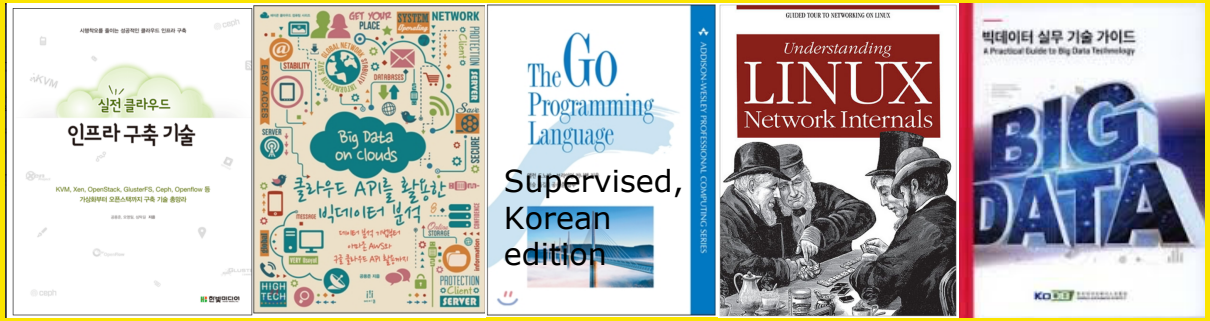
Embracing Clouds

Andrew yongjoon kong
CloudComputingCell, kakao
andrew.kong@kakaocorp.com

Who am I

Andrew. Yongjoon kong

- Cloud Technical Advisory for Government Broad Cast Agency
- Adjunct Prof. Ajou Univ
- Korea Data Base Agency Acting Professor for Bigdata
- Member of National Information Agency Bigdata Advisory committee
- KT cloudware Tech lead(ex)!
- Kakao → Daum Kakao → Kakaocorp, Cloud Computing Cell lead
- Talks
  - Embrace clouds (2017, openstack days, korea)
  - Full route based network with linux (2016, netdev, Tokyo)
  - SDN without SDN (2015, openstack, Vancouber)



2nd Editions are coming…

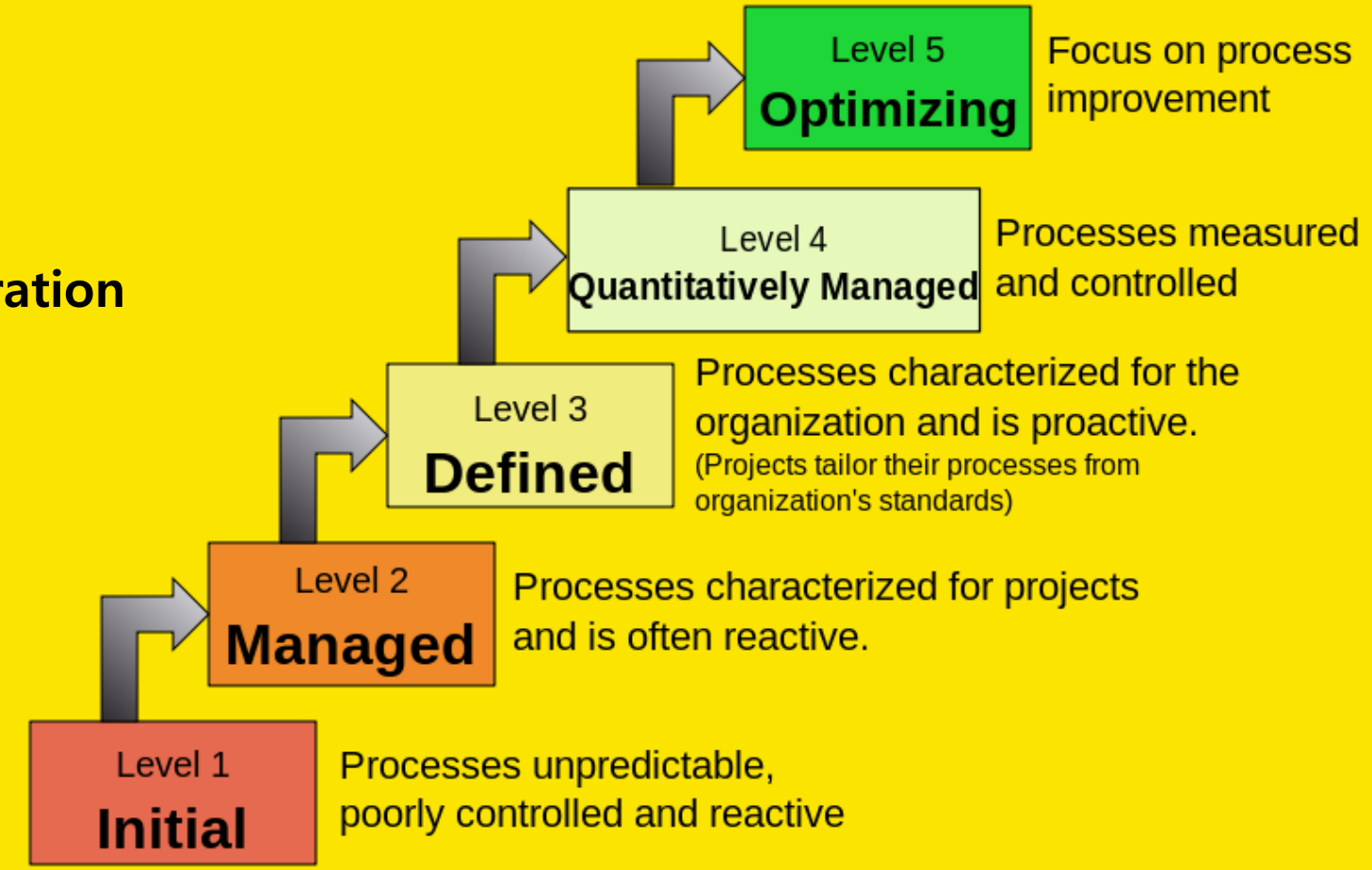
Everyone says

How Far did you go with your cloud?

