
KubeCon
CloudNativeCon
North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022
TiKV
TiKV
HELM

KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

Please scan the QR Code above to
leave feedback on this session

KubeCon CloudNativeCon North America 2022
BUILDING FOR THE ROAD AHEAD
DETROIT 2022

Surviving from Endless Issues Coming from 7K+ Kubernetes Clusters

Seok-yong Hong & Wanhae Lee, Kakao Corp

About Us

Seok-yong Hong

- Seoul, Republic of Korea
- Leader of CloudNative Cell at Kakao corp.
- Developing a Private Cloud at Kakao corp.

About Us

Wanhae Lee

- Seoul, Republic of Korea
- Cloud Engineer at Kakao corp.
- Developing a private KaaS
- First time in...
  - North America
  - KubeCon
  - KubeCon as a Speaker

Background: About Kakao

in Kakao Community,

kakao	Major mobile messenger in South Korea, Web Portal, Map, Blog and more
kakaomobility	Taxi, Bike, Bus, Train, Plane, Package Delivery and more Mobility **as a Service**
kakaopay	Payment, Money Transfer, Investment, Load and more Financial **as a Service**
kakao ENTERTAINMENT	Webtoon, Novel, Music **as a Service** / Creating Films, TV series and more

kakaobrain kakaocommerce kakaostyle kakaobank GROUND X
kakaoenterprise kakaogames kakaospace kakaopiccoma

and more..!

Please visit kakaocorp.com to learn more

Kakao and Kubernetes

Most of Kakao’s services are running on Kubernetes now.

We started the transition in 2018, and complete > 99% transition now.

years	**Container Scale**	**Major Transition**
2018	1M	DAUM Mail, News
2019	7M	KakaoTaxi, Melon (Music Streaming)
2020	35M	KakaoTalk + All Service
2021	100M	All Service
2022	110M+	Global Scale Cloud Expansion

Kubernetes in Kakao

	One Large Cluster	**Lots of Small Clusters**
Cluster Management	Easy	**Hard**
Admission Control	Hard	**Easy**
Resource Efficiency	Good	**Bad**
Isolation	Hard	**Easy**
Security	Hard	**Easy**
Freedom	Bad	**Good**

