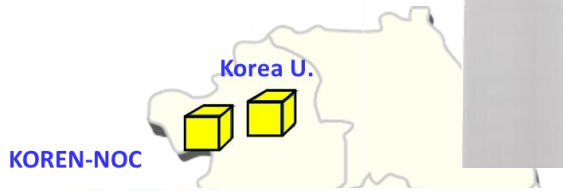

**Open SmartX Portal**
Opennetworking.kr

**SmartX.KOREN Portal**
Smartx.koren.kr

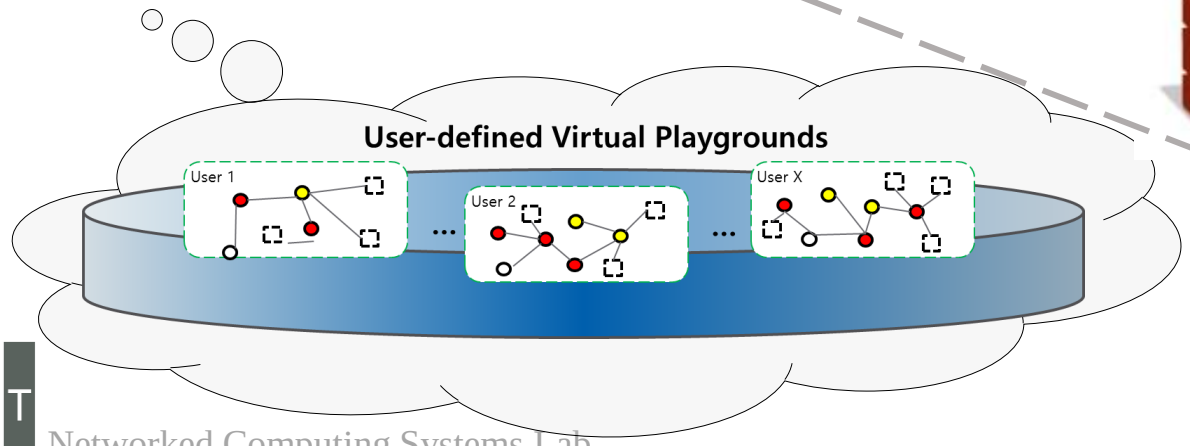

**Operator**
(ops-koren@smartx.kr)

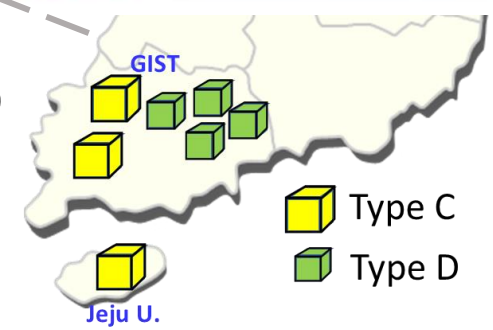


## Users



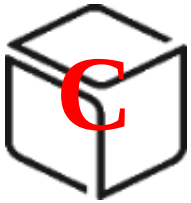




# OpenStack-leveraged SmartX Playground: Shared Resource Capacity

1 Control Box (Type C) + 4 Compute Boxes (Type C) + 4 Data Boxes (Type D)





| | Type C | Type D |
|---|---|---|
| Model | Intel ONP | Intel ONP |
| CPU | Intel(R) Xeon(R) CPU E5-2690 v2, **10-cores x2** | Intel(R) Xeon(R) CPU E5-2650 v3, 2.3Ghz, **10-cores x2** |
| RAM | **96GB** (12x8GB), 1600Mhz | **128GB** (16x8GB), 1600MHz |
| HDD /SDD | **800GB SSD x2 (RAID 0)** / 2TB SATA x2 (RAID 1) | 400GB NVMe SSD, 120GB SATA SSD / **3TB SATA x3** |
| NIC | Intel I350 (1G) x4/ **Intel X520 DA2 (10G) x2** | Intel I350(1G)x2 **Intel X710 DA2 (10G) x2** |

## Nova Compute

- 80 physical cores + 416 GB Memory
- Maximum 150 M1.small flavor VMs / Box

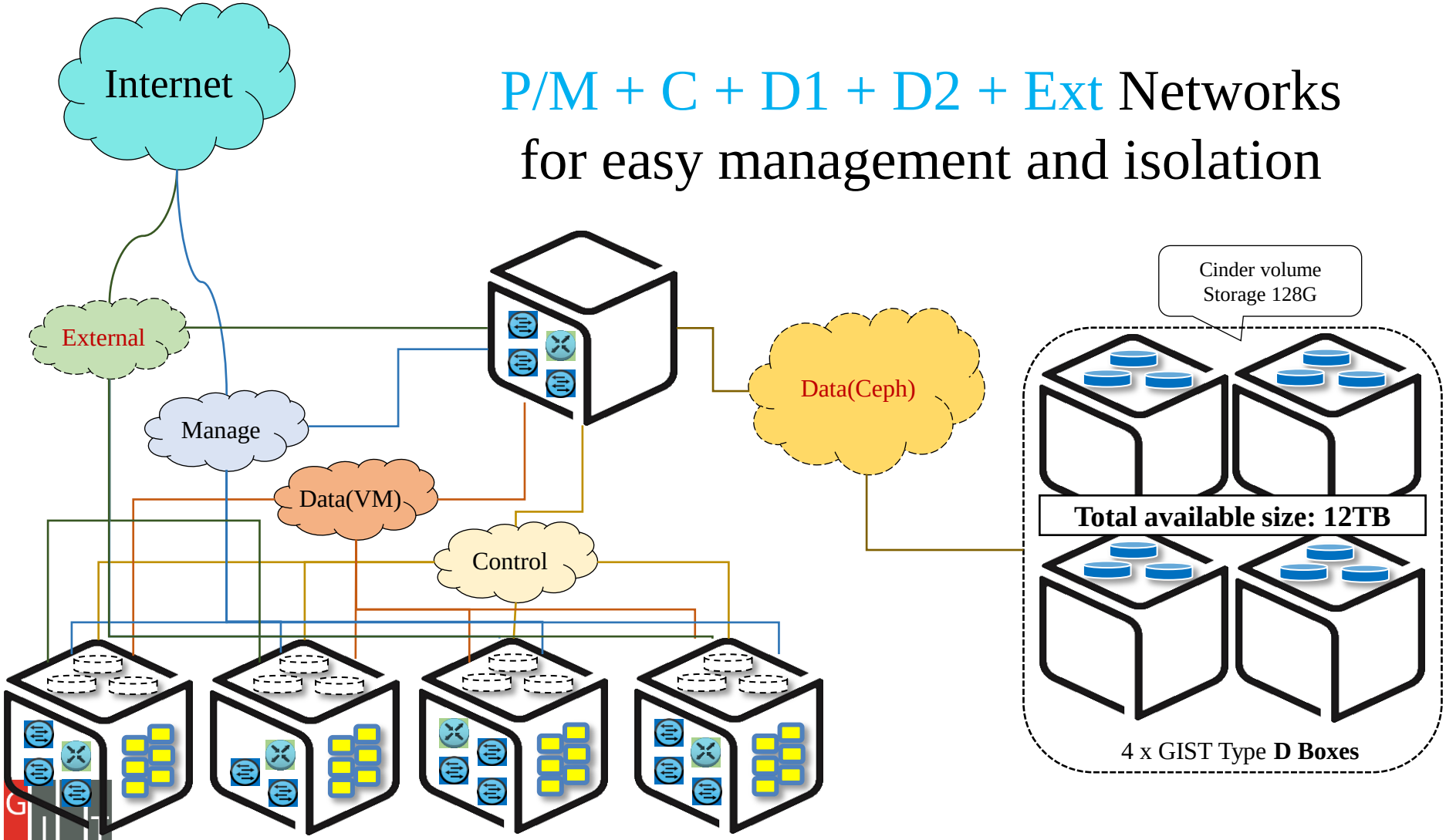
## Neutron Network

- Data (VM) Traffic:10G VXLAN
- External Traffic: 1G

## Glance/Cinder/(Swift) Storage

- 12 TB Ceph Volume

# OpenStack-leveraged SmartX Playground: Physical Inter-Connection Networks



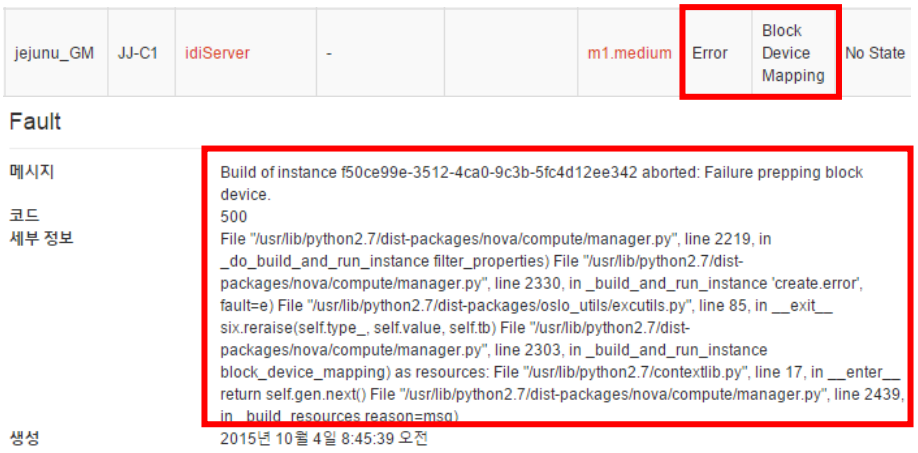
# OpenStack-leveraged SmartX Playground: Operation Challenges