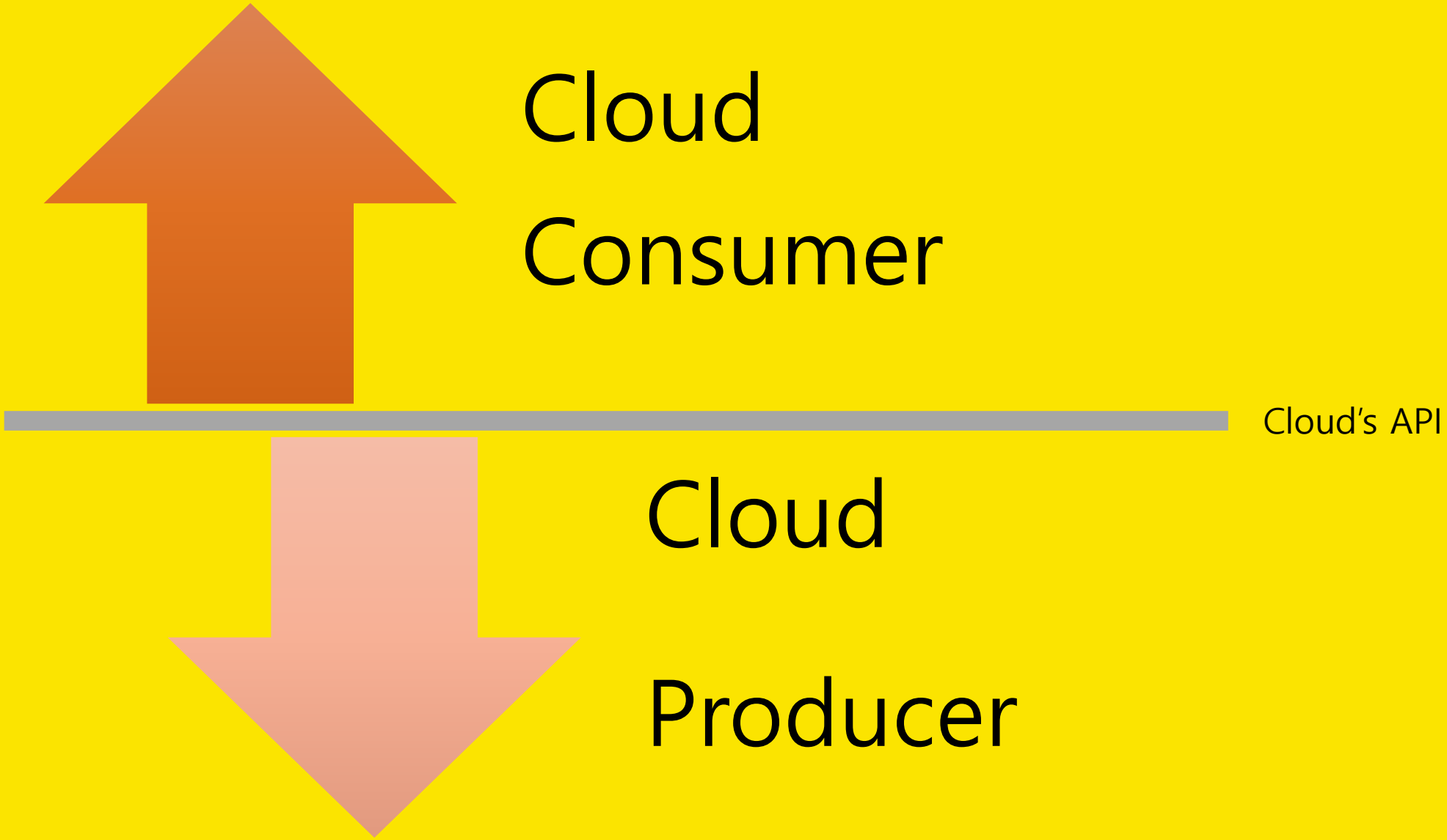
CMMI Model

Capability Maturity Model Integration

developed by CMU



Perspectives



How Far did you go with your cloud? like Kakao

Cloud Producer: CMMI Dev perspective

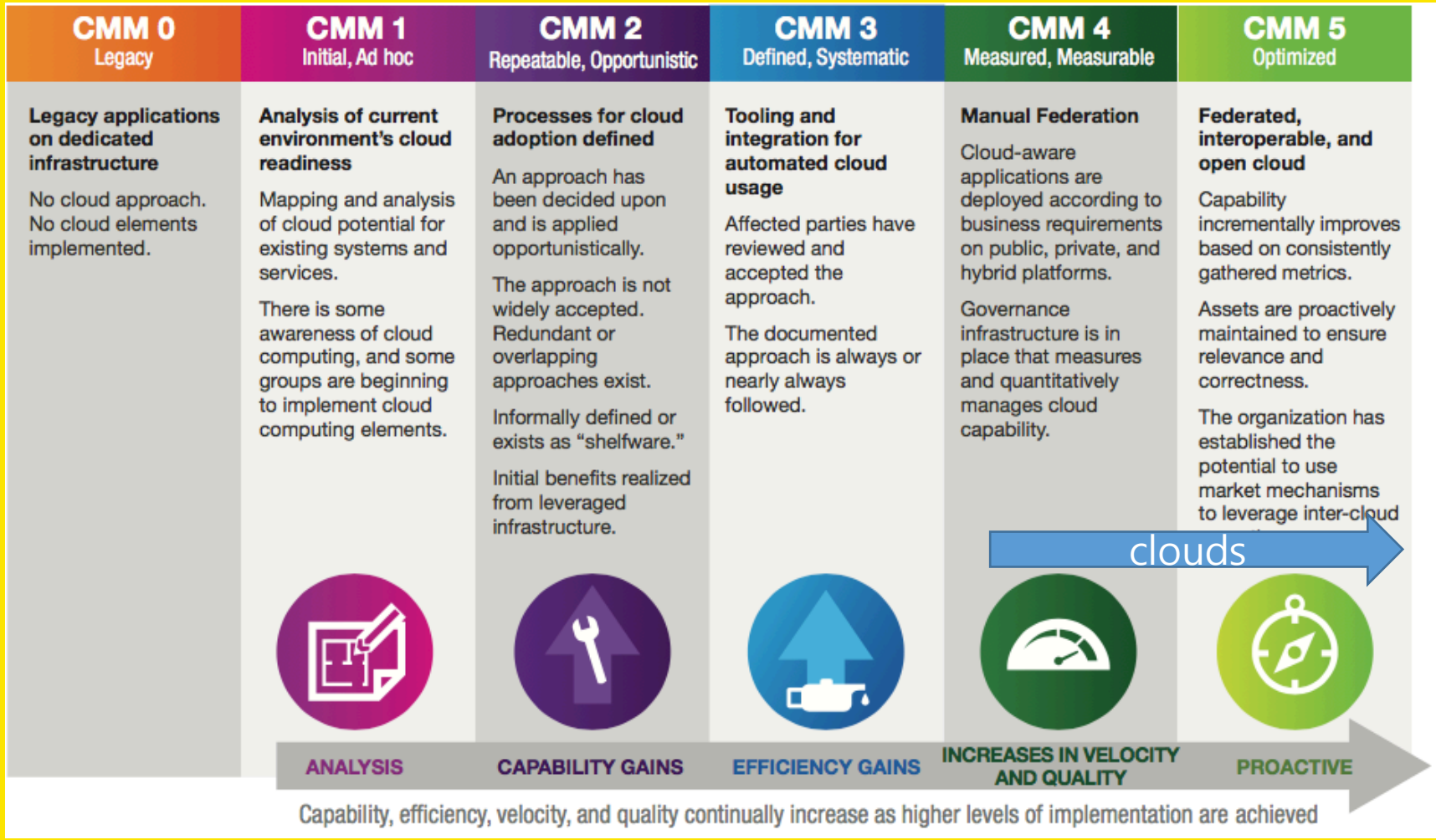
Dreams

Maturity

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
Culture & Organization	• Teams organized based on platform/ technology • Defined and documented processes	• One backlog per team • Adopt agile methodologies • Remove team boundaries	• Extended team collaboration • Remove boundary dev/ ops • Common process for all changes	• Cross-team continuous improvement • Teams responsible all the way to production	• Cross functional teams
Build & Deploy	• Centralized version control • Automated build scripts • No management of artifacts • Manual deployment • Environments are manually provisioned	• Polling CI builds • Any build can be re-created from source control • Management of build artifacts • Automated deployment scripts • Automated provisioning of environments	• Commit hook CI builds • Build fails if quality is not met (code analysis, performance, etc.) • Push button deployment and release of any releasable artifact to any environment • Standard deployment process for all environments	• Team priorities keeping codebase deployable over doing new work • Builds are not left broken • Orchestrated deployments • Blue Green Deployments	• Zero touch Continuous Deployments
Release	• Infrequent and unreliable releases • Manual process	• Painful infrequent but reliable releases	• Infrequent but fully automated and reliable releases in any environment	• Frequent fully automated releases • Deployment disconnected from release • Canary releases	• No rollbacks, always roll forward
Data Management	• Data migrations are performed manually, no scripts	• Data migrations using versioned scripts, performed manually	• Automated and versioned changes to datastores	• Changes to datastores automatically performed as part of the deployment process	• Automatic datastore changes and rollbacks tested with every deployment
Test & Verification	• Automated unit tests • Separate test environment	• Automatic Integration Tests • Static code analysis • Test coverage analysis	• Automatic functional tests • Manual performance/ security tests	• Fully automatic acceptance tests • Automatic performance/security tests • Manual exploratory testing based on risk based testing analysis	• Verify expected business value • Defects found and fixed immediately (roll forward)
Information & Reporting	• Baseline process metrics • Manual reporting • Visible to report runner	• Measure the process • Automatic reporting • Visible to team	• Automatic generation of release notes • Pipeline traceability • Reporting history • Visible to cross-silo	• Report trend analysis • Real time graphs on deployment pipeline metrics	• Dynamic self-service of information • Customizable dashboards • Cross-reference across organizational boundaries

