

# Embracing Clouds

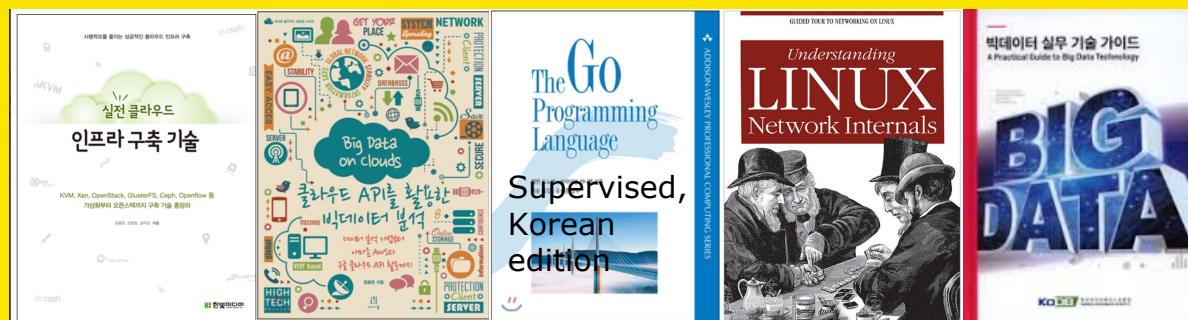
**Andrew yongjoon kong**  
CloudComputingCell, kakao  
[andrew.kong@kakaocorp.com](mailto:andrew.kong@kakaocorp.com)

## 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, Vancouver)



2<sup>nd</sup> Editions are coming...

**Everyone says**

---

Openstack is done  
K8S is not much

Cloud is Done!

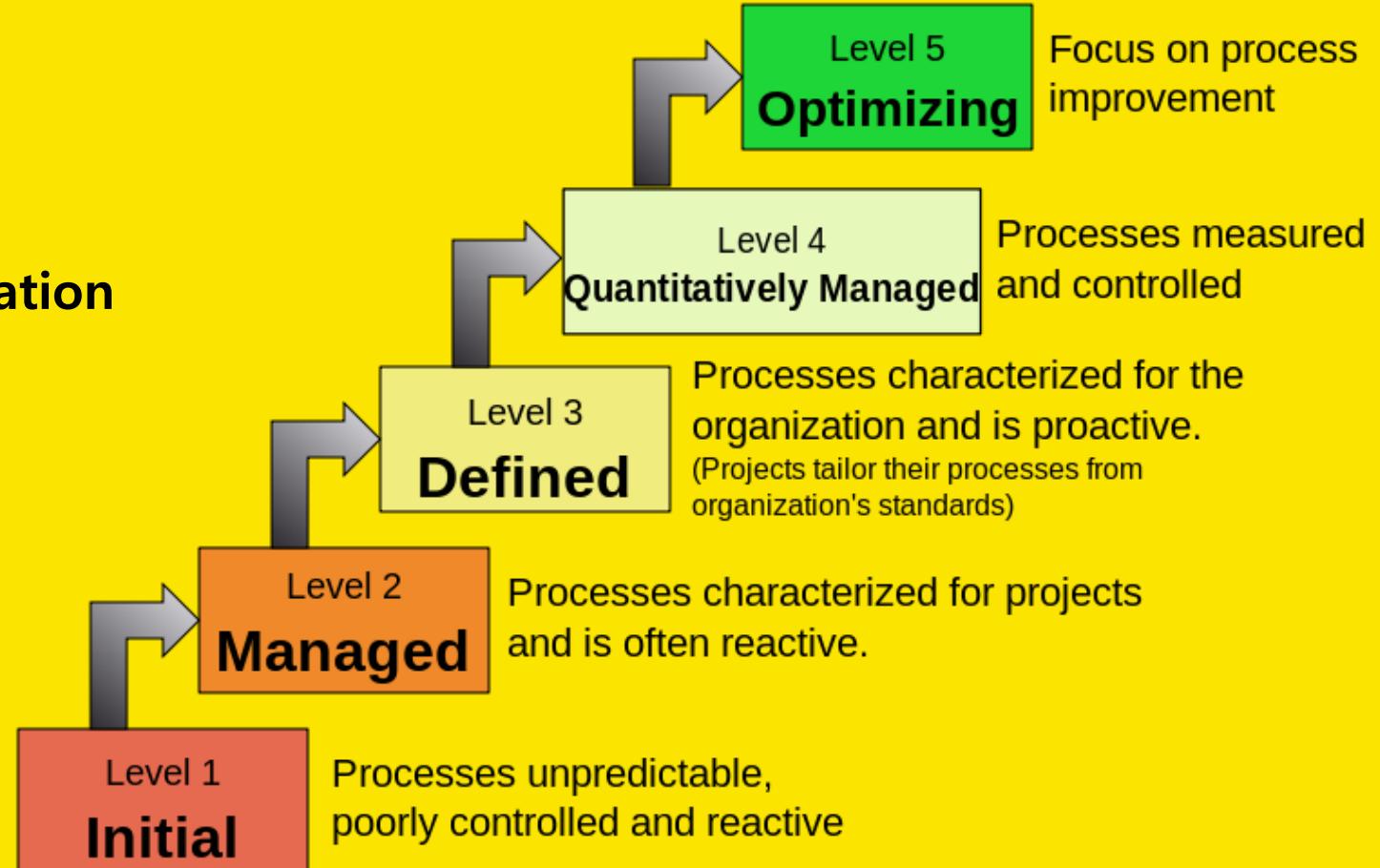


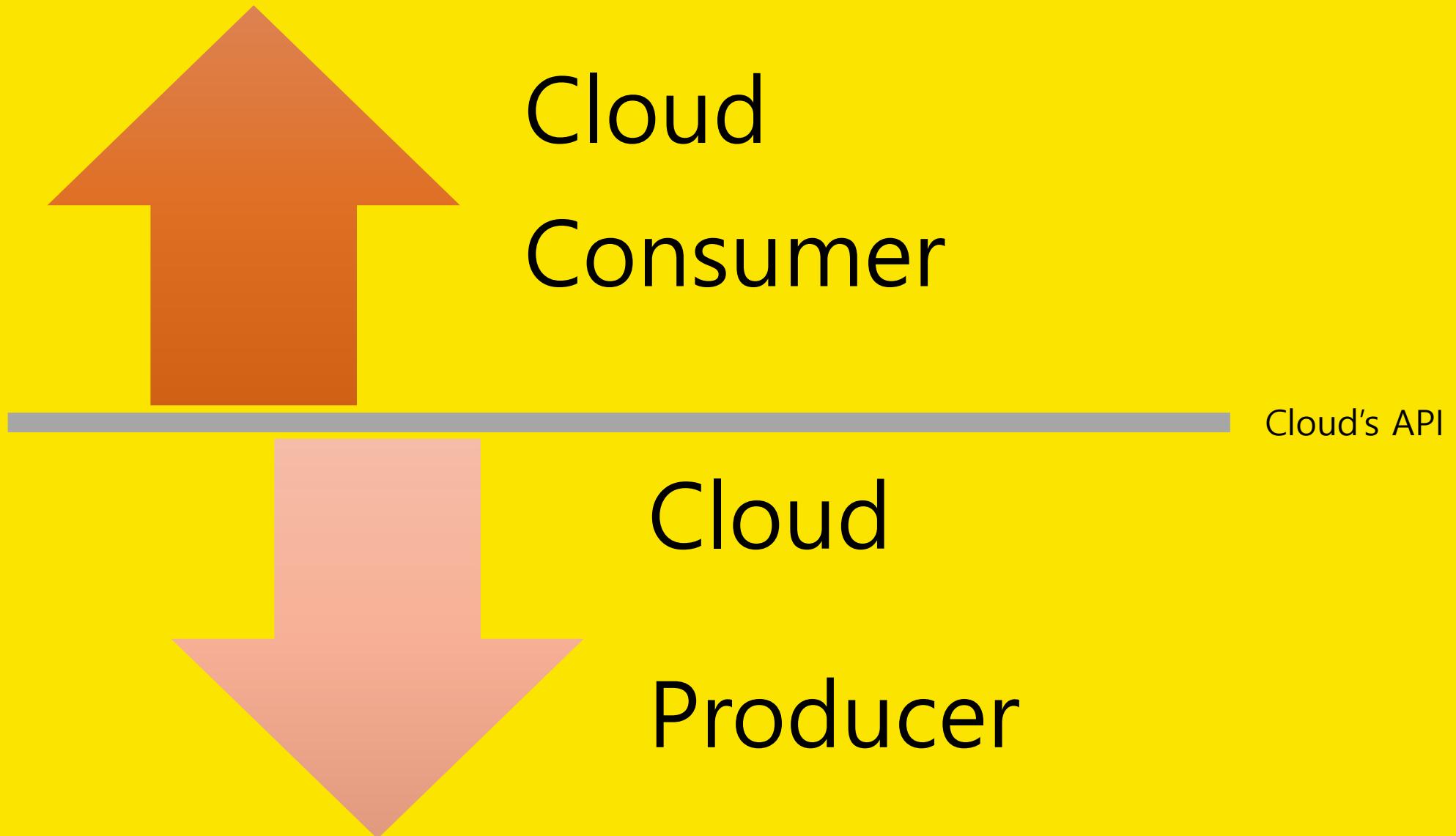
How Far did you go with your cloud?