One Large Cluster

vs

Lots of Small Cluster

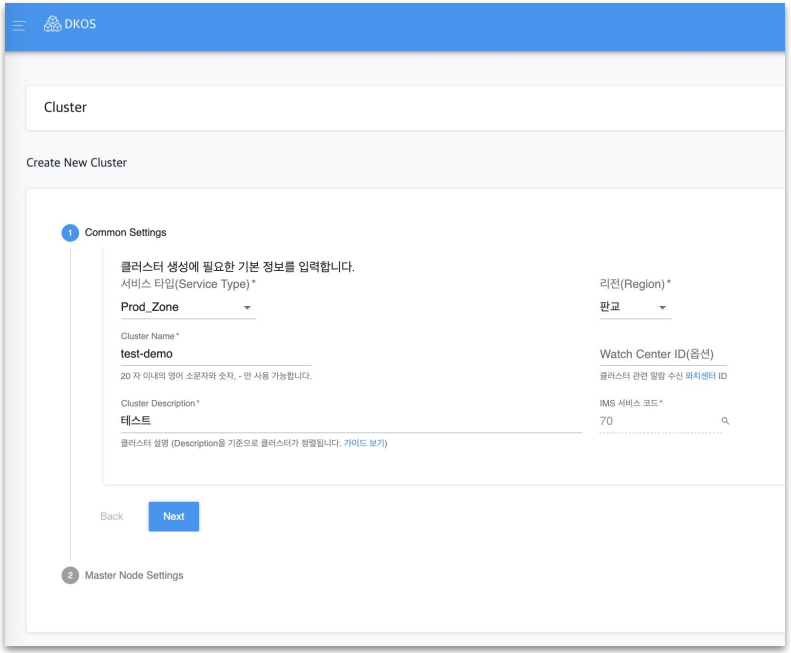
Kubernetes in Kakao

to do that we made a **private Kubernetes as a Service**,

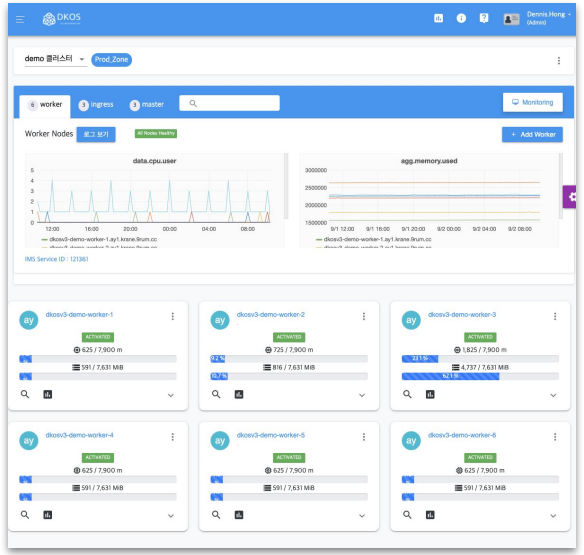
DKOS (Datacenter of Kakao Operating System).

Three steps to get a new cluster.