How Far did you go with your cloud? like Kakao

Cloud Consumer: CMMI Cloud Perspective



Open Data Center Alliance, Inc

Where are you from CMMI-DEV perspective? kakao case

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
Culture & Organization	• Teams organized based on platform/ technology • Defined and documented processes	• One backlog per team • Adopt agile methodologies • Remove team boundaries	• Extended team collaboration • Remove boundar ops • Common process for all changes	• Cross-team continuous improvement • Teams responsible all the way to production	• Cross functional teams
Build & Deploy	• Centralized version control • Automated build scripts • No management of artifacts • Manual deployment • Environments are manually provisioned	• Polling CI builds • Any build can be re-created from source control • Management of build artifacts • Automated deployment scripts • Automated provisioning of environments	• Commit hook CI builds • Build fails if quality is not met (code analysis, performance, etc.) • Push button deployment and release of any releasable artifact to any environment • Standard deployment process for all environments	• Team priorities keeping codebase deployable over doing new work • Builds are not left broken • Orchestrated deployments • Blue Green Deployments	• Zero touch Continuous Deployments
Release	• Infrequent and unreliable releases • Manual process	• Painful infrequent but reliable releases	• Infrequent but fully automated and reliable releases in any environment	• Frequent fully automated releases • Deployment disconnected from release • Canary releases	• No rollbacks, always roll forward
Data Management	• Data migrations are performed manually, no scripts	• Data migrations using versioned scripts, performed manually	• Automated and versioned changes to datastores	• Changes to datastores automatically performed as part of the deployment process	• Automatic datastore changes and rollbacks tested with every deployment
Test & Verification	• Automated unit tests • Separate test environment	• Automatic Integration Tests • Static code analysis • Test coverage analysis	• Automatic functional tests • Manual performance/ security tests	• Fully automatic acceptance tests • Automatic performance/security tests • Manual exploratory testing based on risk based testing analysis	• Verify expected business value • Defects found and fixed immediately (roll forward)
Information & Reporting	• Baseline process metrics • Manual reporting • Visible to report runner	• Measure the process • Automatic reporting • Visible to team	• Automatic generation of release notes • Pipeline traceability • Reporting history • Visible to cross-silo	• Report trend analysis • Real time graphs on deployment pipeline metrics	• Dynamic self-service of information • Customizable dashboards • Cross-reference across organizational boundaries

C

Devops
:scrum, daily standup, planning/restrospect

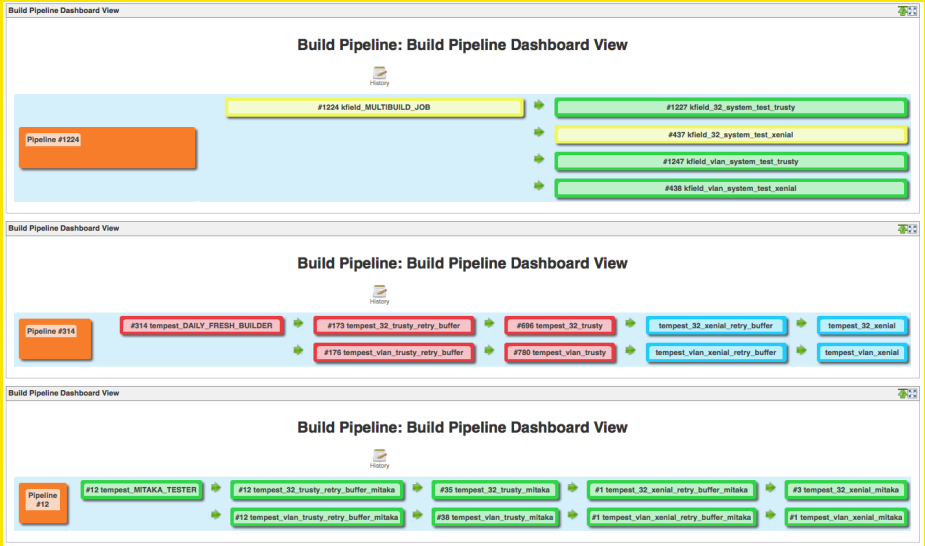
BTW,
What is Devops?