---

## CMMI Model

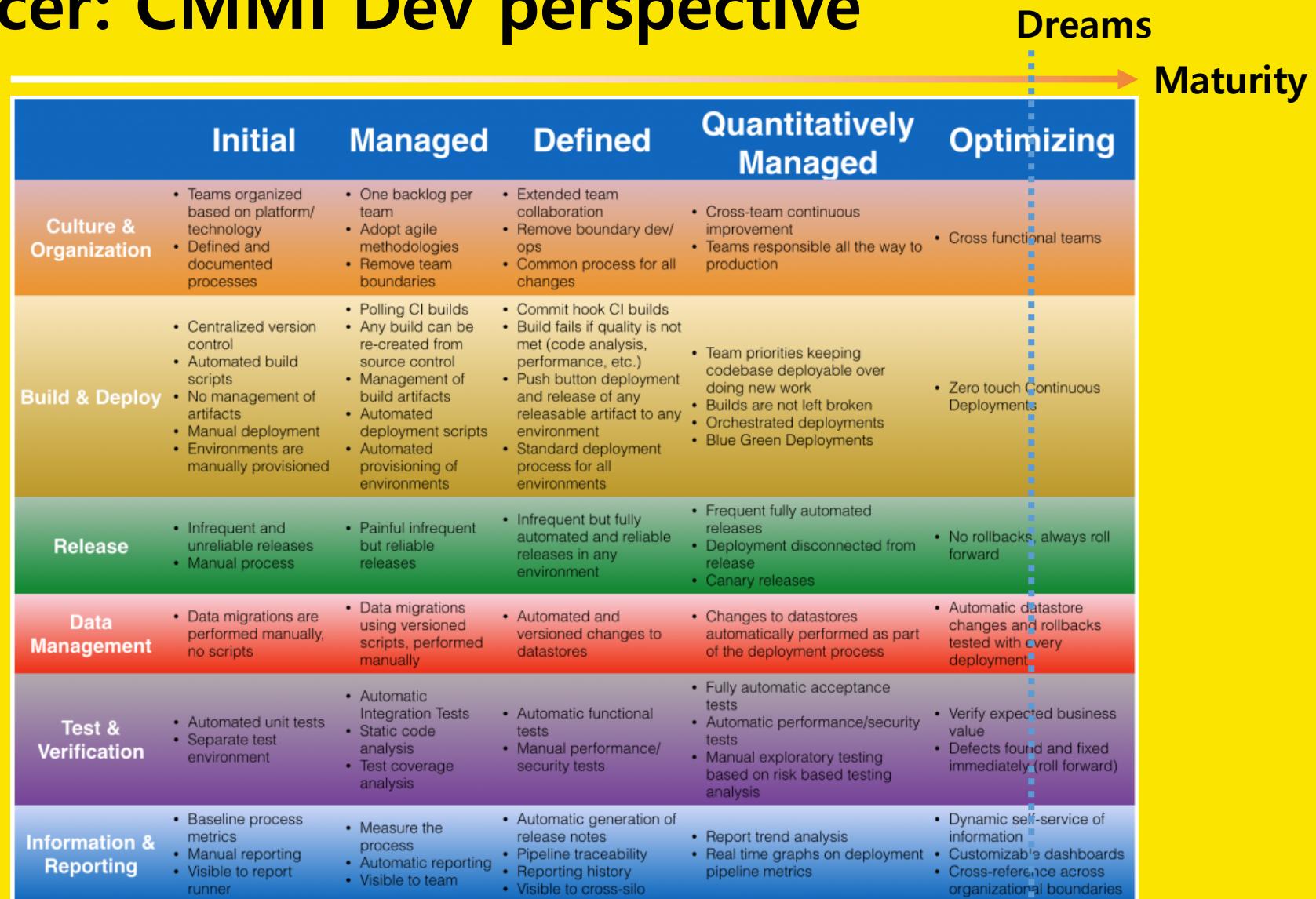
Capability Maturity Model Integration  
developed by CMU





How Far did you go with your cloud? like Kakao

# Cloud Producer: CMMI Dev perspective



The diagram illustrates the CMMI Dev perspective matrix across five maturity levels: Initial, Managed, Defined, Quantitatively Managed, and Optimizing. The matrix is organized by process area (Culture & Organization, Build & Deploy, Release, Data Management, Test & Verification, Information & Reporting) and row.

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
<b>Culture &amp; Organization</b>	<ul style="list-style-type: none"> <li>Teams organized based on platform/technology</li> <li>Defined and documented processes</li> </ul>	<ul style="list-style-type: none"> <li>One backlog per team</li> <li>Adopt agile methodologies</li> <li>Remove team boundaries</li> </ul>	<ul style="list-style-type: none"> <li>Extended team collaboration</li> <li>Remove boundary dev/ops</li> <li>Common process for all changes</li> </ul>	<ul style="list-style-type: none"> <li>Cross-team continuous improvement</li> <li>Teams responsible all the way to production</li> </ul>	<ul style="list-style-type: none"> <li>Cross functional teams</li> </ul>
<b>Build &amp; Deploy</b>	<ul style="list-style-type: none"> <li>Centralized version control</li> <li>Automated build scripts</li> <li>No management of artifacts</li> <li>Manual deployment</li> <li>Environments are manually provisioned</li> </ul>	<ul style="list-style-type: none"> <li>Polling CI builds</li> <li>Any build can be re-created from source control</li> <li>Management of build artifacts</li> <li>Automated deployment scripts</li> <li>Automated provisioning of environments</li> </ul>	<ul style="list-style-type: none"> <li>Commit hook CI builds</li> <li>Build fails if quality is not met (code analysis, performance, etc.)</li> <li>Push button deployment and release of any releasable artifact to any environment</li> <li>Standard deployment process for all environments</li> </ul>	<ul style="list-style-type: none"> <li>Team priorities keeping codebase deployable over doing new work</li> <li>Builds are not left broken</li> <li>Orchestrated deployments</li> <li>Blue Green Deployments</li> </ul>	<ul style="list-style-type: none"> <li>Zero touch Continuous Deployments</li> </ul>
<b>Release</b>	<ul style="list-style-type: none"> <li>Infrequent and unreliable releases</li> <li>Manual process</li> </ul>	<ul style="list-style-type: none"> <li>Painful infrequent but reliable releases</li> </ul>	<ul style="list-style-type: none"> <li>Infrequent but fully automated and reliable releases in any environment</li> </ul>	<ul style="list-style-type: none"> <li>Frequent fully automated releases</li> <li>Deployment disconnected from release</li> <li>Canary releases</li> </ul>	<ul style="list-style-type: none"> <li>No rollbacks, always roll forward</li> </ul>
<b>Data Management</b>	<ul style="list-style-type: none"> <li>Data migrations are performed manually, no scripts</li> </ul>	<ul style="list-style-type: none"> <li>Data migrations using versioned scripts, performed manually</li> </ul>	<ul style="list-style-type: none"> <li>Automated and versioned changes to datastores</li> </ul>	<ul style="list-style-type: none"> <li>Changes to datastores automatically performed as part of the deployment process</li> </ul>	<ul style="list-style-type: none"> <li>Automatic datastore changes and rollbacks tested with every deployment</li> </ul>
<b>Test &amp; Verification</b>	<ul style="list-style-type: none"> <li>Automated unit tests</li> <li>Separate test environment</li> </ul>	<ul style="list-style-type: none"> <li>Automatic Integration Tests</li> <li>Static code analysis</li> <li>Test coverage analysis</li> </ul>	<ul style="list-style-type: none"> <li>Automatic functional tests</li> <li>Manual performance/security tests</li> </ul>	<ul style="list-style-type: none"> <li>Fully automatic acceptance tests</li> <li>Automatic performance/security tests</li> <li>Manual exploratory testing based on risk based testing analysis</li> </ul>	<ul style="list-style-type: none"> <li>Verify expected business value</li> <li>Defects found and fixed immediately (roll forward)</li> </ul>
<b>Information &amp; Reporting</b>	<ul style="list-style-type: none"> <li>Baseline process metrics</li> <li>Manual reporting</li> <li>Visible to report runner</li> </ul>	<ul style="list-style-type: none"> <li>Measure the process</li> <li>Automatic reporting</li> <li>Visible to team</li> </ul>	<ul style="list-style-type: none"> <li>Automatic generation of release notes</li> <li>Pipeline traceability</li> <li>Reporting history</li> <li>Visible to cross-silo</li> </ul>	<ul style="list-style-type: none"> <li>Report trend analysis</li> <li>Real time graphs on deployment pipeline metrics</li> </ul>	<ul style="list-style-type: none"> <li>Dynamic self-service of information</li> <li>Customizable dashboards</li> <li>Cross-reference across organizational boundaries</li> </ul>