- User (Developer) requirements
  - More functionalities
  - User specific customized environment

- **Efficient Management & Orchestration of Playground resources**
  - Resource pooling and isolation
  - Cost and energy-aware resource management

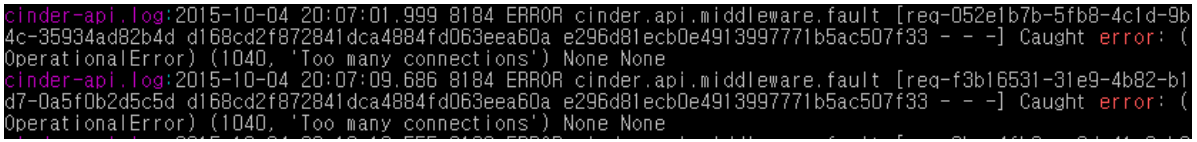
- **Reacting to unexpected circumstances**
  - Service failures due to hardware/software issues
  - Security attacks, …

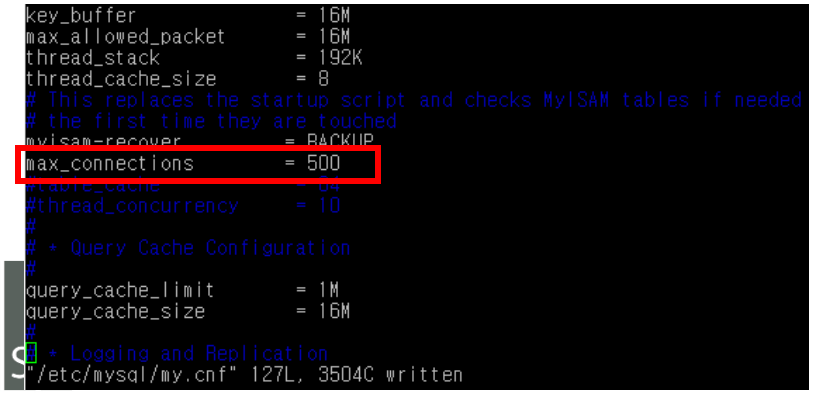
# SmartX Playground Operation Experiences:
## MySQL Max_connection Configuration Issue




- OF@KOREN Testbed Openstack Horizon
  - couldn't create VMs

- in /var/log/cinder-api.log



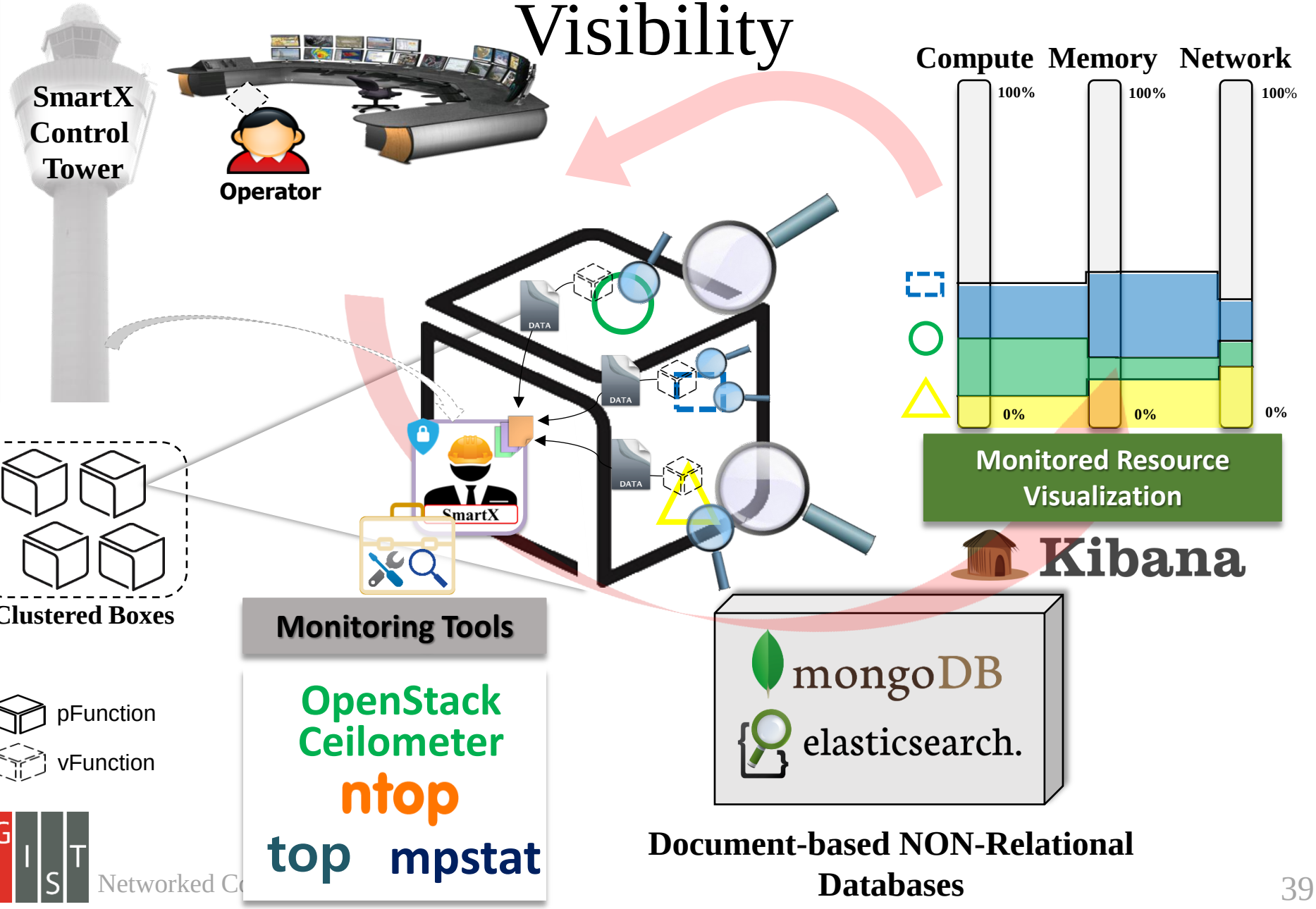




- All OpenStack Projects share single mysql instances
- But, default value of "max_connections" is just 100
- Therefore, "Too many connections" error occurred

# OpenStack-leveraged SmartX Playground: Operation Visibility Challenge


- Visibility (Monitoring & Logging)
  - Understanding about Playground (e.g., resources, functions, and services) availability and utilization
  - Can assist the operation-side reaction for unexpected circumstances
  - Need to isolated physical and virtual resources and associated inter-connection flows

# SmartX Playground Visibility: Box-centric Visibility



# SmartX Playground Visibility: p(physical)+v(virtual) Box Resources

- Data format – JSON (JavaScript Object Notation)
  - Why? Simple syntax and fast; Used for document-based database; elasticsearch uses JSON over REST API
- p+v Box-centric Resource Visibility: Measurement & Collection
  - Script-based p+v Resource Collector (SmartX Agent): Parsing resource visibility data from monitoring tool APIs
  - Physical: CPU (mpstat), Memory (top), Network (ntop)
  - Virtual: CPU/Memory/Network (OpenStack Ceilometer APIs)
- Storage – elasticsearch.
- Visualization – Kibana

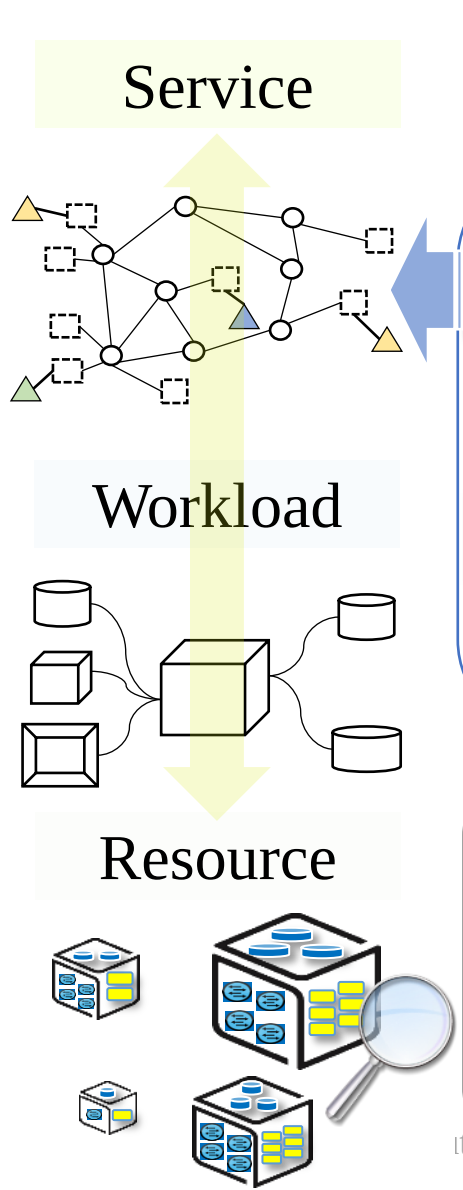
# OpenStack Orchestration

# OpenStack-leveraged SmartX Playground: Orchestration Challenges

openstack™
CLOUD SOFTWARE


- **Cost and energy-aware resource management**
- Need to deploy user-customized applications (with computing resources) to the SmartX Multi-site Cloud
- Need to support diverse (bare-metal/virtual-machine/container) functions