Where are you from CMMI-DEV perspective? kakao case

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
Culture & Organization	• Teams organized based on platform/ technology • Defined and documented processes	• One backlog per team • Adopt agile methodologies • Remove team boundaries	• Extended team collaboration • Remove boundary dev/ ops • Common process for all changes	• Cross-team continuous improvement • Teams responsible all the way to production	• Cross functional teams
Build & Deploy	• Centralized version control • Automated build scripts • No management of artifacts • Manual deployment • Environments are manually provisioned	• Polling CI builds • Any build can be re-created from source control • Management of build artifacts • Automated deployment scripts • Automated provisioning of environments	• Commit hook CI builds • Build fails if quality is not met (code analysis, performance, etc.) • Push button deployment and release of any releasable artifact to any environment • Standard deployment process for all environments	• C • iorities keeping codebase deployable over doing new work • Builds are not left broken • Orchestrated deployments • Blue Green Deployments	• Zero touch Continuous Deployments
Release	• Infrequent and unreliable releases • Manual process	• Painful infrequent but reliable releases	• Infrequent but fully automated and reliable releases in any environment	• Frequent fully automated releases • Deployment disconnected from release • Canary releases	• No rollbacks, always roll forward
Data Management	• Data migrations are performed manually, no scripts	• Data migrations using versioned scripts, performed manually	• Automated and versioned changes to datastores	• Changes to datastores automatically performed as part of the deployment process	• Automatic datastore changes and rollbacks tested with every deployment
Test & Verification	• Automated unit tests • Separate test environment	• Automatic Integration Tests • Static code analysis • Test coverage analysis	• Automatic functional tests • Manual performance/ security tests	• Fully automatic acceptance tests • Automatic performance/security tests • Manual exploratory testing based on risk based testing analysis	• Verify expected business value • Defects found and fixed immediately (roll forward)
Information & Reporting	• Baseline process metrics • Manual reporting • Visible to report runner	• Measure the process • Automatic reporting • Visible to team	• Automatic generation of release notes • Pipeline traceability • Reporting history • Visible to cross-silo	• Report trend analysis • Real time graphs on deployment pipeline metrics	• Dynamic self-service of information • Customizable dashboards • Cross-reference across organizational boundaries

-Triggered Test
-Daily Test

Targeting zero touching deployment



Where are you from CMMI-DEV perspective? kakao case

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
Culture & Organization	• Teams organized based on platform/ technology • Defined and documented processes	• One backlog per team • Adopt agile methodologies • Remove team boundaries	• Extended team collaboration • Remove boundary dev/ ops • Common process for all changes	• Cross-team continuous improvement • Teams responsible all the way to production	• Cross functional teams
Build & Deploy	• Centralized version control • Automated build scripts • No management of artifacts • Manual deployment • Environments are manually provisioned	• Polling CI builds • Any build can be re-created from source control • Management of build artifacts • Automated deployment scripts • Automated provisioning of environments	• Commit hook CI builds • Build fails if quality is not met (code analysis, performance, etc.) • Push button deployment and release of any releasable artifact to any environment • Standard deployment process for all environments	• Team priorities keeping codebase deployable over doing new work • Builds are not left broken • Orchestrated deployments • Blue Green Deployments	• Zero touch Continuous Deployments
Release	• Infrequent and unreliable releases • Manual process	• Painful infrequent but reliable releases	• Infrequent but fully automated and releases in any environment	• Frequent fully automated releases • Deployment disconnected from release • Canary releases	• No rollbacks, always roll forward
Data Management	• Data migrations are performed manually, no scripts	• Data migrations using versioned scripts, performed manually	• Automated and versioned changes to datastores	• Changes to datastores automatically performed as part of the deployment process	• Automatic datastore changes and rollbacks tested with every deployment
Test & Verification	• Automated unit tests • Separate test environment	• Automatic Integration Tests • Static code analysis • Test coverage analysis	• Automatic functional tests • Manual performance/ security tests	• Fully automatic acceptance tests • Automatic performance/security tests • Manual exploratory testing based on risk based testing analysis	• Verify expected business value • Defects found and fixed immediately (roll forward)
Information & Reporting	• Baseline process metrics • Manual reporting • Visible to report runner	• Measure the process • Automatic reporting • Visible to team	• Automatic generation of release notes • Pipeline traceability • Reporting history • Visible to cross-silo	• Report trend analysis • Real time graphs on deployment pipeline metrics	• Dynamic self-service of information • Customizable dashboards • Cross-reference across organizational boundaries

C

After huge neutron network break down, Regular basis Release

Targeting to Frequent releases

Where are you from CMMI-DEV perspective? kakao case

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
Culture & Organization	• Teams organized based on platform/ technology • Defined and documented processes	• One backlog per team • Adopt agile methodologies • Remove team boundaries	• Extended team collaboration • Remove boundary dev/ ops • Common process for all changes	• Cross-team continuous improvement • Teams responsible all the way to production	• Cross functional teams
Build & Deploy	• Centralized version control • Automated build scripts • No management of artifacts • Manual deployment • Environments are manually provisioned	• Polling CI builds • Any build can be re-created from source control • Management of build artifacts • Automated deployment scripts • Automated provisioning of environments	• Commit hook CI builds • Build fails if quality is not met (code analysis, performance, etc.) • Push button deployment and release of any releasable artifact to any environment • Standard deployment process for all environments	• Team priorities keeping codebase deployable over doing new work • Builds are not left broken • Orchestrated deployments • Blue Green Deployments	• Zero touch Continuous Deployments
Release	• Infrequent and unreliable releases • Manual process	• Painful infrequent but reliable releases	• Infrequent but fully automated and reliable releases in any environment	• Frequent fully automated releases • Deployment disconnected from release • Canary releases	• No rollbacks, always roll forward
Data Management	• Data migrations are performed manually, no scripts	• Data migrations using versioned scripts, performed manually	• Automated and versioned changes to datastores	• C to datastores cally performed as part ployment process	• Automatic datastore changes and rollbacks tested with every deployment
Test & Verification	• Automated unit tests • Separate test environment	• Automatic Integration Tests • Static code analysis • Test coverage analysis	• Automatic functior tests • Manual performan security tests	• Fully automatic acceptance tests • Automatic performance/security tests • Manual exploratory testing based on risk based testing analysis	• Verify expected business value • Defects found and fixed immediately (roll forward)
Information & Reporting	• Baseline process metrics • Manual reporting • Visible to report runner	• Measure the process • Automatic reporting • Visible to team	• Automatic generation of release notes • Pipeline traceabilit • Reporting history • Visible to cross-silo	• Report trend analysis • Real time graphs on deployment pipeline metrics	• Dynamic self-service of information • Customizable dashboards • Cross-reference across organizational boundaries

C

C

C

Jenkins manages Version
Configuration Server Stores the data

Type: [STATUS]
description: Build finished. 🌞🌞🌞🌞🌞🌞🌞🌞🌞🌞
github: https://github.daumkakao.com/kemi/crow-eventhandler/commit/72b9e08b4c489dea4bb7903cf23e9c6d23fff3a5
jenkins_link: http://kemi-jenkins.s1.krane.9rum.cc:8080/job/crow-eventhandler/46/
message: remove elastic search

What is the sole purpose of doing cloud?