How Far did you go with your cloud? like Kakao

# Cloud Consumer: CMMI Cloud Perspective

CMM 0 Legacy	CMM 1 Initial, Ad hoc	CMM 2 Repeatable, Opportunistic	CMM 3 Defined, Systematic	CMM 4 Measured, Measurable	CMM 5 Optimized
<p><b>Legacy applications on dedicated infrastructure</b></p> <p>No cloud approach. No cloud elements implemented.</p>	<p><b>Analysis of current environment's cloud readiness</b></p> <p>Mapping and analysis of cloud potential for existing systems and services.</p> <p>There is some awareness of cloud computing, and some groups are beginning to implement cloud computing elements.</p>	<p><b>Processes for cloud adoption defined</b></p> <p>An approach has been decided upon and is applied opportunistically.</p> <p>The approach is not widely accepted. Redundant or overlapping approaches exist.</p> <p>Informally defined or exists as "shelfware."</p> <p>Initial benefits realized from leveraged infrastructure.</p>	<p><b>Tooling and integration for automated cloud usage</b></p> <p>Affected parties have reviewed and accepted the approach.</p> <p>The documented approach is always or nearly always followed.</p>	<p><b>Manual Federation</b></p> <p>Cloud-aware applications are deployed according to business requirements on public, private, and hybrid platforms.</p> <p>Governance infrastructure is in place that measures and quantitatively manages cloud capability.</p>	<p><b>Federated, interoperable, and open cloud</b></p> <p>Capability incrementally improves based on consistently gathered metrics.</p> <p>Assets are proactively maintained to ensure relevance and correctness.</p> <p>The organization has established the potential to use market mechanisms to leverage inter-cloud</p>

**clouds** →

**ANALYSIS**    **CAPABILITY GAINS**    **EFFICIENCY GAINS**    **INCREASES IN VELOCITY AND QUALITY**    **PROACTIVE**

Capability, efficiency, velocity, and quality continually increase as higher levels of implementation are achieved

# Where are you from CMMI-DEV perspective? kakao case

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
Culture & Organization	<ul style="list-style-type: none"> <li>Teams organized based on platform/technology</li> <li>Defined and documented processes</li> </ul>	<ul style="list-style-type: none"> <li>One backlog per team</li> <li>Adopt agile methodologies</li> <li>Remove team boundaries</li> </ul>	<ul style="list-style-type: none"> <li>Extended team collaboration</li> <li>Remove boundary ops</li> <li>Common process for all changes</li> </ul>	<ul style="list-style-type: none"> <li>Cross-team continuous improvement</li> <li>Teams responsible all the way to production</li> </ul>	<ul style="list-style-type: none"> <li>Cross functional teams</li> </ul>
Build & Deploy	<ul style="list-style-type: none"> <li>Centralized version control</li> <li>Automated build scripts</li> <li>No management of artifacts</li> <li>Manual deployment</li> <li>Environments are manually provisioned</li> </ul>	<ul style="list-style-type: none"> <li>Any build can be re-created from source control</li> <li>Management of build artifacts</li> <li>Automated deployment scripts</li> <li>Automated provisioning of environments</li> </ul>	<ul style="list-style-type: none"> <li>Commit hook CI builds</li> <li>Build fails if quality is not met (code analysis, performance, etc.)</li> <li>Push button deployment and release of any releasable artifact to any environment</li> <li>Standard deployment process for all environments</li> </ul>	<ul style="list-style-type: none"> <li>Team priorities keeping codebase deployable over doing new work</li> <li>Builds are not left broken</li> <li>Orchestrated deployments</li> <li>Blue Green Deployments</li> </ul>	<ul style="list-style-type: none"> <li>Zero touch Continuous Deployments</li> </ul>
Release	<ul style="list-style-type: none"> <li>Infrequent and unreliable releases</li> <li>Manual process</li> </ul>	<ul style="list-style-type: none"> <li>Painful infrequent but reliable releases</li> </ul>	<ul style="list-style-type: none"> <li>Infrequent but fully automated and reliable releases in any environment</li> </ul>	<ul style="list-style-type: none"> <li>Frequent fully automated releases</li> <li>Deployment disconnected from release</li> <li>Canary releases</li> </ul>	<ul style="list-style-type: none"> <li>No rollbacks, always roll forward</li> </ul>
Data Management	<ul style="list-style-type: none"> <li>Data migrations are performed manually, no scripts</li> </ul>	<ul style="list-style-type: none"> <li>Data migrations using versioned scripts, performed manually</li> </ul>	<ul style="list-style-type: none"> <li>Automated and versioned changes to datastores</li> </ul>	<ul style="list-style-type: none"> <li>Changes to datastores automatically performed as part of the deployment process</li> </ul>	<ul style="list-style-type: none"> <li>Automatic datastore changes and rollbacks tested with every deployment</li> </ul>
Test & Verification	<ul style="list-style-type: none"> <li>Automated unit tests</li> <li>Separate test environment</li> </ul>	<ul style="list-style-type: none"> <li>Automatic Integration Tests</li> <li>Static code analysis</li> <li>Test coverage analysis</li> </ul>	<ul style="list-style-type: none"> <li>Automatic functional tests</li> <li>Manual performance/security tests</li> </ul>	<ul style="list-style-type: none"> <li>Fully automatic acceptance tests</li> <li>Automatic performance/security tests</li> <li>Manual exploratory testing based on risk based testing analysis</li> </ul>	<ul style="list-style-type: none"> <li>Verify expected business value</li> <li>Defects found and fixed immediately (roll forward)</li> </ul>
Information & Reporting	<ul style="list-style-type: none"> <li>Baseline process metrics</li> <li>Manual reporting</li> <li>Visible to report runner</li> </ul>	<ul style="list-style-type: none"> <li>Measure the process</li> <li>Automatic reporting</li> <li>Visible to team</li> </ul>	<ul style="list-style-type: none"> <li>Automatic generation of release notes</li> <li>Pipeline traceability</li> <li>Reporting history</li> <li>Visible to cross-silo</li> </ul>	<ul style="list-style-type: none"> <li>Report trend analysis</li> <li>Real time graphs on deployment pipeline metrics</li> </ul>	<ul style="list-style-type: none"> <li>Dynamic self-service of information</li> <li>Customizable dashboards</li> <li>Cross-reference across organizational boundaries</li> </ul>