Sign in with SSO

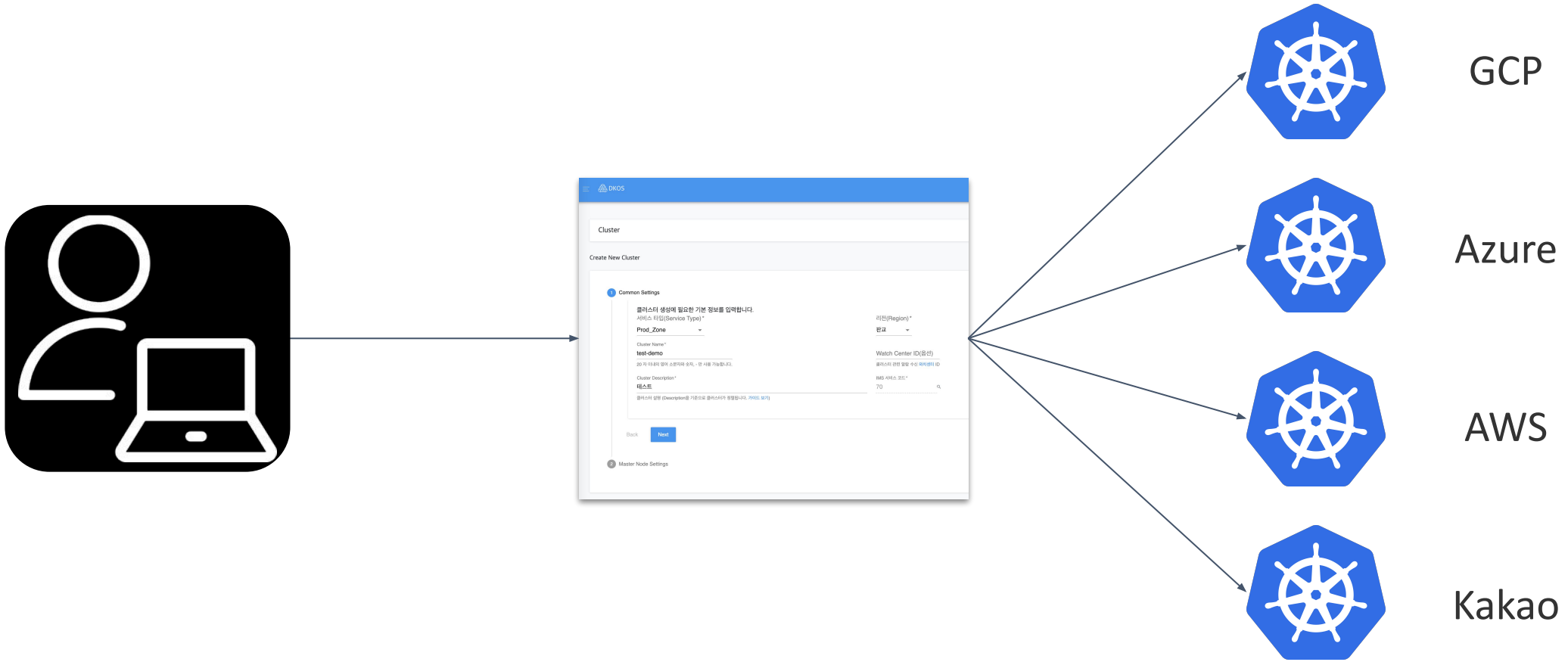


Set Cluster Name
and
Which Zone to provision



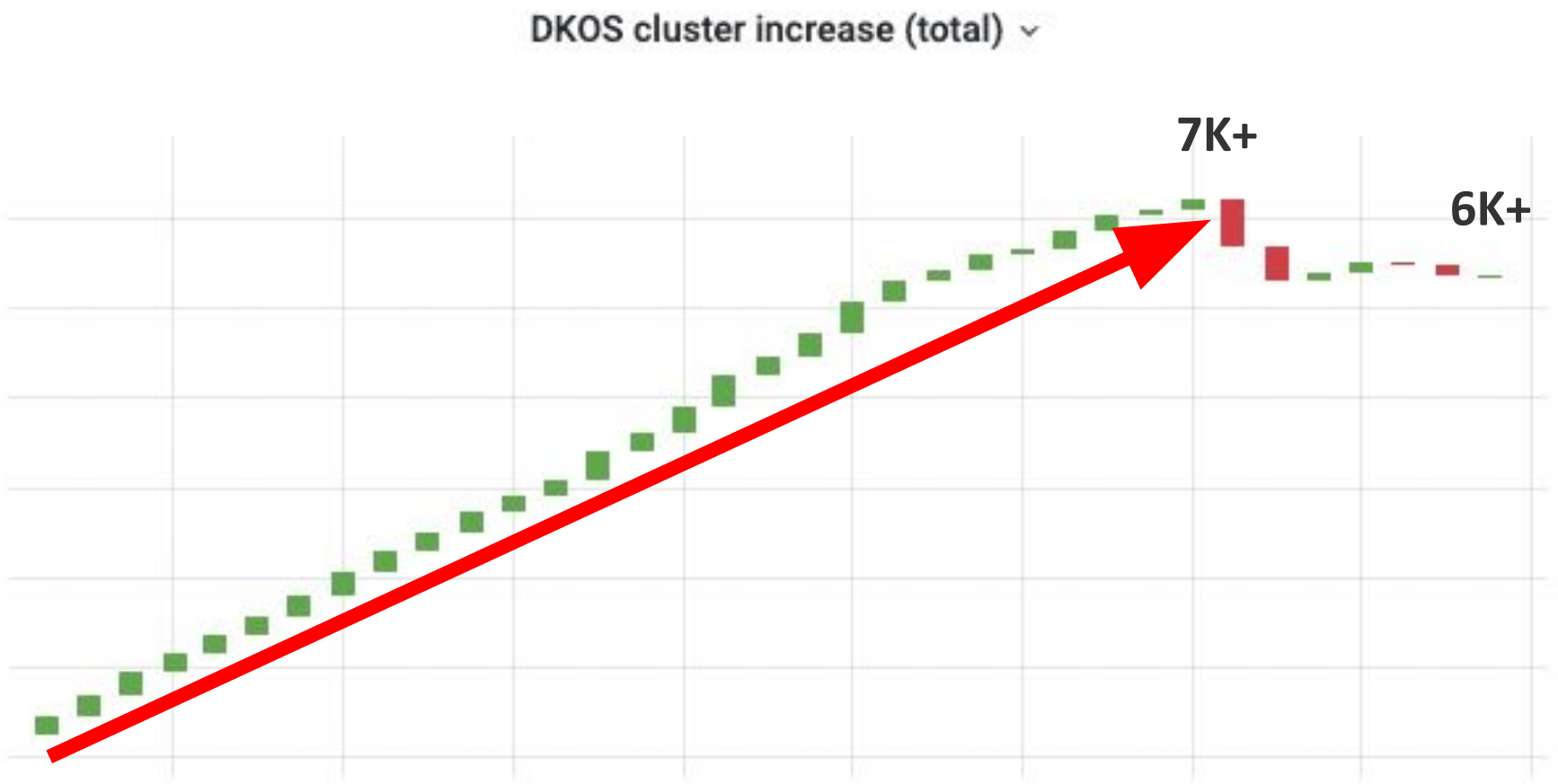
You have one
(with CNI, CSI, CloudProvider,
Ingress, Logging, Metrics, etc)