CMM0	CMM1
legacy	self service Dev resource
output: ITF	output: krane (openstack cloud)

Some Numbers about kakao openstack

8703

vms

15,xxx

vms

1563

projects

2,xxx

projects

9x,xxx active cores

632

pull request since 2014.9

807

pull request since 2014.9

88 about

VMs are created/deleted per day

2016.8

100 about

VMs are created/deleted per day

2017.7

Some information about kakao openstack

Kakaocorp

from grizzly to **Mitaka**

7 times upgraded

total **4Region**

Heat/

Trove/

Octavia/

barbican

2017.7

from grizzly to **Kilo**

5 times upgraded

total **4Region**

additional service **Heat/**

Trove/

Sahara

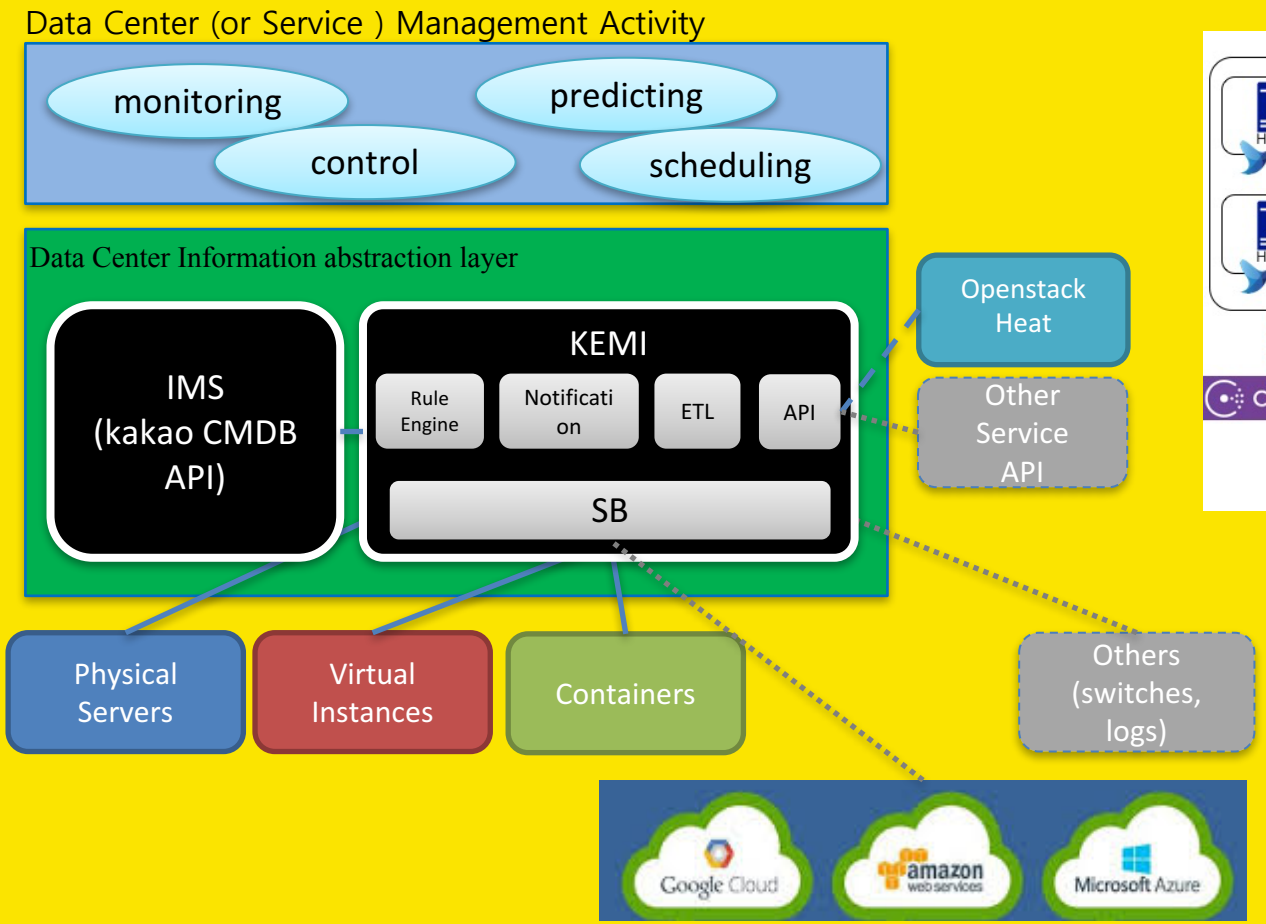
2016.8

What is the sole purpose of doing cloud?

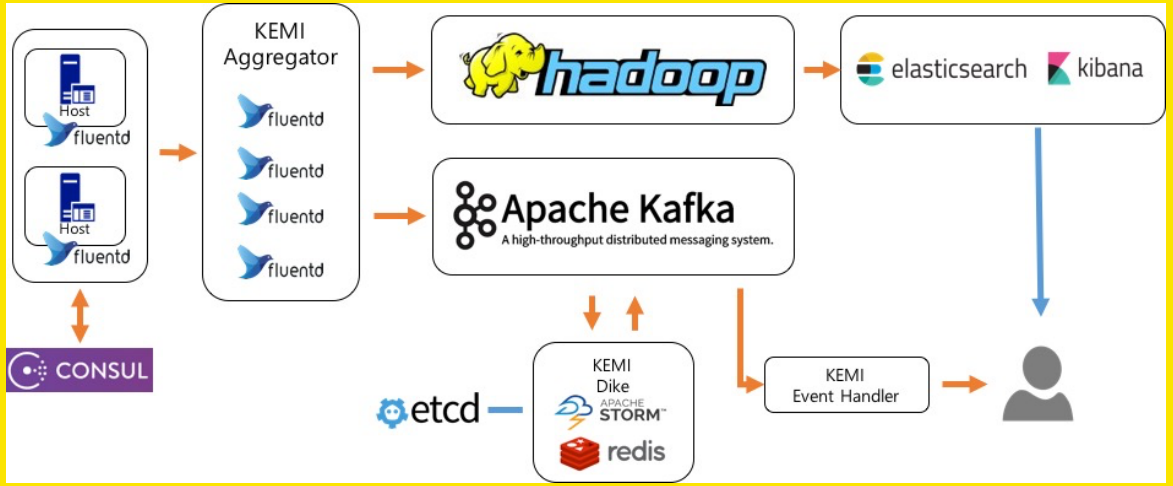
CMM0	CMM1	CMM2
legacy	self service Dev resource	limited Prod resources
output: cloudTF	output: krane (openstack cloud)	output: kemi (MaaS)