C

→

**Devops**  
:scrum, daily standup, planning/restrospect

**BTW,**  
**What is Devops?**

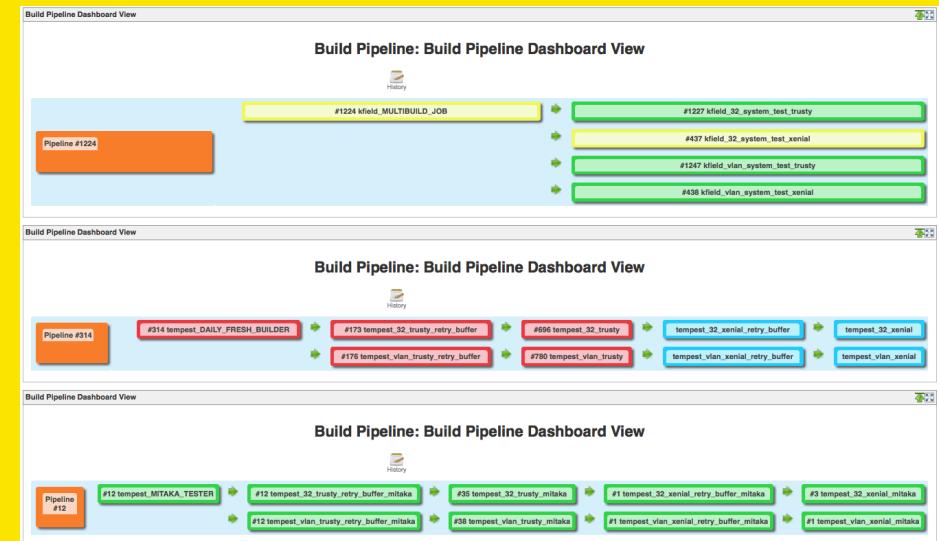
# Where are you from CMMI-DEV perspective? kakao case

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
<b>Culture &amp; Organization</b>	<ul style="list-style-type: none"> <li>Teams organized based on platform/technology</li> <li>Defined and documented processes</li> </ul>	<ul style="list-style-type: none"> <li>One backlog per team</li> <li>Adopt agile methodologies</li> <li>Remove team boundaries</li> </ul>	<ul style="list-style-type: none"> <li>Extended team collaboration</li> <li>Remove boundary dev/ops</li> <li>Common process for all changes</li> </ul>	<ul style="list-style-type: none"> <li>Cross-team continuous improvement</li> <li>Teams responsible all the way to production</li> </ul>	Cross functional teams
<b>Build &amp; Deploy</b>	<ul style="list-style-type: none"> <li>Centralized version control</li> <li>Automated build scripts</li> <li>No management of artifacts</li> <li>Manual deployment</li> <li>Environments are manually provisioned</li> </ul>	<ul style="list-style-type: none"> <li>Polling CI builds</li> <li>Any build can be re-created from source control</li> <li>Management of build artifacts</li> <li>Automated deployment scripts</li> <li>Automated provisioning of environments</li> </ul>	<ul style="list-style-type: none"> <li>Commit hook CI builds</li> <li>Build fails if quality is not met (code analysis, performance, etc.)</li> <li>Push button deployment and release of any releasable artifact to any environment</li> <li>Standard deployment process for all environments</li> </ul>	<p><b>C</b> Priorities keeping codebase deployable over doing new work</p> <ul style="list-style-type: none"> <li>Builds are not left broken</li> <li>Orchestrated deployments</li> <li>Blue Green Deployments</li> </ul>	<ul style="list-style-type: none"> <li>Zero touch Continuous Deployments</li> </ul>
<b>Release</b>	<ul style="list-style-type: none"> <li>Infrequent and unreliable releases</li> <li>Manual process</li> </ul>	<ul style="list-style-type: none"> <li>Painful infrequent but reliable releases</li> </ul>	<ul style="list-style-type: none"> <li>Infrequent but fully automated and reliable releases in any environment</li> </ul>	<ul style="list-style-type: none"> <li>Frequent fully automated releases</li> <li>Deployment disconnected from release</li> <li>Canary releases</li> </ul>	<ul style="list-style-type: none"> <li>No rollbacks, always roll forward</li> </ul>
<b>Data Management</b>	<ul style="list-style-type: none"> <li>Data migrations are performed manually, no scripts</li> </ul>	<ul style="list-style-type: none"> <li>Data migrations using versioned scripts, performed manually</li> </ul>	<ul style="list-style-type: none"> <li>Automated and versioned changes to datastores</li> </ul>	<ul style="list-style-type: none"> <li>Changes to datastores automatically performed as part of the deployment process</li> </ul>	<ul style="list-style-type: none"> <li>Automatic datastore changes and rollbacks tested with every deployment</li> </ul>
<b>Test &amp; Verification</b>	<ul style="list-style-type: none"> <li>Automated unit tests</li> <li>Separate test environment</li> </ul>	<ul style="list-style-type: none"> <li>Automatic Integration Tests</li> <li>Static code analysis</li> <li>Test coverage analysis</li> </ul>	<ul style="list-style-type: none"> <li>Automatic functional tests</li> <li>Manual performance/security tests</li> <li>Manual exploratory testing based on risk based testing analysis</li> </ul>	<ul style="list-style-type: none"> <li>Fully automatic acceptance tests</li> <li>Automatic performance/security tests</li> <li>Manual exploratory testing based on risk based testing analysis</li> </ul>	<ul style="list-style-type: none"> <li>Verify expected business value</li> <li>Defects found and fixed immediately (roll forward)</li> </ul>
<b>Information &amp; Reporting</b>	<ul style="list-style-type: none"> <li>Baseline process metrics</li> <li>Manual reporting</li> <li>Visible to report runner</li> </ul>	<ul style="list-style-type: none"> <li>Measure the process</li> <li>Automatic reporting</li> <li>Visible to team</li> </ul>	<ul style="list-style-type: none"> <li>Automatic generation of release notes</li> <li>Pipeline traceability</li> <li>Reporting history</li> <li>Visible to cross-silo</li> </ul>	<ul style="list-style-type: none"> <li>Report trend analysis</li> <li>Real time graphs on deployment pipeline metrics</li> </ul>	<ul style="list-style-type: none"> <li>Dynamic self-service of information</li> <li>Customizable dashboards</li> <li>Cross-reference across organizational boundaries</li> </ul>