Three steps to get a new cluster. Anywhere (WIP)



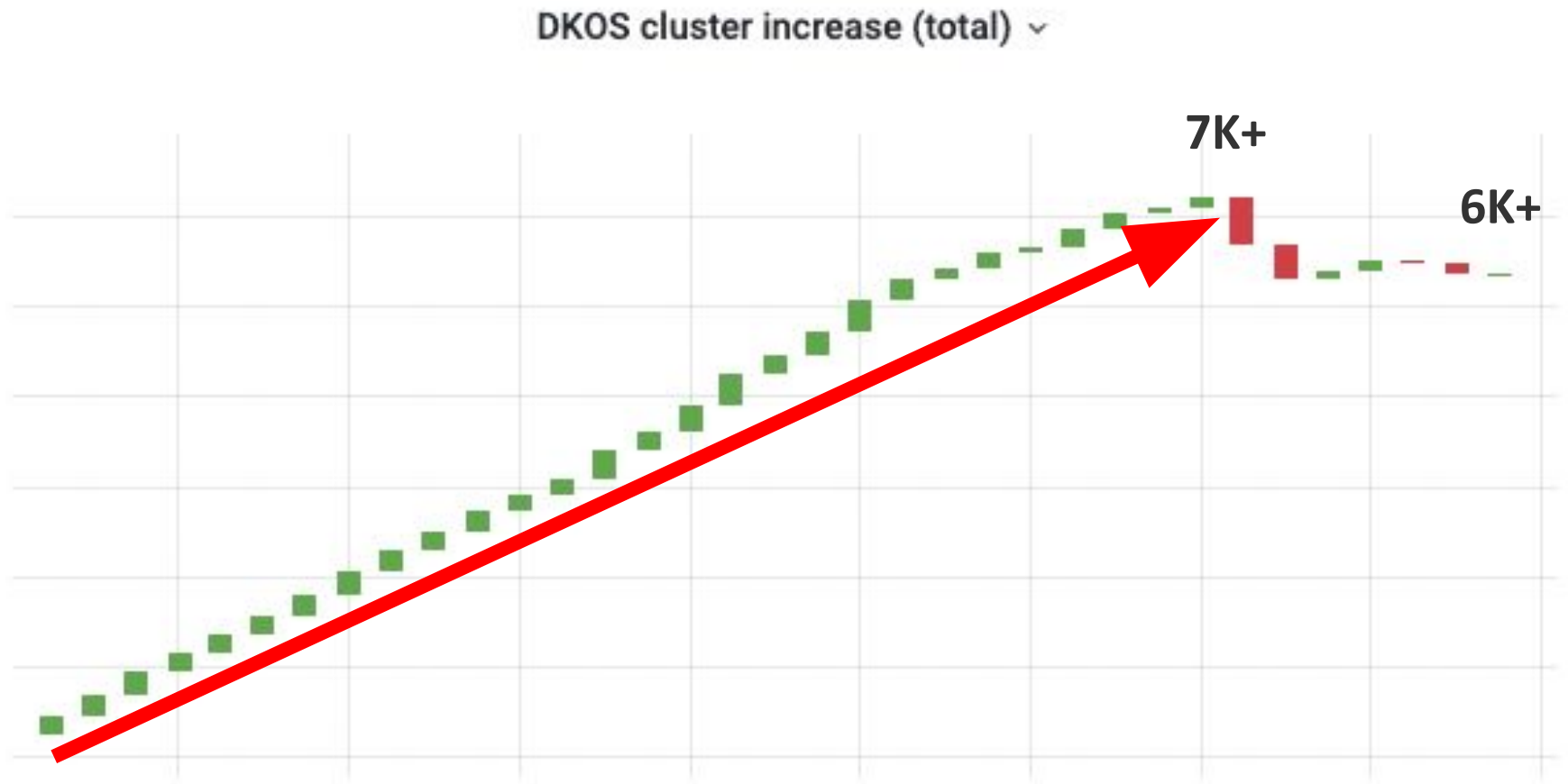
Kubernetes in Kakao: Good News

Developers do use Kubernetes (linearly)