# SmartX Playground Orchestration: Efficient Utilization of Virtualized Resources




openstack™ CLOUD SOFTWARE

## OpenStack Heat

Functions Inter-connection

Resource-efficient Function Placement

Workload-aware Service Templates

Resource-efficient Virtual Box (VM) Creation

## OpenStack Ceilometer

- CPU Usage
- Memory Usage
- The number of Send/Receive Packets

Agent

Send data

Collector

Ceilometer API

Meta Store

Resource Metering

## SmartX Lifecycle Service Orchestration

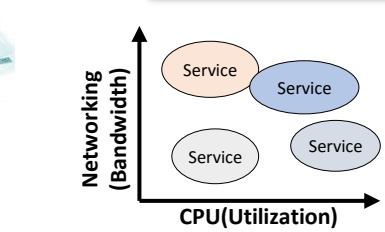
Resource-efficient Services

Workload-aware Service Templates

Profiling

Benchmarking

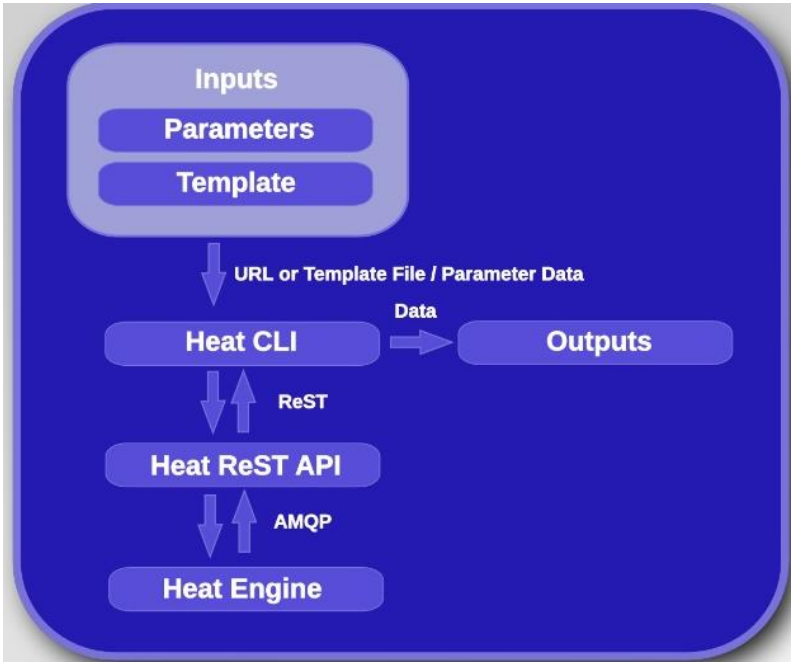
Workload




Data-/Compute-/Networking-Intensive Workloads

# SmartX Playground Orchestration: OpenStack Heat for Orchestration Service

OpenStack Heat Architecture

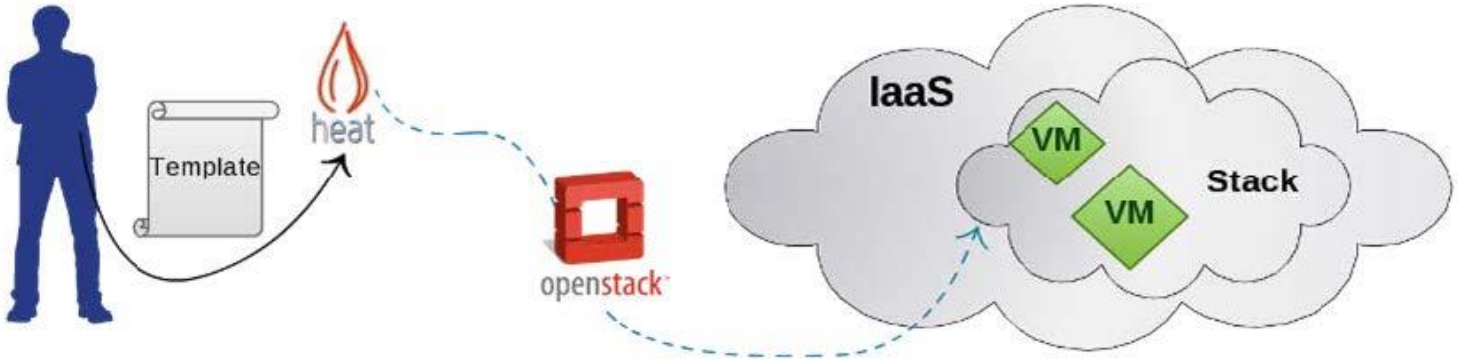



- Orchestration service based on template mechanisms
- Control complex groups of cloud resources

- 2 types of Heat templates

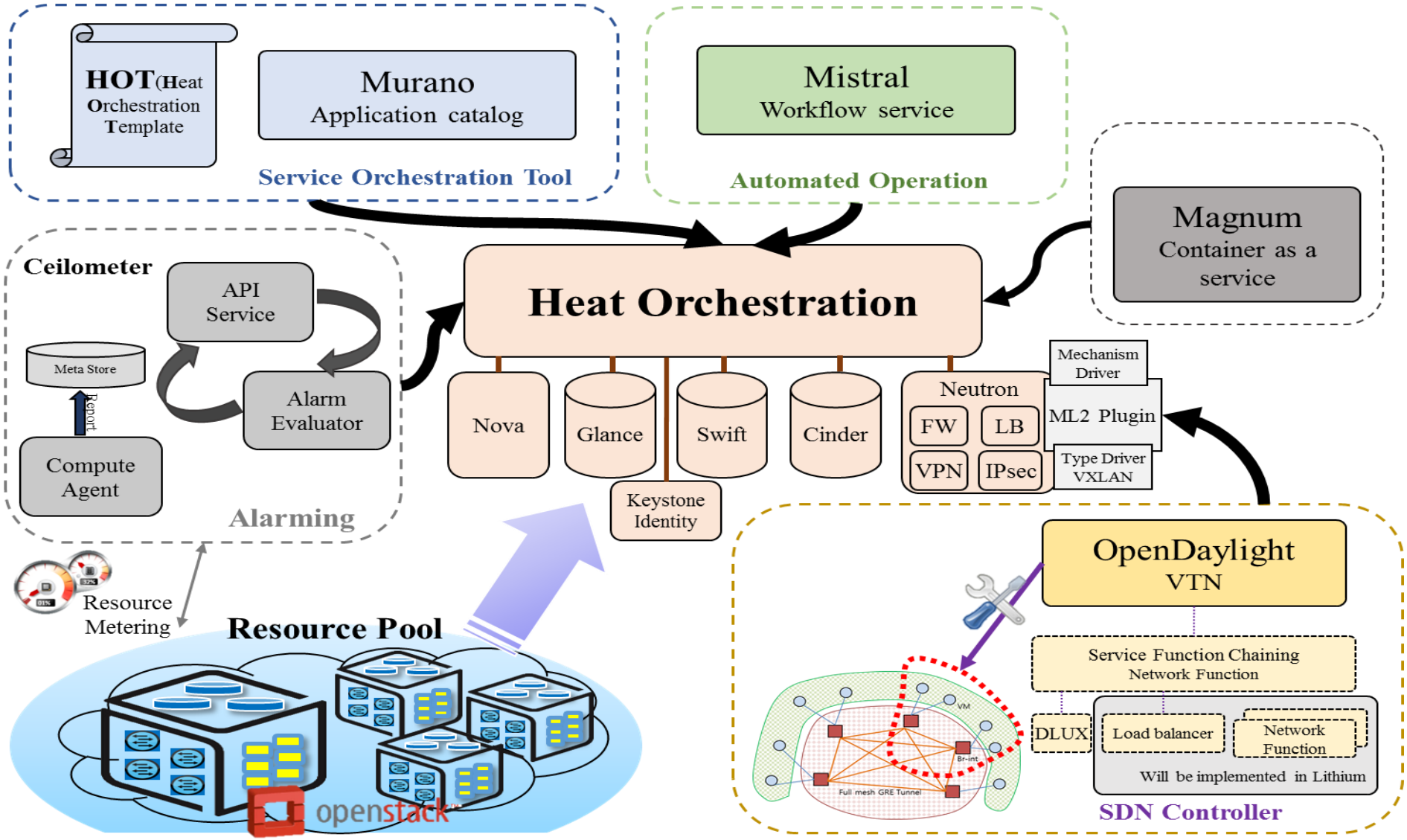
- HOT (Heat Orchestration Template) – YAML
- CFN (AWS CloudFormation) - JSON

- Parameters: Specific inputs to customize a template during deployment (Image ID, Network ID, Keypair ID, …)




Service orchestration on Cloud infrastructure side

# SmartX Playground Orchestration:
## Relationship between OpenStack Heat and others



# SmartX Playground Orchestration: OpenStack Heat Orchestration Template(HOT)

```
heat_template_version: 2015-04-30

description: Simple template to deploy a single compute instance

parameters:
  key_name:
    type: string
    label: Key Name
    description: Name of key-pair to be used for compute instance
  image_id:
    type: string
    label: Image ID
    description: Image to be used for compute instance
  instance_type:
    type: string
    label: Instance Type
    description: Type of instance (flavor) to be used

resources:
  my_instance:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image_id }
      flavor: { get_param: instance_type }
```

# heat stack-create –f 파일명.yaml –P 파라미터들

Ex) heat stack-create –f simple.yaml –P ‘key_name=userkey;image_id=ubuntu14.04;instance_type=m1.small’