C

-Triggered Test  
-Daily Test

Targeting zero touching deployment



# Where are you from CMMI-DEV perspective? kakao case

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
<b>Culture &amp; Organization</b>	<ul style="list-style-type: none"> <li>Teams organized based on platform/technology</li> <li>Defined and documented processes</li> </ul>	<ul style="list-style-type: none"> <li>One backlog per team</li> <li>Adopt agile methodologies</li> <li>Remove team boundaries</li> </ul>	<ul style="list-style-type: none"> <li>Extended team collaboration</li> <li>Remove boundary dev/ops</li> <li>Common process for all changes</li> </ul>	<ul style="list-style-type: none"> <li>Cross-team continuous improvement</li> <li>Teams responsible all the way to production</li> </ul>	Cross functional teams
<b>Build &amp; Deploy</b>	<ul style="list-style-type: none"> <li>Centralized version control</li> <li>Automated build scripts</li> <li>No management of artifacts</li> <li>Manual deployment</li> <li>Environments are manually provisioned</li> </ul>	<ul style="list-style-type: none"> <li>Polling CI builds</li> <li>Any build can be re-created from source control</li> <li>Management of build artifacts</li> <li>Automated deployment scripts</li> <li>Automated provisioning of environments</li> </ul>	<ul style="list-style-type: none"> <li>Commit hook CI builds</li> <li>Build fails if quality is not met (code analysis, performance, etc.)</li> <li>Push button deployment and release of any releasable artifact to any environment</li> <li>Standard deployment process for all environments</li> </ul>	<ul style="list-style-type: none"> <li>Team priorities keeping codebase deployable over doing new work</li> <li>Builds are not left broken</li> <li>Orchestrated deployments</li> <li>Blue Green Deployments</li> </ul>	<ul style="list-style-type: none"> <li>Zero touch Continuous Deployments</li> </ul>
<b>Release</b>	<ul style="list-style-type: none"> <li>Infrequent and unreliable releases</li> <li>Manual process</li> </ul>	<ul style="list-style-type: none"> <li>Painful infrequent but reliable releases</li> </ul>	<ul style="list-style-type: none"> <li>Infrequent but fully automated and releases in any environment</li> </ul>	<ul style="list-style-type: none"> <li>Frequent fully automated releases</li> <li>Deployment disconnected from release</li> <li>Canary releases</li> </ul>	<ul style="list-style-type: none"> <li>No rollbacks, always roll forward</li> </ul>
<b>Data Management</b>	<ul style="list-style-type: none"> <li>Data migrations are performed manually, no scripts</li> </ul>	<ul style="list-style-type: none"> <li>Data migrations using versioned scripts, performed manually</li> </ul>	<ul style="list-style-type: none"> <li>Automated and versioned changes to datastores</li> </ul>	<ul style="list-style-type: none"> <li>Changes to datastores automatically performed as part of the deployment process</li> </ul>	<ul style="list-style-type: none"> <li>Automatic datastore changes and rollbacks tested with every deployment</li> </ul>
<b>Test &amp; Verification</b>	<ul style="list-style-type: none"> <li>Automated unit tests</li> <li>Separate test environment</li> </ul>	<ul style="list-style-type: none"> <li>Automatic Integration Tests</li> <li>Static code analysis</li> <li>Test coverage analysis</li> </ul>	<ul style="list-style-type: none"> <li>Automatic functional tests</li> <li>Manual performance/security tests</li> </ul>	<ul style="list-style-type: none"> <li>Fully automatic acceptance tests</li> <li>Automatic performance/security tests</li> <li>Manual exploratory testing based on risk based testing analysis</li> </ul>	<ul style="list-style-type: none"> <li>Verify expected business value</li> <li>Defects found and fixed immediately (roll forward)</li> </ul>
<b>Information &amp; Reporting</b>	<ul style="list-style-type: none"> <li>Baseline process metrics</li> <li>Manual reporting</li> <li>Visible to report runner</li> </ul>	<ul style="list-style-type: none"> <li>Measure the process</li> <li>Automatic reporting</li> <li>Visible to team</li> </ul>	<ul style="list-style-type: none"> <li>Automatic generation of release notes</li> <li>Pipeline traceability</li> <li>Reporting history</li> <li>Visible to cross-silo</li> </ul>	<ul style="list-style-type: none"> <li>Report trend analysis</li> <li>Real time graphs on deployment pipeline metrics</li> </ul>	<ul style="list-style-type: none"> <li>Dynamic self-service of information</li> <li>Customizable dashboards</li> <li>Cross-reference across organizational boundaries</li> </ul>

**After huge neutron network break down,  
Regular basis Release**

**Targeting to Frequent releases**

C

# Where are you from CMMI-DEV perspective? kakao case

	Initial	Managed	Defined	Quantitatively Managed	Optimizing
Culture & Organization	<ul style="list-style-type: none"> <li>Teams organized based on platform/technology</li> <li>Defined and documented processes</li> </ul>	<ul style="list-style-type: none"> <li>One backlog per team</li> <li>Adopt agile methodologies</li> <li>Remove team boundaries</li> </ul>	<ul style="list-style-type: none"> <li>Extended team collaboration</li> <li>Remove boundary dev/ops</li> <li>Common process for all changes</li> </ul>	<ul style="list-style-type: none"> <li>Cross-team continuous improvement</li> <li>Teams responsible all the way to production</li> </ul>	<ul style="list-style-type: none"> <li>Cross functional teams</li> </ul>
Build & Deploy	<ul style="list-style-type: none"> <li>Centralized version control</li> <li>Automated build scripts</li> <li>No management of artifacts</li> <li>Manual deployment</li> <li>Environments are manually provisioned</li> </ul>	<ul style="list-style-type: none"> <li>Polling CI builds</li> <li>Any build can be re-created from source control</li> <li>Management of build artifacts</li> <li>Automated deployment scripts</li> <li>Automated provisioning of environments</li> </ul>	<ul style="list-style-type: none"> <li>Commit hook CI builds</li> <li>Build fails if quality is not met (code analysis, performance, etc.)</li> <li>Push button deployment and release of any releasable artifact to any environment</li> <li>Standard deployment process for all environments</li> </ul>	<ul style="list-style-type: none"> <li>Team priorities keeping codebase deployable over doing new work</li> <li>Builds are not left broken</li> <li>Orchestrated deployments</li> <li>Blue Green Deployments</li> </ul>	<ul style="list-style-type: none"> <li>Zero touch Continuous Deployments</li> </ul>
Release	<ul style="list-style-type: none"> <li>Infrequent and unreliable releases</li> <li>Manual process</li> </ul>	<ul style="list-style-type: none"> <li>Painful infrequent but reliable releases</li> </ul>	<ul style="list-style-type: none"> <li>Infrequent but fully automated and reliable releases in any environment</li> </ul>	<ul style="list-style-type: none"> <li>Frequent fully automated releases</li> <li>Deployment disconnected from release</li> <li>Canary releases</li> </ul>	<ul style="list-style-type: none"> <li>No rollbacks, always roll forward</li> </ul>
Data Management	<ul style="list-style-type: none"> <li>Data migrations are performed manually, no scripts</li> </ul>	<ul style="list-style-type: none"> <li>Data migrations using versioned scripts, performed manually</li> </ul>	<ul style="list-style-type: none"> <li>Automated and versioned changes to datastores</li> </ul>	<p><b>C</b></p> <ul style="list-style-type: none"> <li>Changes to datastores automatically performed as part of deployment process</li> </ul>	<ul style="list-style-type: none"> <li>Automatic datastore changes and rollbacks tested with every deployment</li> </ul>
Test & Verification	<ul style="list-style-type: none"> <li>Automated unit tests</li> <li>Separate test environment</li> </ul>	<ul style="list-style-type: none"> <li>Automatic Integration Tests</li> <li>Static code analysis</li> <li>Test coverage analysis</li> </ul>	<p><b>C</b></p> <ul style="list-style-type: none"> <li>Automatic function tests</li> <li>Manual performance security tests</li> </ul>	<ul style="list-style-type: none"> <li>Fully automatic acceptance tests</li> <li>Automatic performance/security tests</li> <li>Manual exploratory testing based on risk based testing analysis</li> </ul>	<ul style="list-style-type: none"> <li>Verify expected business value</li> <li>Defects found and fixed immediately (roll forward)</li> </ul>
Information & Reporting	<ul style="list-style-type: none"> <li>Baseline process metrics</li> <li>Manual reporting</li> <li>Visible to report runner</li> </ul>	<ul style="list-style-type: none"> <li>Measure the process</li> <li>Automatic reporting</li> <li>Visible to team</li> </ul>	<p><b>C</b></p> <ul style="list-style-type: none"> <li>Automatic generation of release notes</li> <li>Pipeline traceability</li> <li>Reporting history</li> <li>Visible to cross-silo</li> </ul>	<p>Report trend analysis Real time graphs on deployment pipeline metrics</p>	<ul style="list-style-type: none"> <li>Dynamic self-service of information</li> <li>Customizable dashboards</li> <li>Cross-reference across organizational boundaries</li> </ul>