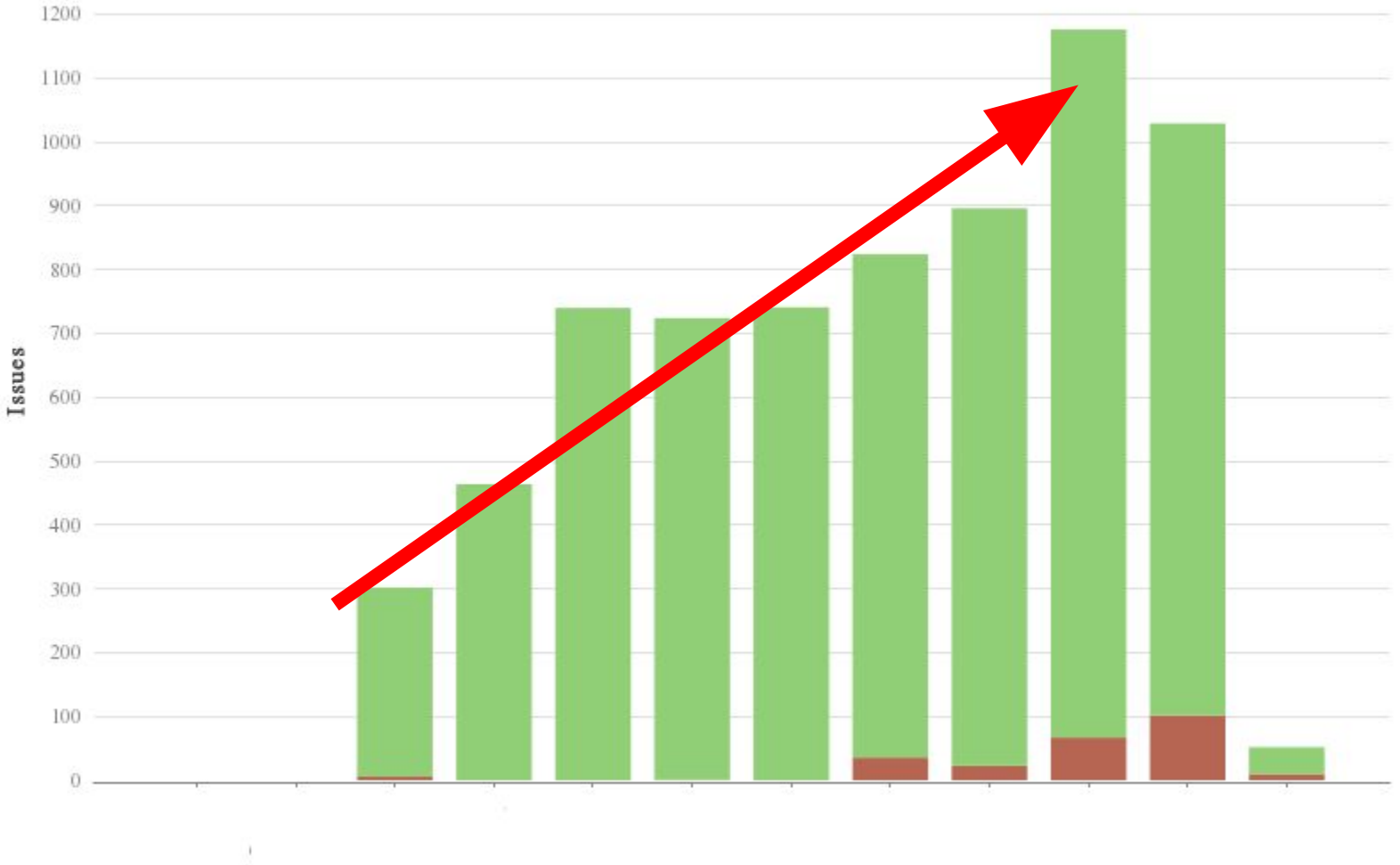
Kubernetes in Kakao: Good News

Developers do use Kubernetes (linearly) TOO MUCH



Kubernetes in Kakao: Bad News

On-call issues grows up, accordingly. 😂



Kubernetes in Kakao Corp.