# SmartX Playground Orchestration: Auto Scaling for Dynamic Resource Utilization

**Use OpenStack Heat Template with OpenStack Ceilometer & Load Balancer**

- Load Balancer is not default services in OpenStack Neutron ➔ Need to install Load Balancer

**In Heat Templates, we need to define three attributes**

- OS::Heat::AutoScalingGroup
- OS::Heat::ScalingPolicy
- OS:Ceilometer::Alarm

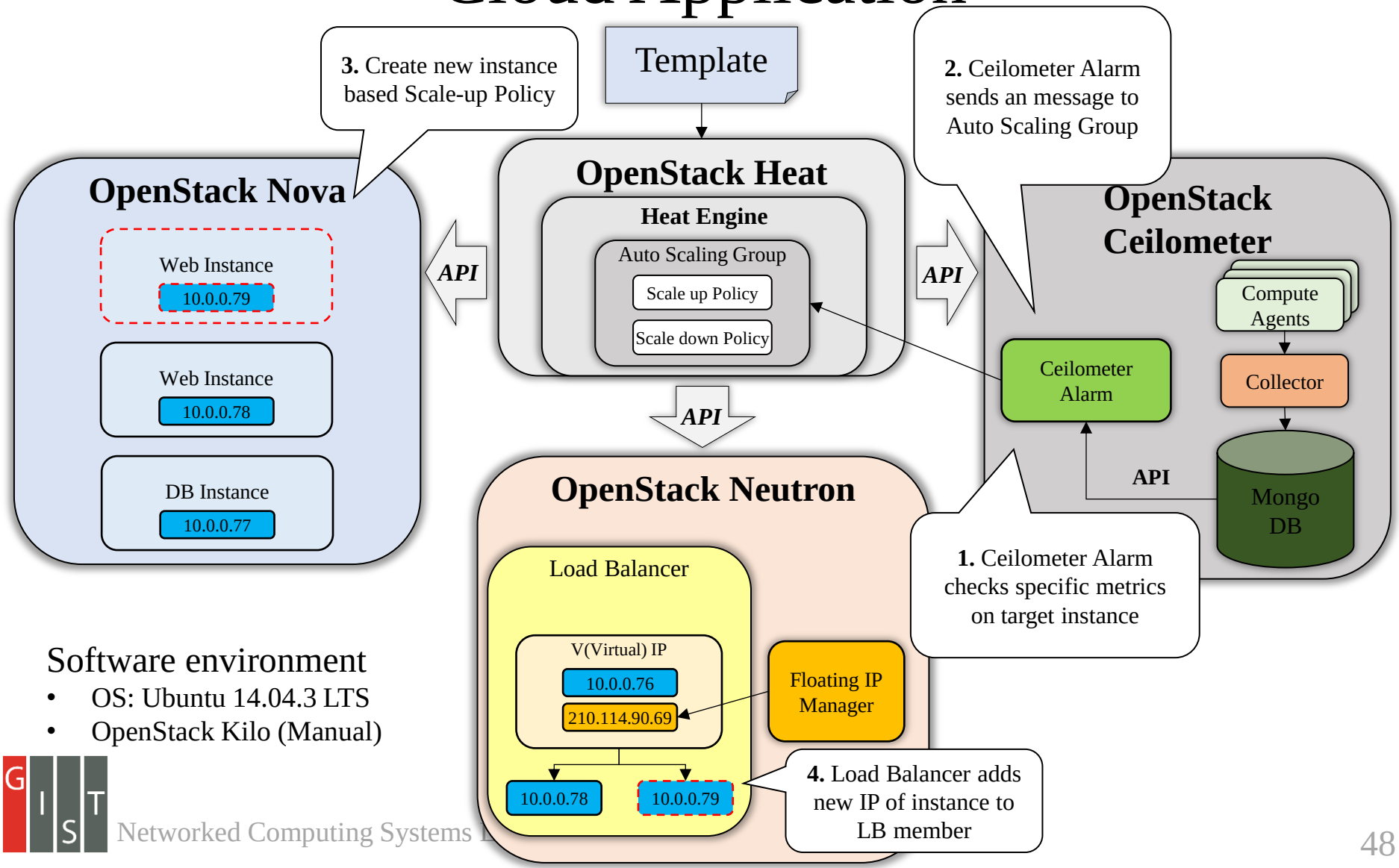
**Ways to trigger Scaling**

- Via Rest API
- Auto Scaling with Scaling Policy

**OpenStack Ceilometer Metrics for ScalingPolicy**

- Cpu_utils
- Network.Incoming.packets, Network.outgoing.packets
- Memory_usage
- Disk.write.bytes, Disk.read.bytes
- …

# SmartX Playground Orchestration: OpenStack Heat based Auto Scaling for 3-tier Cloud Application




Software environment

- OS: Ubuntu 14.04.3 LTS
- OpenStack Kilo (Manual)

# SmartX Playground Orchestration: Demo

Trigger Condition: cpu_util > 50% during 60 seconds

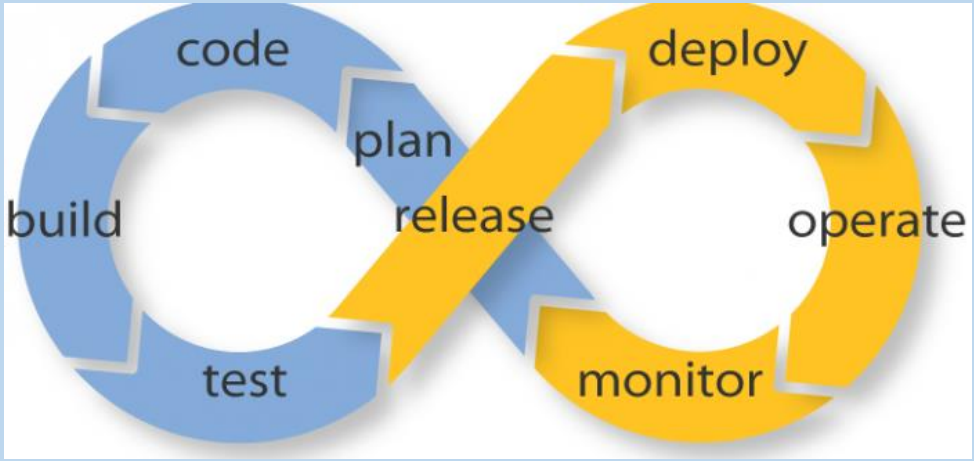
| 인스턴스 이름 | 이미지 이름 | IP 주소 | 크기 | 키 페어 | 상태 | 가용 영역 | 작업 | 전원 상태 | 생성된 이후 시간 |
|---|---|---|---|---|---|---|---|---|---|
| au-65qv-fu5hhvwojn64-whu6poj72sny-server-6uri6my7ol3l | F20 | 10.0.0.79 | m1.small | root_control_box | Active | nova | None | Running | 1분 |
| au-65qv-fdcsrni7yt7y-zk2eeool7w4u-server-sxcjj56izk5p | F20 | 10.0.0.78<br>유동 IP:<br>210.114.90.71 | m1.small | root_control_box | Active | nova | None | Running | 33분 |
| auto-db-kvb75fnuqs3m | F20 | 10.0.0.77 | m1.small | root_control_box | Active | nova | None | Running | 33분 |
| Wordpress-WebServer | | | | | | | | | 일,18시간 |
| Wordpress-DatabaseS | | | | | | | | | 일,18시간 |
| test | | | | | | | | | 일,6시간 |

Displaying 6 items

root@control-box: /home/tein/openstack

```
root@control-box:/home/tein/openstack# ceilometer alarm-list
+----------------------------------+---------------------------------+-------+----------+---------+------------+-----------------------------+
| Alarm ID                         | Name                            | State | Severity | Enabled | Continuous | Alarm condition             |
| Time constraints |
+----------------------------------+---------------------------------+-------+----------+---------+------------+-----------------------------+
| 0b0a8ce3-6596-4b0f-8a40-5b524c52512d | auto-cpu_alarm_high-ggrv3rcqepa7 | alarm | low      | True    | True       | cpu_util > 50.0 during 1 x 60s |
| None |
| e7701313-426a-44ae-95ba-7084967b681e | auto-cpu_alarm_low-ibe3xr7g2ygs  | ok    | low      | True    | True       | cpu_util < 15.0 during 1 x 600 |
s | None |
+----------------------------------+---------------------------------+-------+----------+---------+------------+-----------------------------+
root@control-box:/home/tein/openstack# clear
root@control-box:/home/tein/openstack# ceilometer statistics -m cpu_util -q metadata.user_metadata.stack=1a2bfdbc-c2ed-48ee-882d-60f4648450f8 -p 60
-a avg
+--------+---------------------+---------------------+---------------+----------+---------------------+---------------------+
| Period | Period Start        | Period End          | Avg           | Duration | Duration Start      | Duration End        |
+--------+---------------------+---------------------+---------------+----------+---------------------+---------------------+
| 60     | 2015-09-09T07:31:47 | 2015-09-09T07:32:47 | 4.39666666667 | 0.0      | 2015-09-09T07:32:30 | 2015-09-09T07:32:30 |
| 60     | 2015-09-09T07:41:47 | 2015-09-09T07:42:47 | 0.154742096506 | 0.0      | 2015-09-09T07:42:31 | 2015-09-09T07:42:31 |
| 60     | 2015-09-09T07:51:47 | 2015-09-09T07:52:47 | 86.8447412354 | 0.0      | 2015-09-09T07:52:30 | 2015-09-09T07:52:30 |
+--------+---------------------+---------------------+---------------+----------+---------------------+---------------------+
```