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

from grizzly to

# Mitaka

7 times upgraded

total **4Region**  
**Heat/**  
**Trove/**  
**Octavia/**  
**barbican**

2017.7

from grizzly to

# Kilo

5 times upgraded

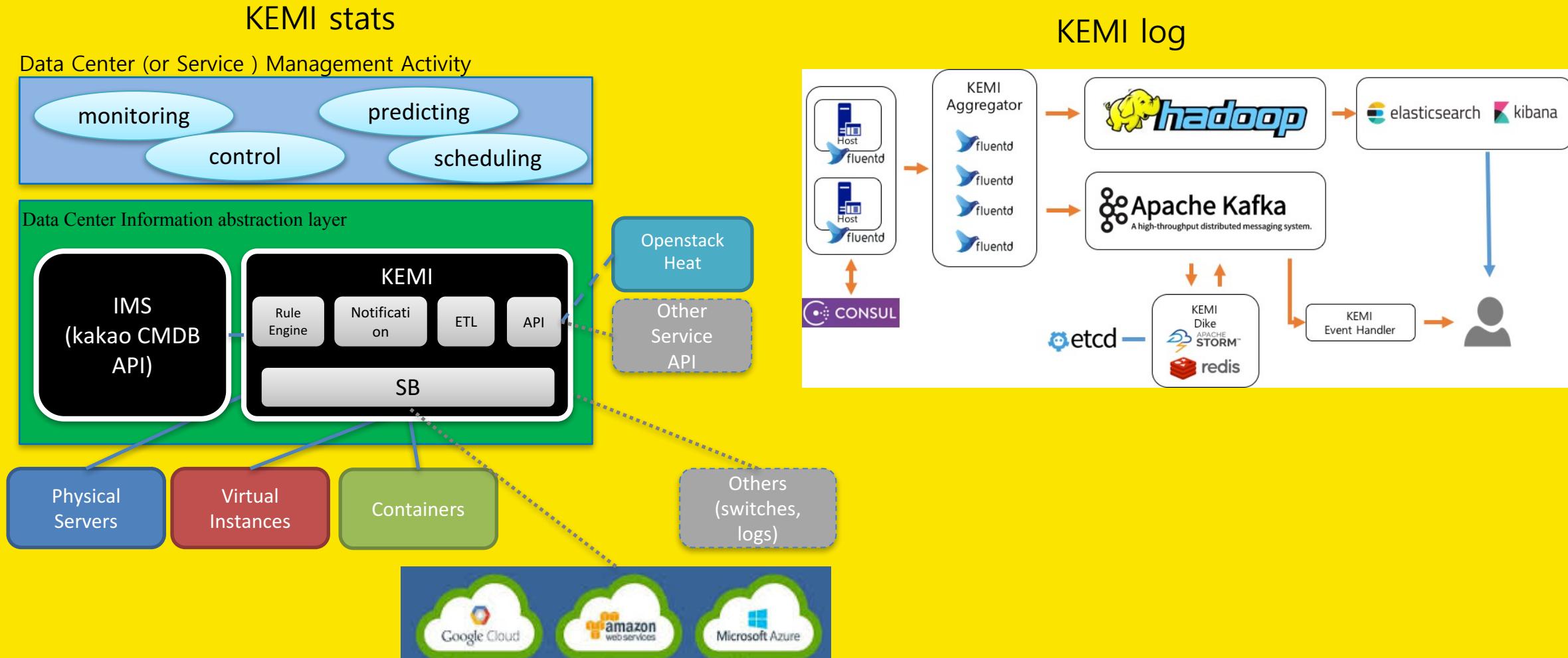
total **4Region**  
**Heat/**  
**Trove/**  
**Sahara**

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)

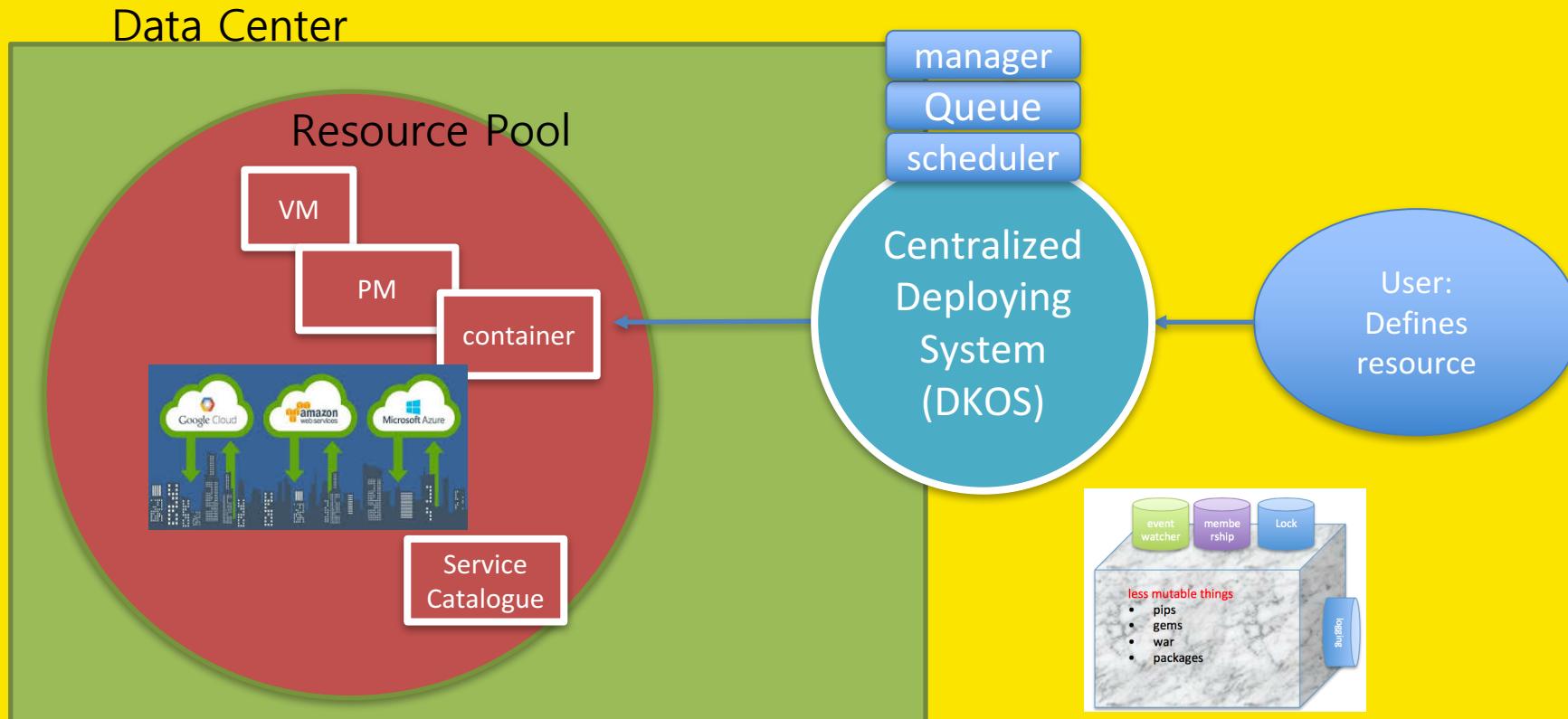
# CMMI 2 Case: event monitoring/alert platform kakao, KEMI