CMMI 2 Case:
event monitoring/alert platform kakao, KEMI

KEMI stats



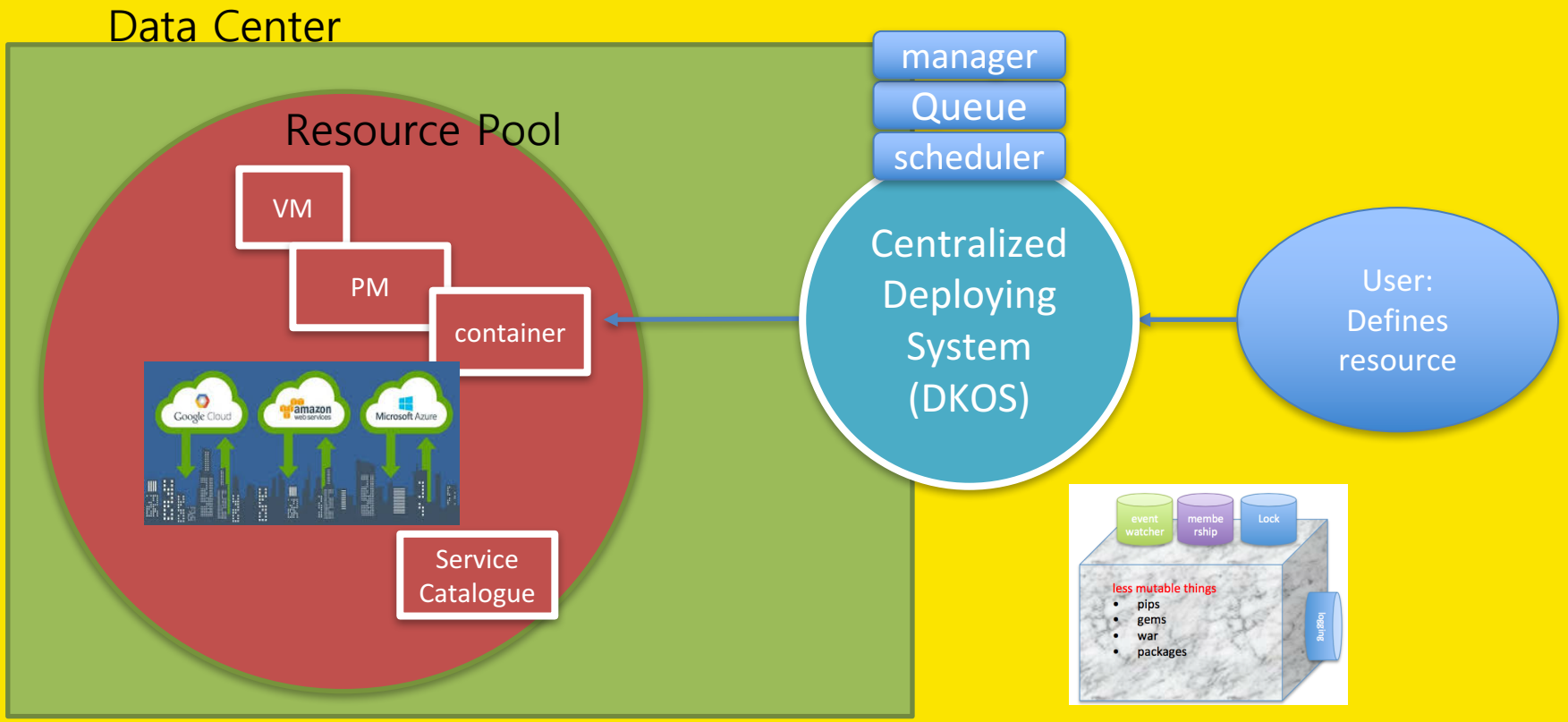
KEMI log



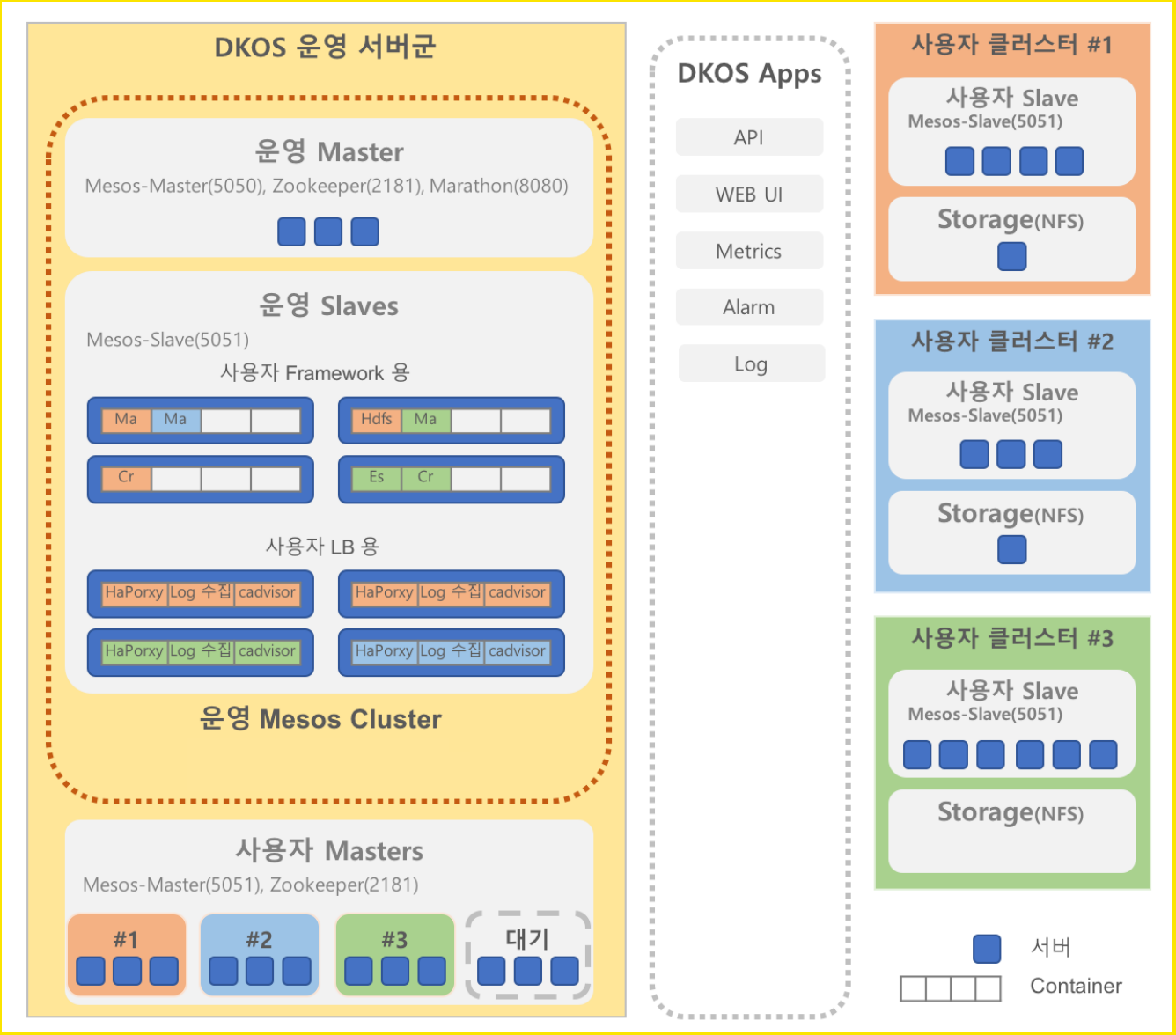
What is the sole purpose of doing cloud?