# CI/CD on OpenStack Playground



# Why We Need Continuous Delivery?
## - Everything changes so rapidly -

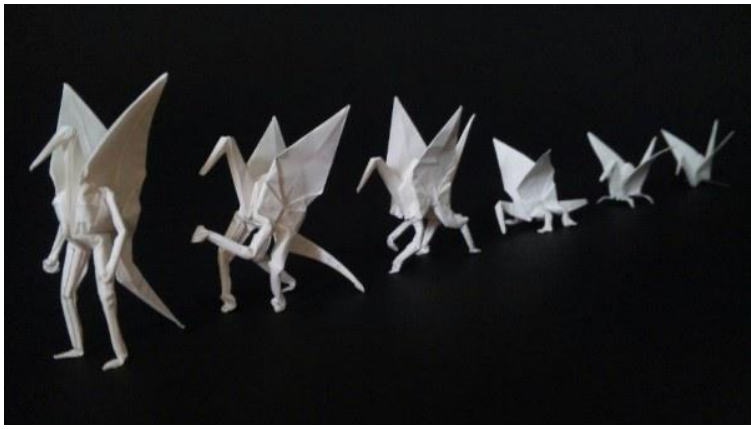


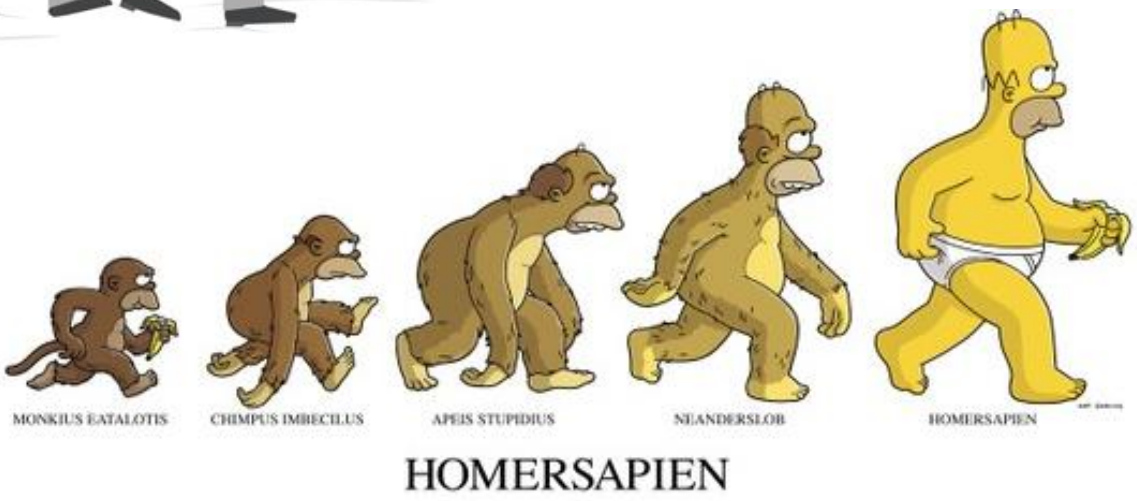

http://www.trendmonitor.co.kr/Data/CKOREA/4%5B104%5D.jpg

koreancontent.kr

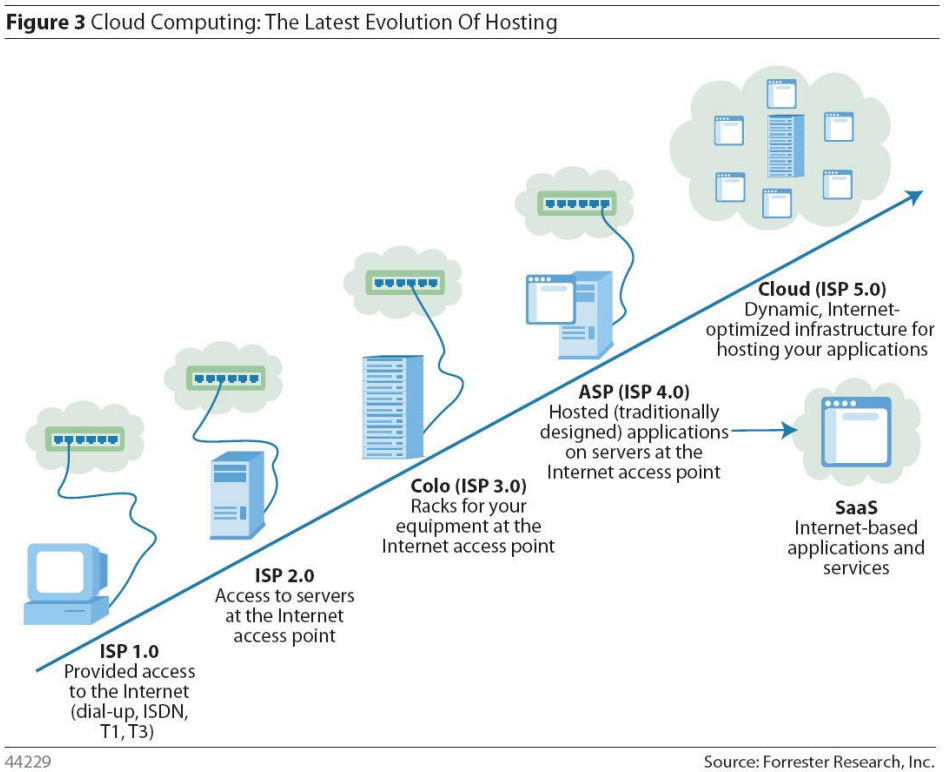
# Why We Need Continuous Delivery? (1/3)