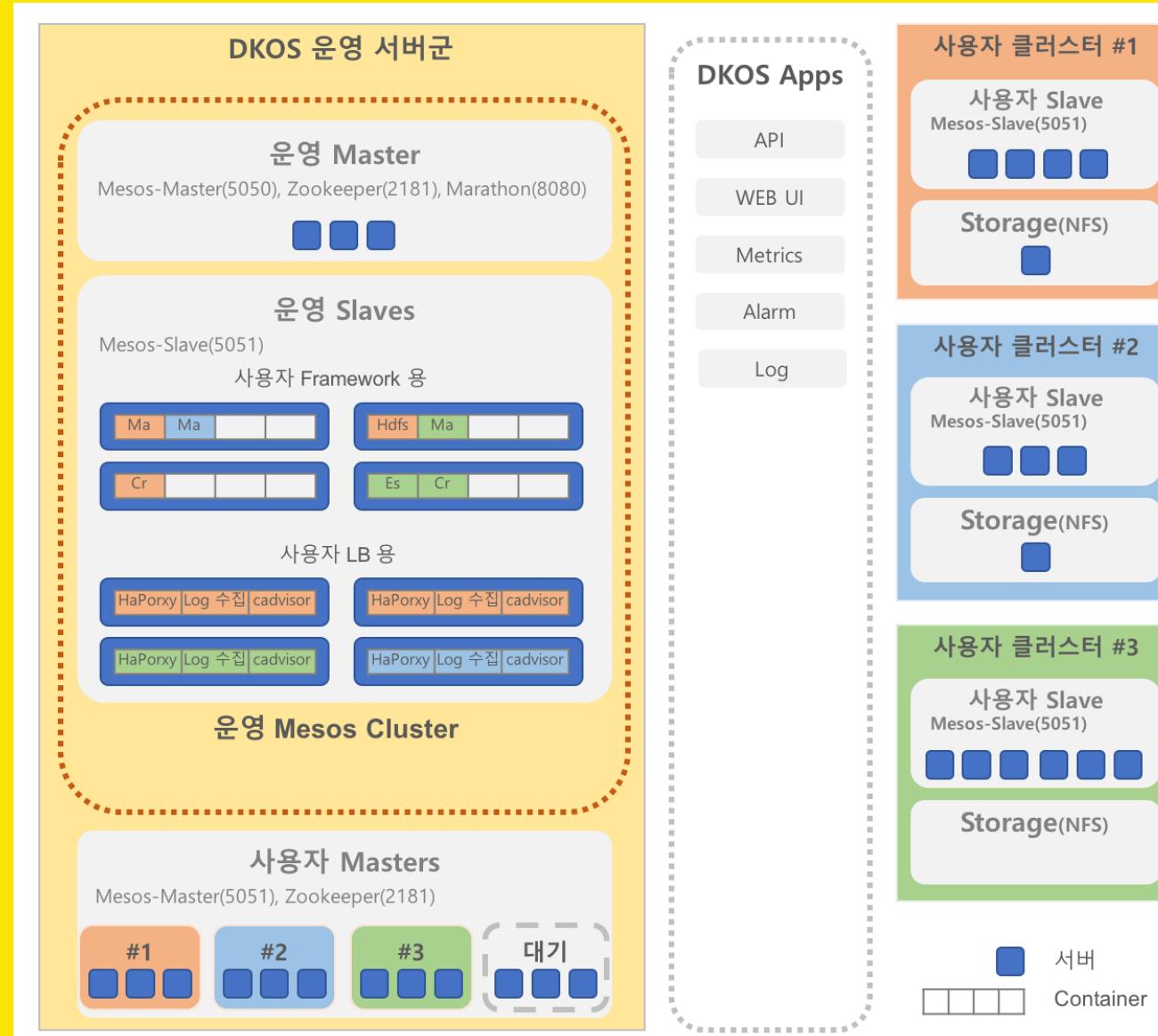
## What is the sole purpose of doing cloud?

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 Architecture



# Services over DKOS

This section shows a collage of screenshots from different mobile platforms:

- Top Left:** A blurred screenshot of a messaging application showing a list of contacts.
- Bottom Left:** A screenshot of a news app titled "SBS News" displaying a story about a university student's death.
- Bottom Center:** A screenshot of a game titled "게임별" featuring a cartoon rabbit character and a mini-game interface.
- Bottom Right:** A screenshot of a news app showing a story about a university student's death.

This screenshot shows a mobile application interface with the following elements:

- Header:** SKT logo, signal strength, battery level (92%), and time (오후 10:10).
- Section Header:** 게임별 (Game by Game).
- Profile:** Lv. 6 정원천\_hardy (Profile picture of a person in a blue shirt).
- Text:** 별이 쏟아지는 아름다운 스펙게임 '보고싶네오' 오픈! 지금 당장 하기!! (A beautiful starry speck game '보고싶네오' has opened! Play now!!)
- Image:** A small image of a colorful cube puzzle.
- Text:** 평의 네모네모팡 (Ping! Bang! Tearing off a colorful tile puzzle).
- Text:** 32일 남음 (32 days left).
- Text:** 게임하기 (Play Game).
- Text:** 마이리틀세프 (My Little Chef) with a price of ₩20,000.
- Text:** 11시간 남음 · 169,049개 남음 (11 hours left · 169,049 items left).
- Text:** 피자 수퍼클리어 (Super Clear Pizza) with a price of ₩2,369.23.

This screenshot shows a mobile application interface with the following elements:

- Header:** SKT logo, signal strength, battery level (93%), and time (오후 10:13).
- Section Header:** kakaohairshop (카카오헤어샵).
- Navigation:** 내 주변 (Nearby), 스타일 (Style), 나의 예약 (My Reservation).
- Image:** A large image of a man with a new hairstyle.
- Text:** 손질이 편한 투블럭 쉐도우펌 (Trend Style).
- Image:** A woman with a short, layered haircut.
- Text:** 미에로화이바 마시고, 할인쿠폰 겟! (Get a discount coupon by drinking Mierohwiba).
- Image:** Two women with different hairstyles.

This screenshot shows a mobile application interface with the following elements:

- Header:** SKT logo, signal strength, battery level (89%), and time (오후 11:11).
- Section Header:** 주문하기 (Order).
- Navigation:** 치킨 (Chicken), 피자 (Pizza), 버거 (Burger), 한식+ (Korean+).
- Category:** BBQ (bbq) with a promotion: 매주 화요일 최대 5,000원 할인 (5% off every Tuesday).
- Category:** BHC (bhc) with a promotion: 최대 5천원 할인 + 토니모리 썬크림 증정 (5% off + TonyMoly Suncream gift).
- Category:** 또래오래 (Doreoage) with a promotion: 매주 목 치킨 최대 5,000원 할인 (5% off every Thursday).
- Category:** 네네치킨 (Nene Chicken) with a promotion: 배달시간이 아닙니다. (Delivery time is not applicable).
- Category:** KFC (KFC) with a promotion: 올쉐킷+그릴바베큐버거 무료 (Free All-Shield+Grilled BBQ Burger).
- Category:** Goobne (Goobne) with a promotion: 배달시간이 아닙니다. (Delivery time is not applicable).

This screenshot shows a mobile application interface with the following elements:

- Header:** kakaofarmer (카카오팜) logo.
- Section Header:** 신규상품 (New Product).
- Text:** 집에서 만드는 건강 애기스 청매실청 DIY KIT.
- Image:** A bowl of green apples and a container of the product.
- Section Header:** 추천 (Recommendation).
- Image:** Various food products: 맨나박스 해미믹스 세트(150gx2), 맨나박스 오리지널 채소(400g), 무한생체 육삼+오리 지발 쇠고기 세트, 무한생체 육삼+오리 지발 쇠고기 세트.
- Text:** 매주 화요일 최대 5,000원 할인 (5% off every Tuesday).
- Section Header:** 내 손으로 만드는 건강한 과일청 (My Hand-Made Healthy Fruit Juices).
- Image:** Various fruit juice products: 청매실청 DIY KIT, 황마차청 DIY KIT, 하급청 DIY KIT.
- Text:** 주식회사 카카오는 둘신판매중개자로서 카카오몰 주문하기와 거래 담당자가 아니며, 입점 판매자들이 등록한 상품 정보 및 가격에 대해 책임을 지지 않습니다.

- 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 2<sup>nd</sup> Session “카카오 DKOS”

Where are you from CMMI-Cloud perspective?