CMM0	CMM1	CMM2	CMM3
legacy	self service Dev resource	limited Prod resources	Automated CloudUsage
output: cloudTF	output: krane (openstack cloud)	output: kemi (MaaS)	output: DKOS (CaaS)

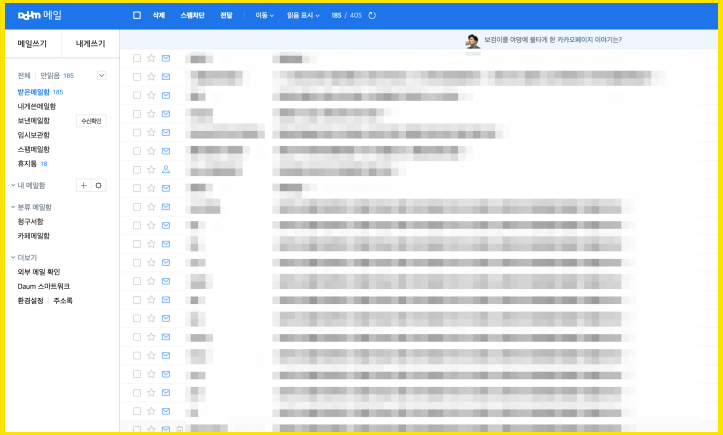
CMM3 case:
Deployment abstraction in Kakao, DKOS

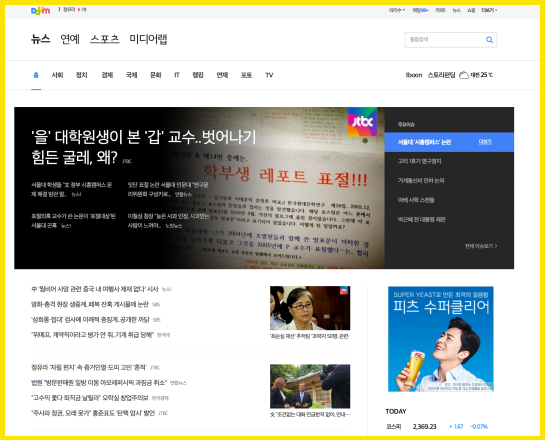


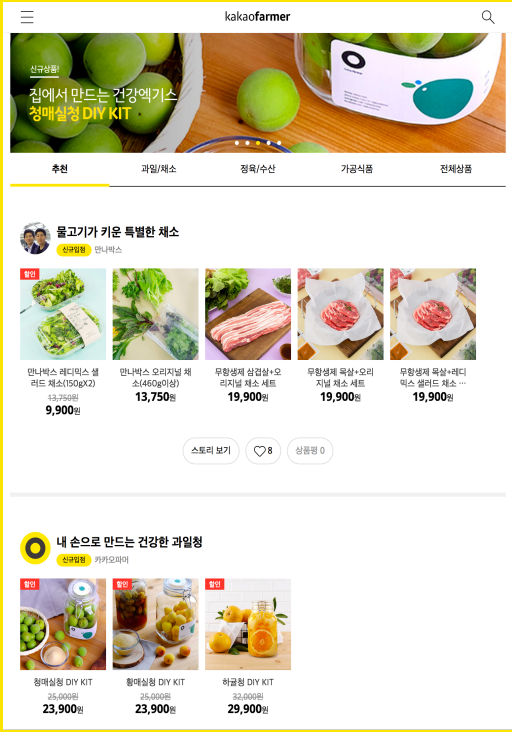
DKOS Archtecture



Services over DKOS







DKOS Situation

- Active cluster : 400
- Total compute node : 4000 (vm+pm)
- Container counts : 8800
- Managed by?

DKOS Situation

- Why use DKOS(container)?
  - Container is easy
  - Container is cool
  - dc/os is great

- Nop!
  - Very summit point of integrated/automated infra service api
    - authentication, authorization, compute resources, network, volumes
    - Monitoring, Notifications
- More On Todays A-3 2nd Session “카카오 DKOS”