## - Everything has its' evolution -



# Why We Need Continuous Delivery? (2/3)
## - IT also has RAPID evolution -

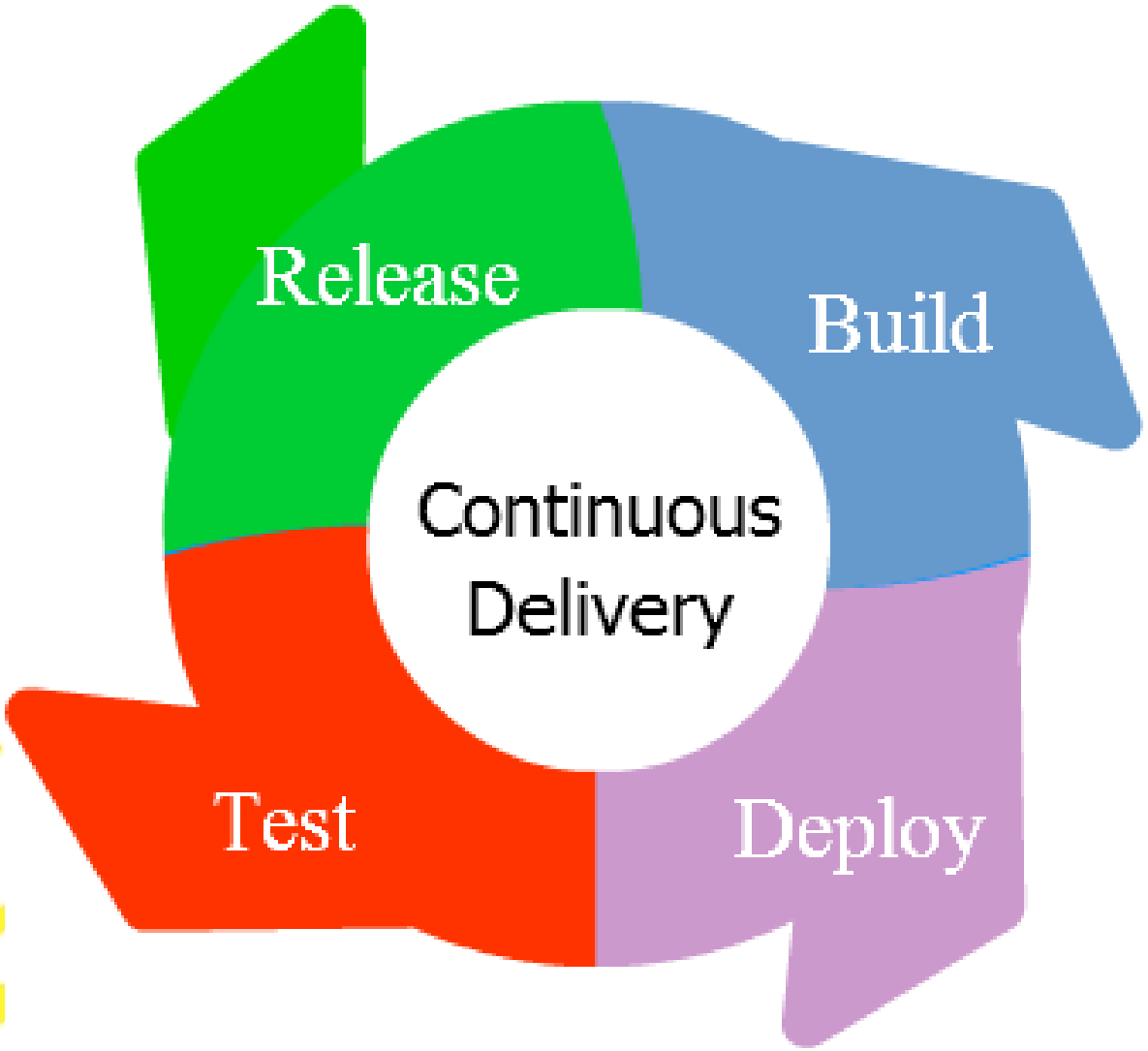



# Why We Need Continuous Delivery? (3/3)

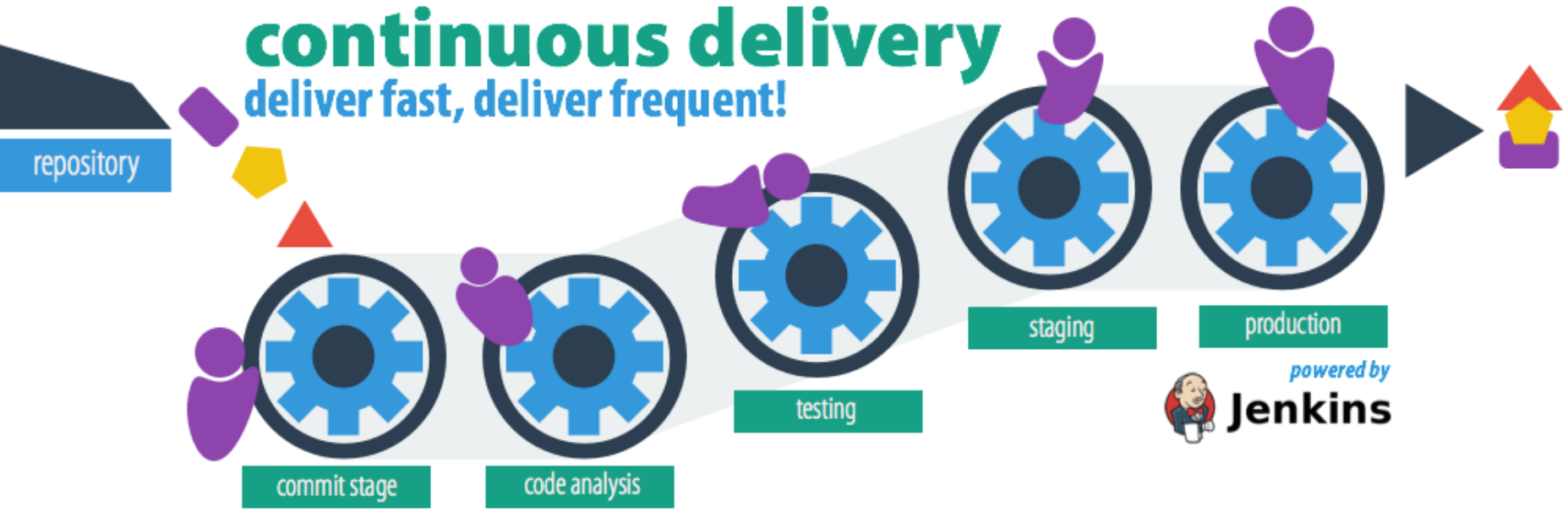
## - However, we can't do it "By" one time -