---

## For CMM4, Time to embrace Clouds, not a Cloud

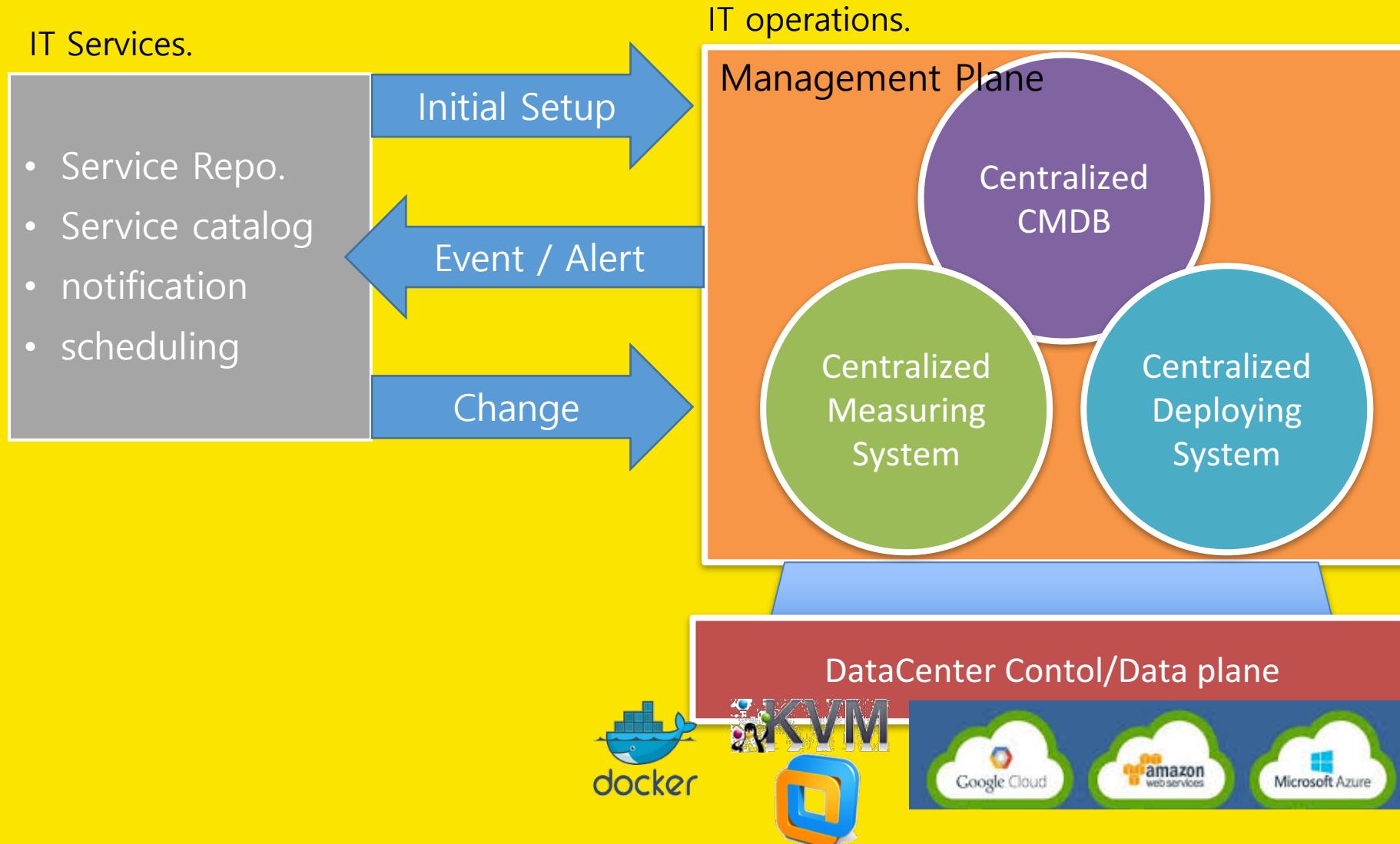
CMM0	CMM1	CMM2	CMM3	CMM4	CMM5
legacy	self service Dev resource	limited Prod resources	Automated CloudUsage	Manual Cloud Usage	Federated Cloud usage
output: cloudTF	output: krane (openstack cloud)	output: kemi (MaaS)	output: DKOS (CaaS)	--	--

Before tackling CloudS?

---

**Abstraction  
should come first**

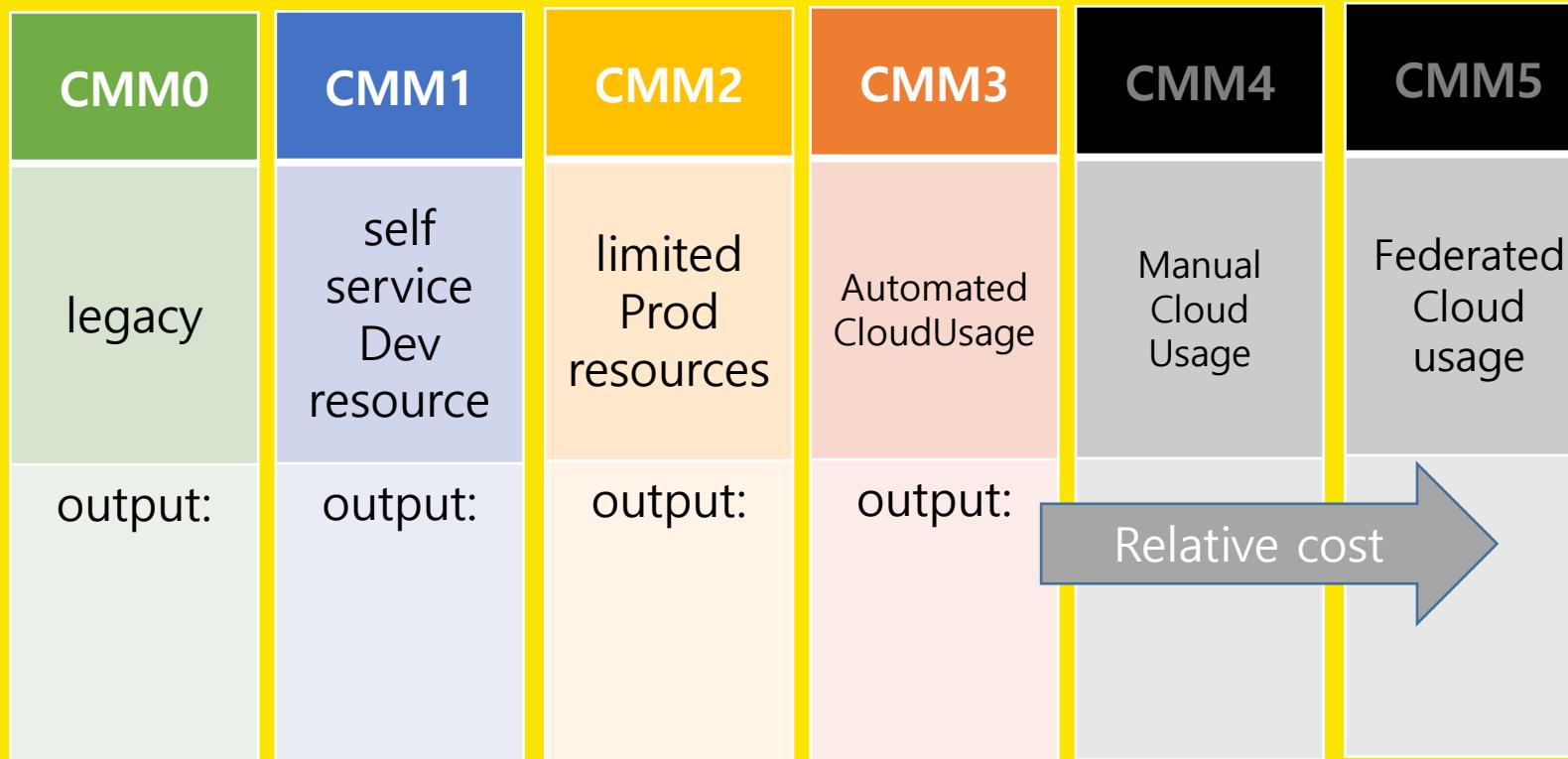
# Fully abstracted operation



Which drives you wild?

---

## What drives you from CMM3 to CMM4/5?



You should say

---

Cloud is Not Done!  
Clouds is just started



# Thanks