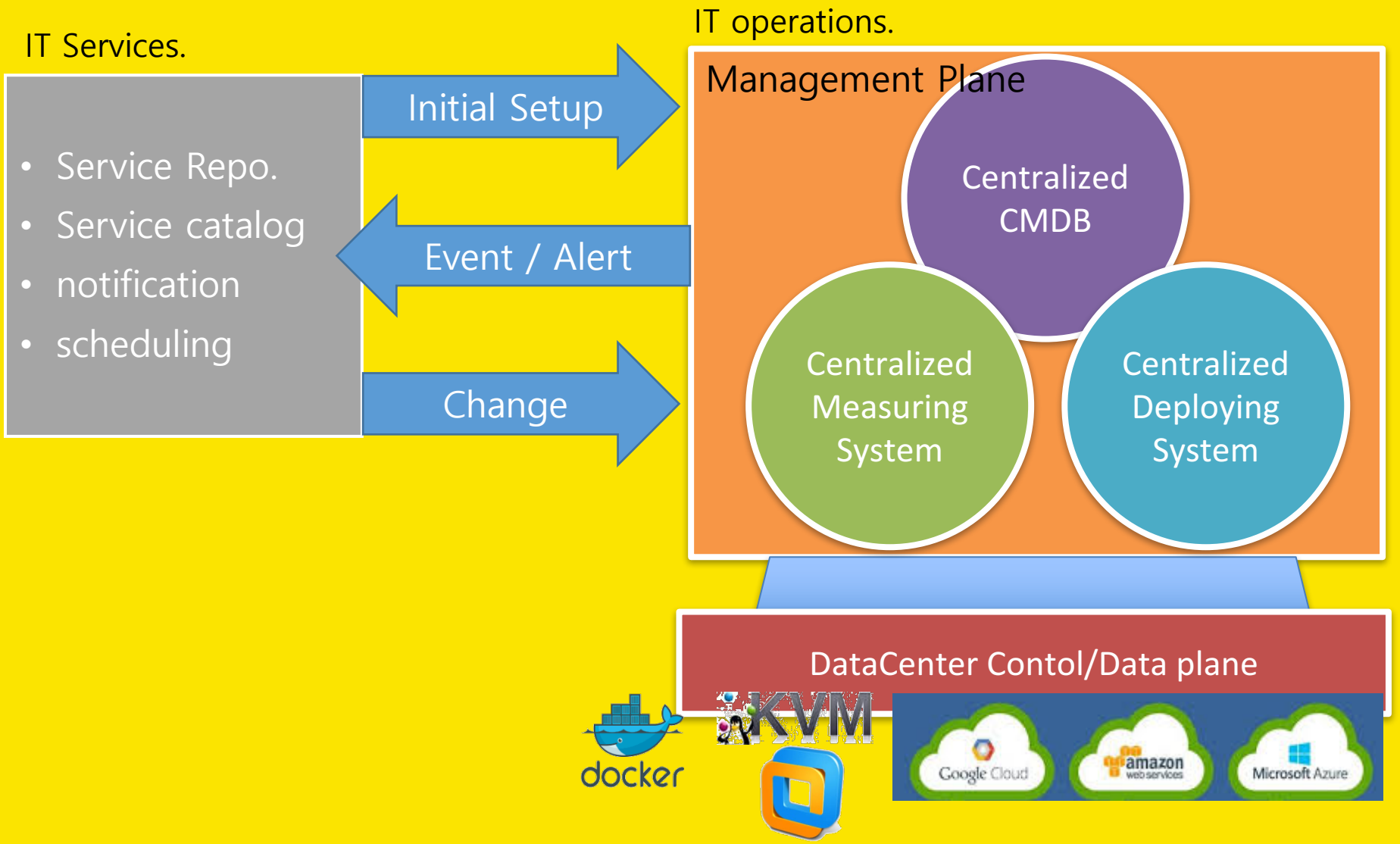
Where are you from CMMI-Cloud perspective?

For CMM4, Time to embrace Clouds, not a Cloud

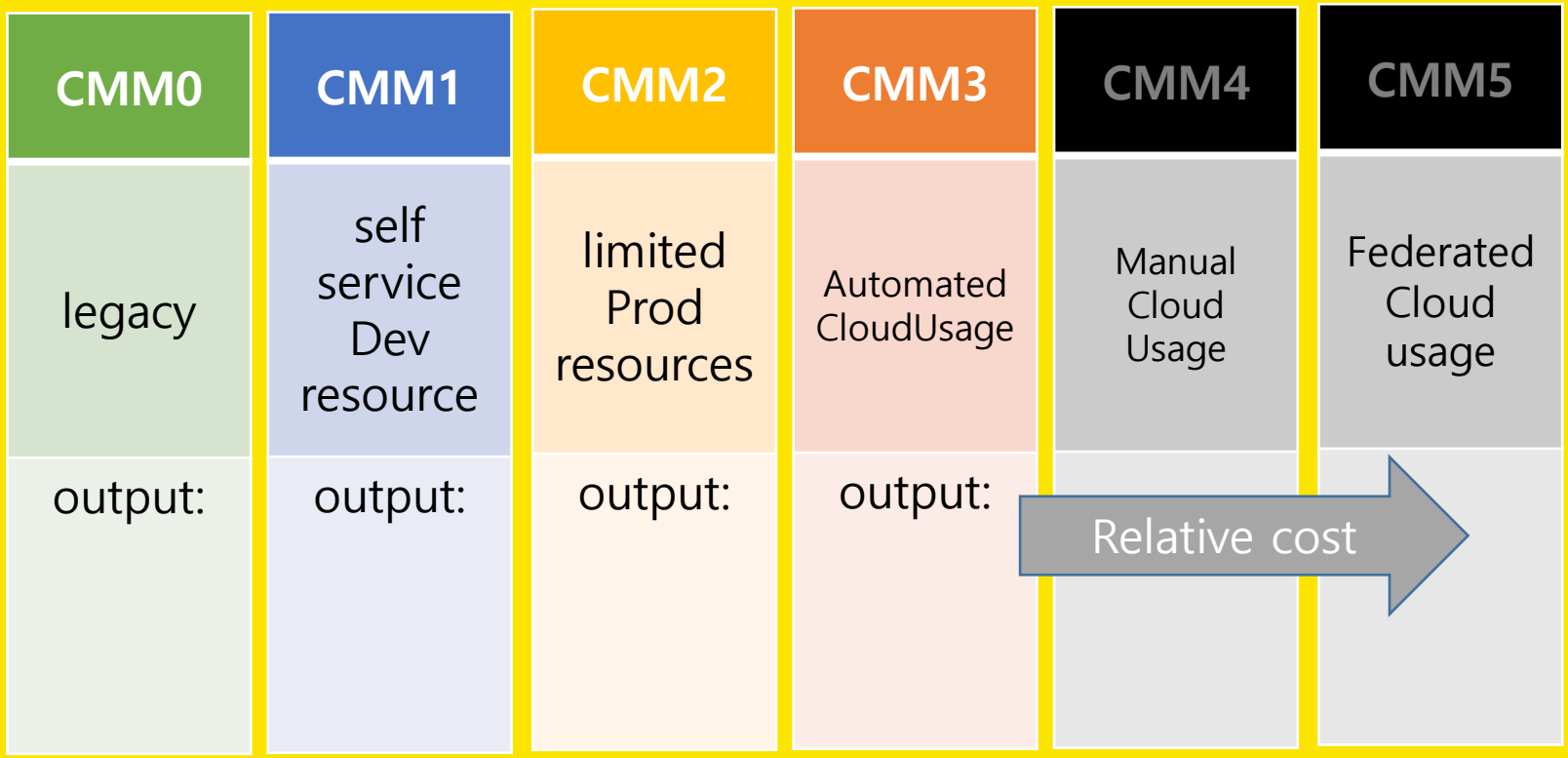
CMM0	CMM1	CMM2	CMM3	CMM4	CMM5
legacy	self service Dev resource	limited Prod resources	Automated CloudUsage	Manual Cloud Usage	Federated Cloud usage
output: cloudTF	output: krane (openstack cloud)	output: kemi (MaaS)	output: DKOS (CaaS)	--	--

Abstraction should come first

Fully abstracted operation



What drives you from CMM3 to CMM4/5?



You should say

Thanks