# What is Continuous Delivery? (1/2)

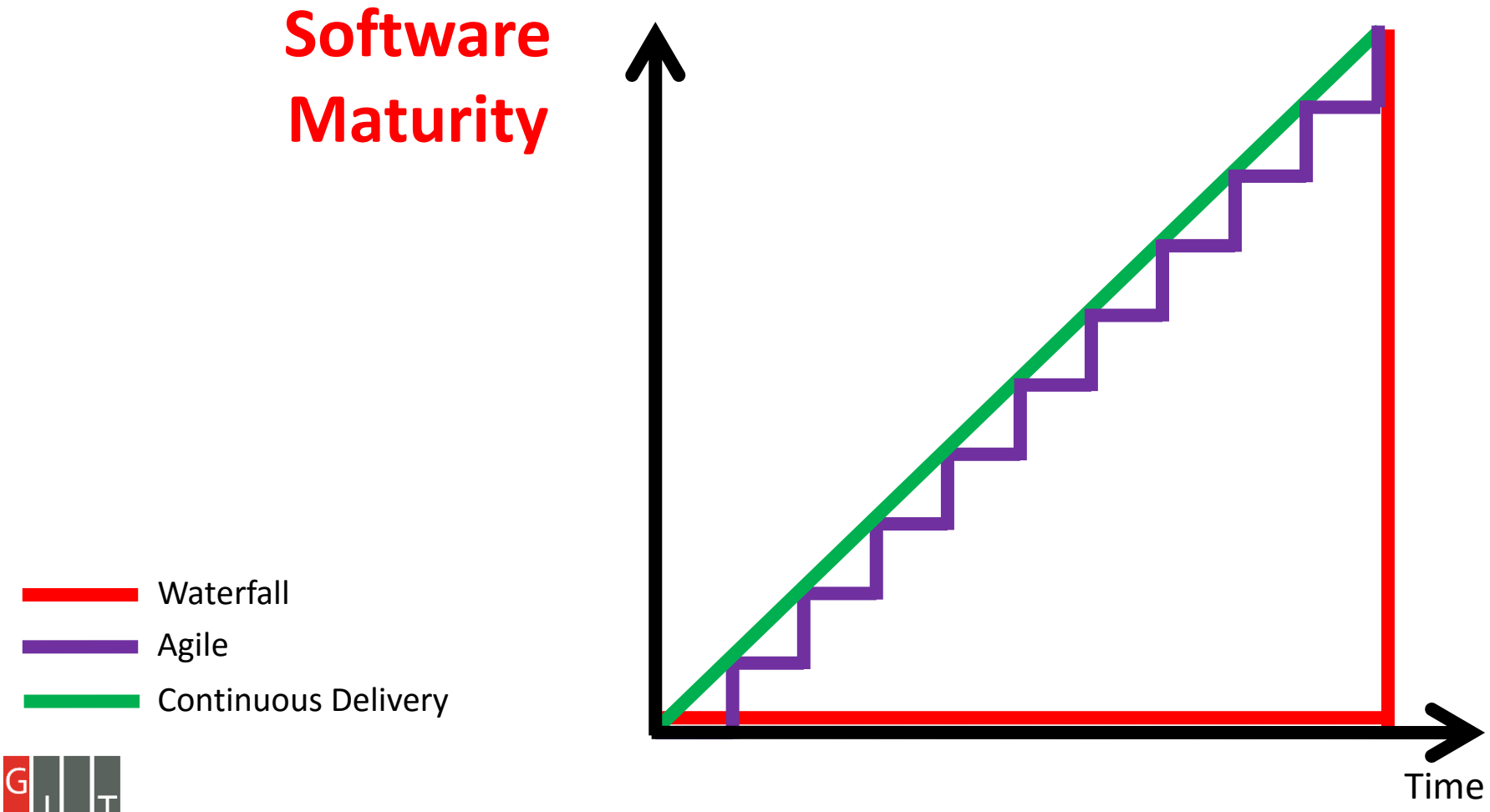
## - Concept Diagram -



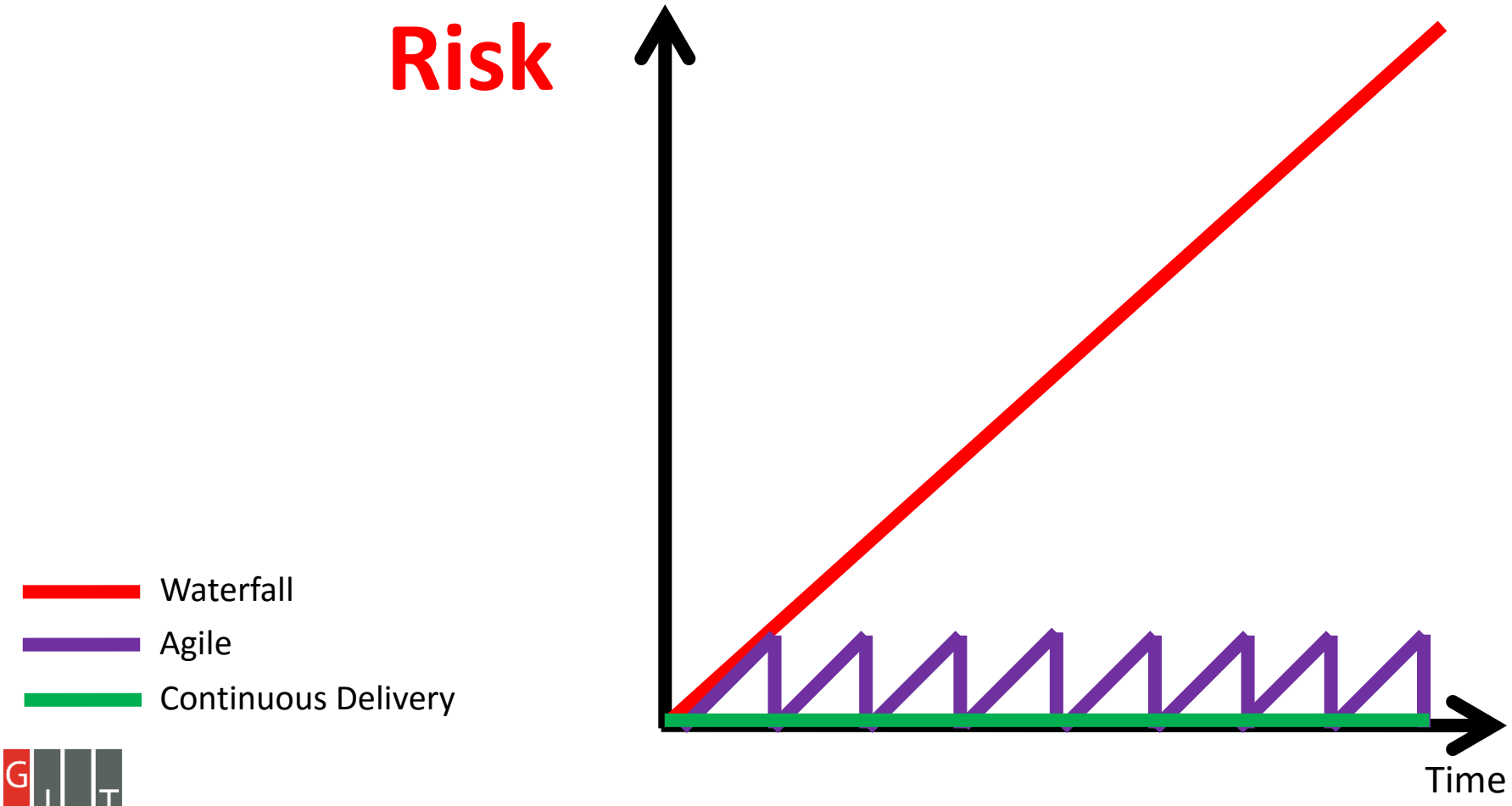

# What is Continuous Delivery? (2/2)
## - Continuous Delivery Pipeline -



# Software Development Methodology Evolution: Waterfall, Agile, Continuous Delivery (1/2)

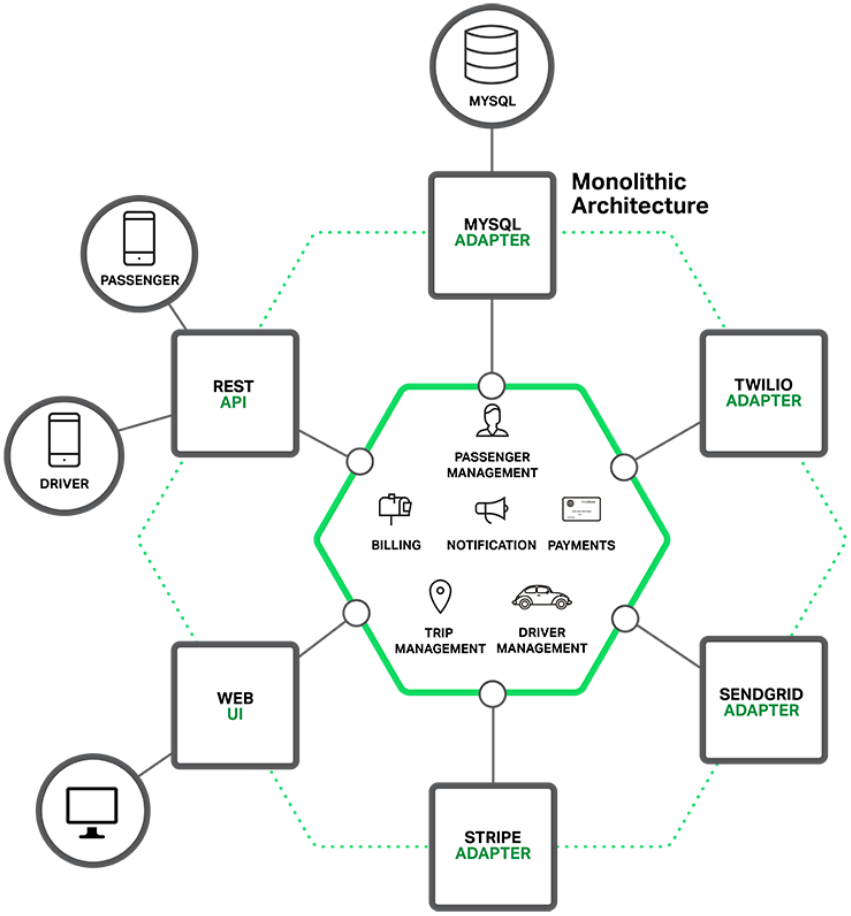


# Software Development Methodology Evolution: Waterfall, Agile, Continuous Delivery (2/2)

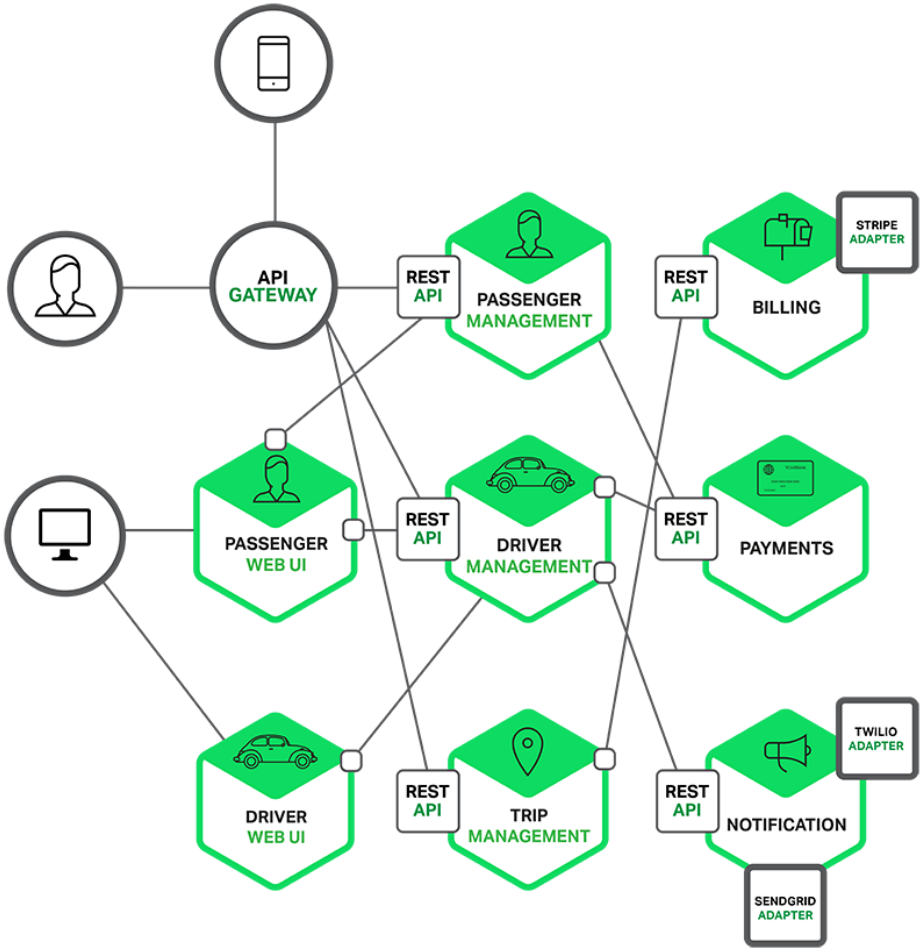


# Software Development Methodology Evolution: Monolithic Applications ➔ Microservice

Monolithic Architecture



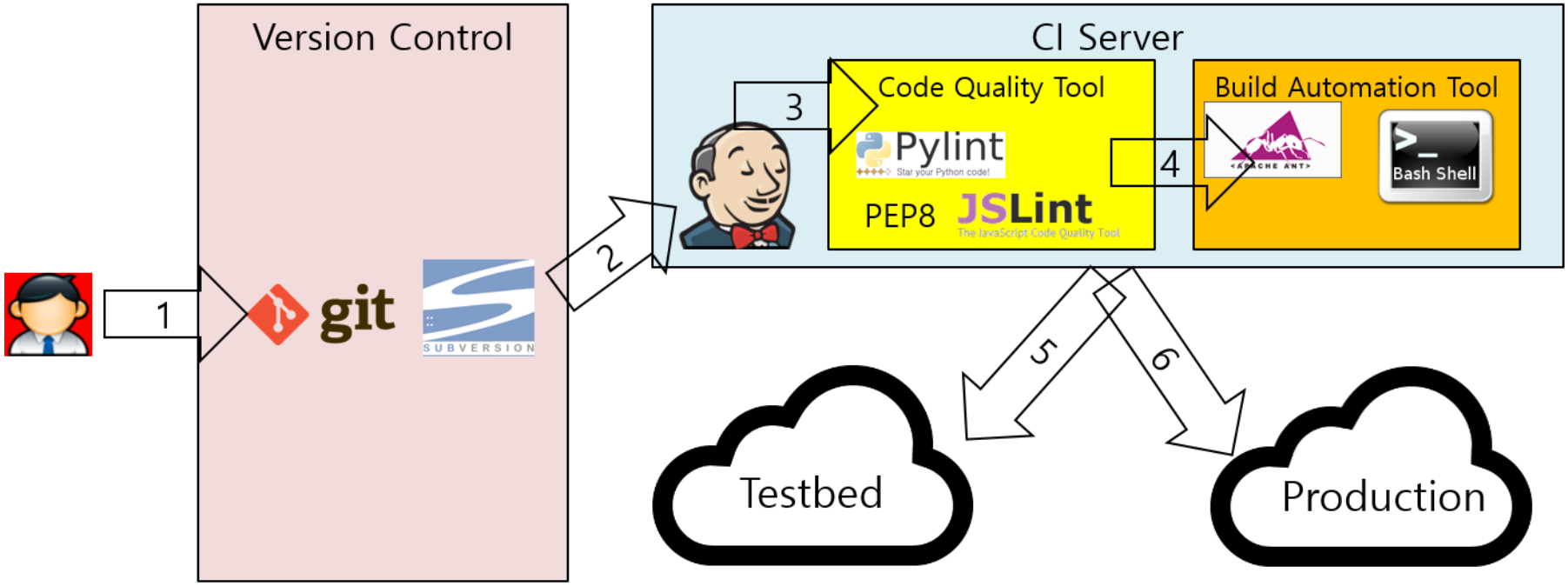

Microservice Architecture



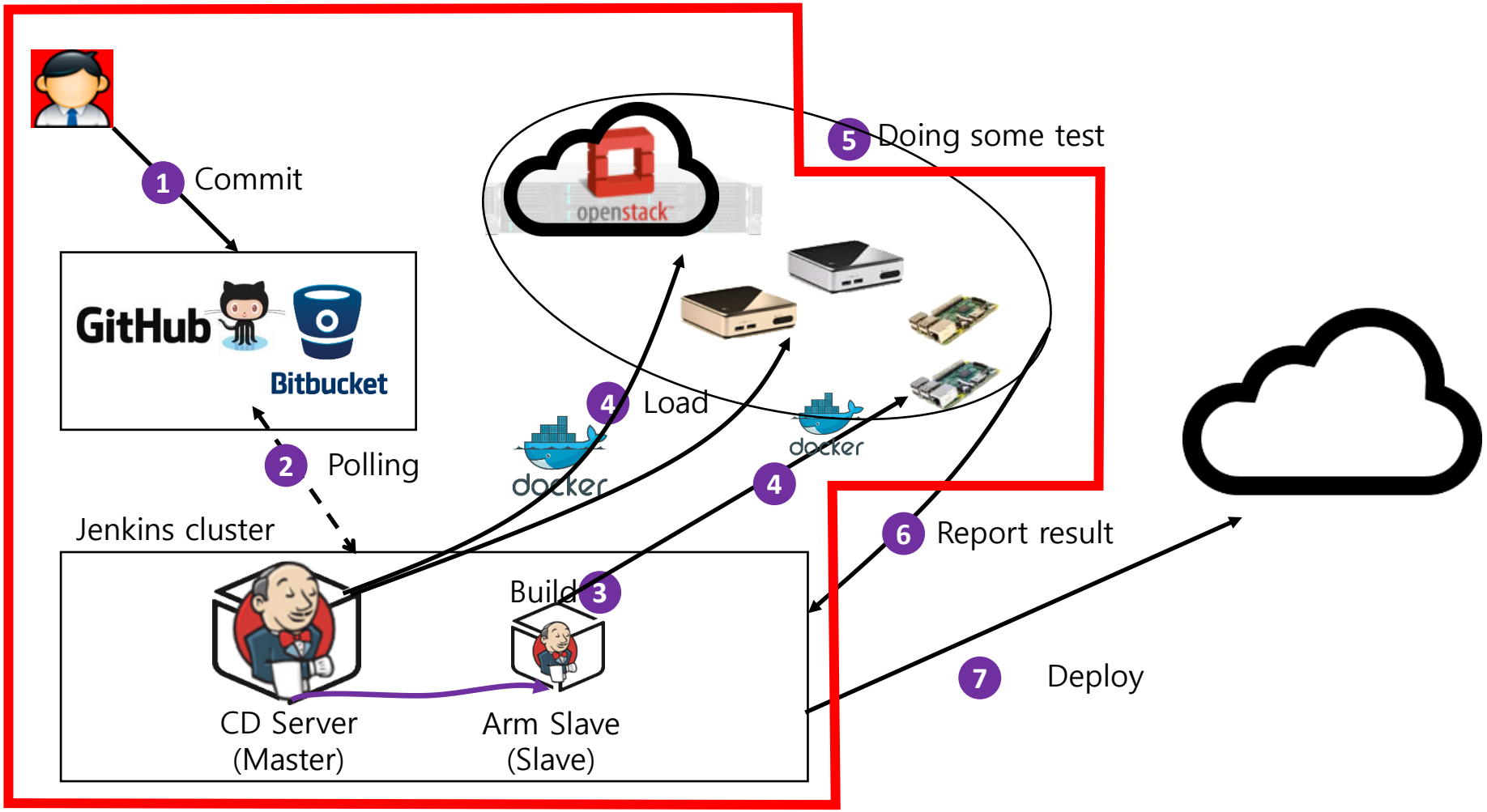



https://www.nginx.com/blog/introduction-to-microservices/

# Continuous Delivery: How (1/3)
## - Workflow -



# Continuous Delivery: How (2/3)
## - More detailed workflow -

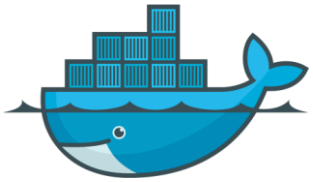



※Inner part of red border is what we have done.

# Continuous Delivery: How (3/3)
## - Useful tools -


Git & SVN
소스 코드 관리를 위한 버전 관리 시스템.
협업을 통한 소스 코드 통합, 버전 관리를 위해서는 필수적!

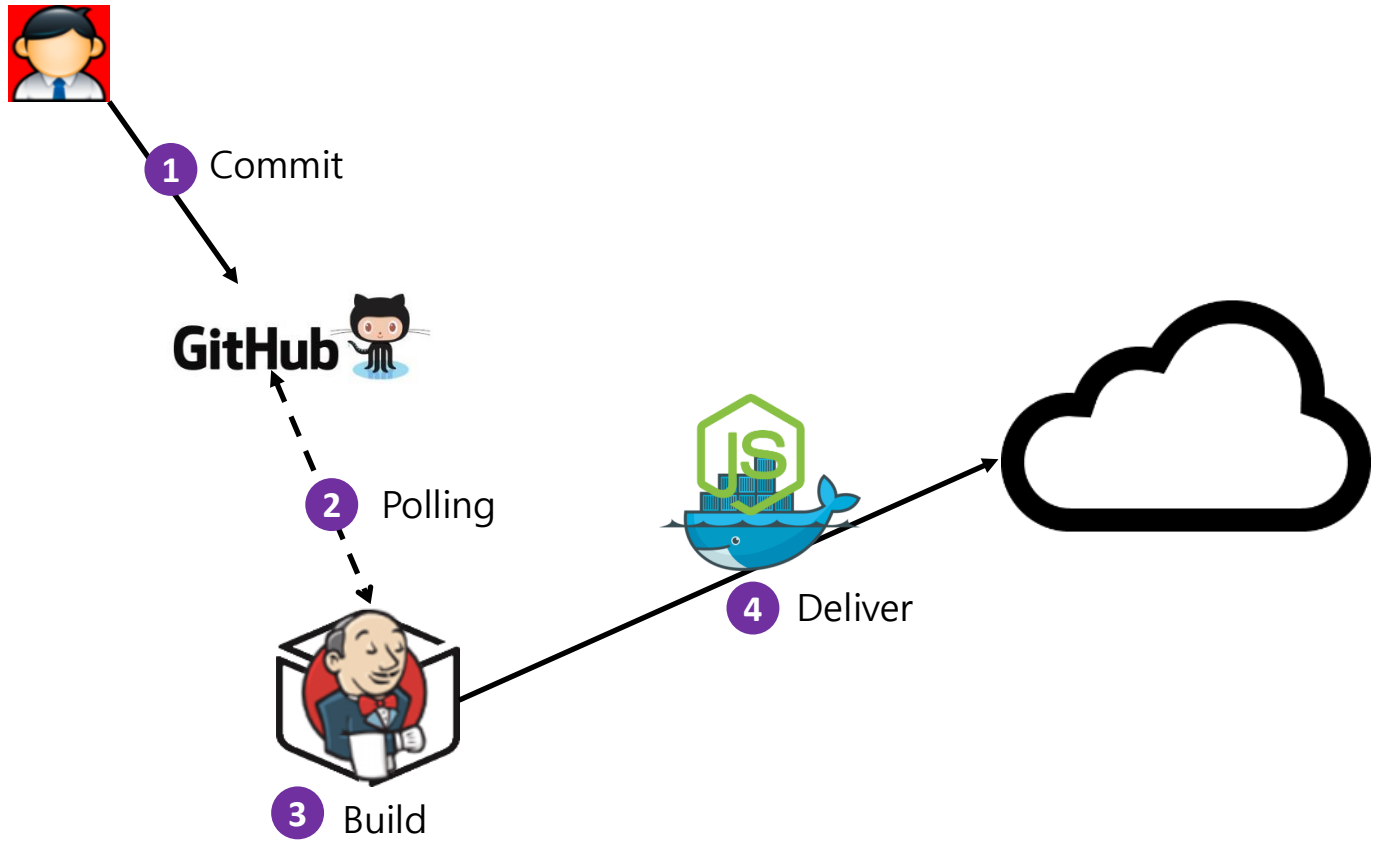
Jenkins
가장 유명한 오픈소스 Continuous Integration Software
300개 이상의 플러그인을 통해 코드의 빌드부터 테스트, 배포까지 자동으로 할 수 있도록 설정 할 수 있다.



Docker
Container 에서 가장 핫한 오픈소스 프로젝트
어플리케이션의 빌드 및 배포를 자동화 하는데 강점을 가진다.

# Continuous Delivery Demonstration

## Continuous Delivery of Web Services over SmartX Multi-site Cloud Playground



Gwangju Institute of
Science & Technology


# *Thank you!*

*netcs@smartx.kr*