Pros

- Easy to deploy clean new K8s cluster. (whenever needed immediately)
- Developer can use them for their own use cases. (almost no restriction)

Cons

- **Too easy to make one.**
  The number of clusters **>** the number of developers in Kakao.
- **Almost every edge cases are reported.**
  Starting from collisions between 3rd party Kubernetes applications
  to Linux kernel bugs that happens once a trillion requests.
- **The growth of operational costs is barely manageable.**

Problem 1. Too Many (Unused) Clusters

Too many clusters are not necessary to have remained.

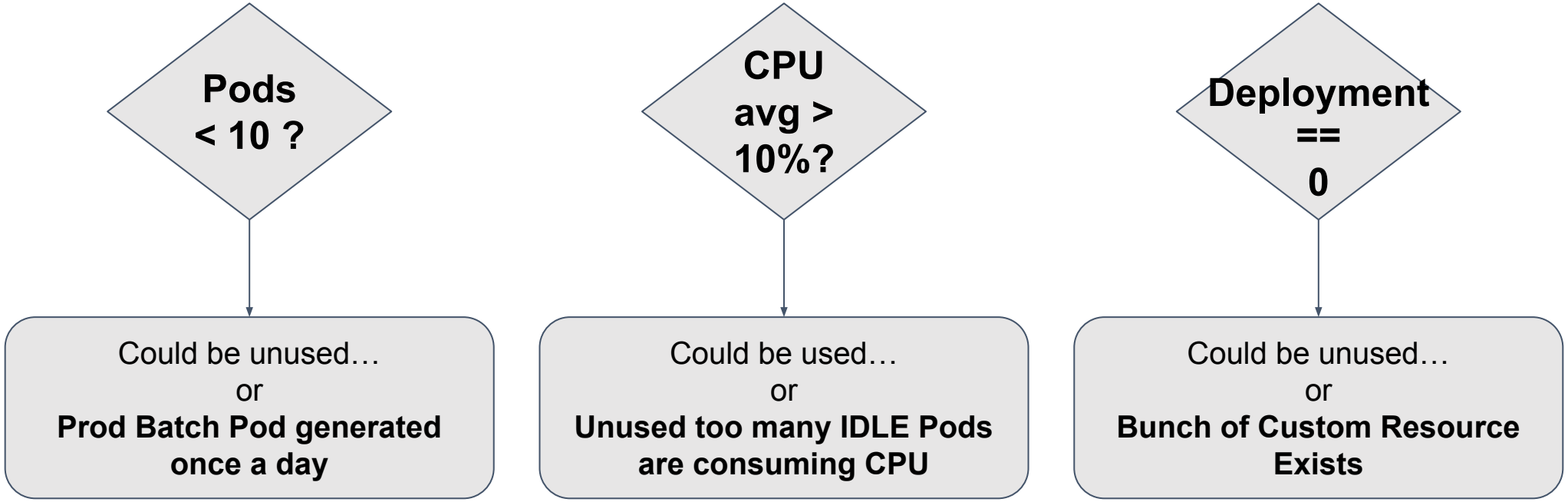
e.g,

- Deployed for a testing purpose, but forgot to delete.
- Allocating much more resources than they need.
- The person in charge has moved.
- Create one, just not delete.

It is okay if we have “**an infinite space of data center**” with “**an infinite number of servers**”, which is not :-(

Problem 1. Too Many (Unused) Clusters

Determining whether "**is this cluster in use?**" is not an easy task.



Exceptions Everywhere!

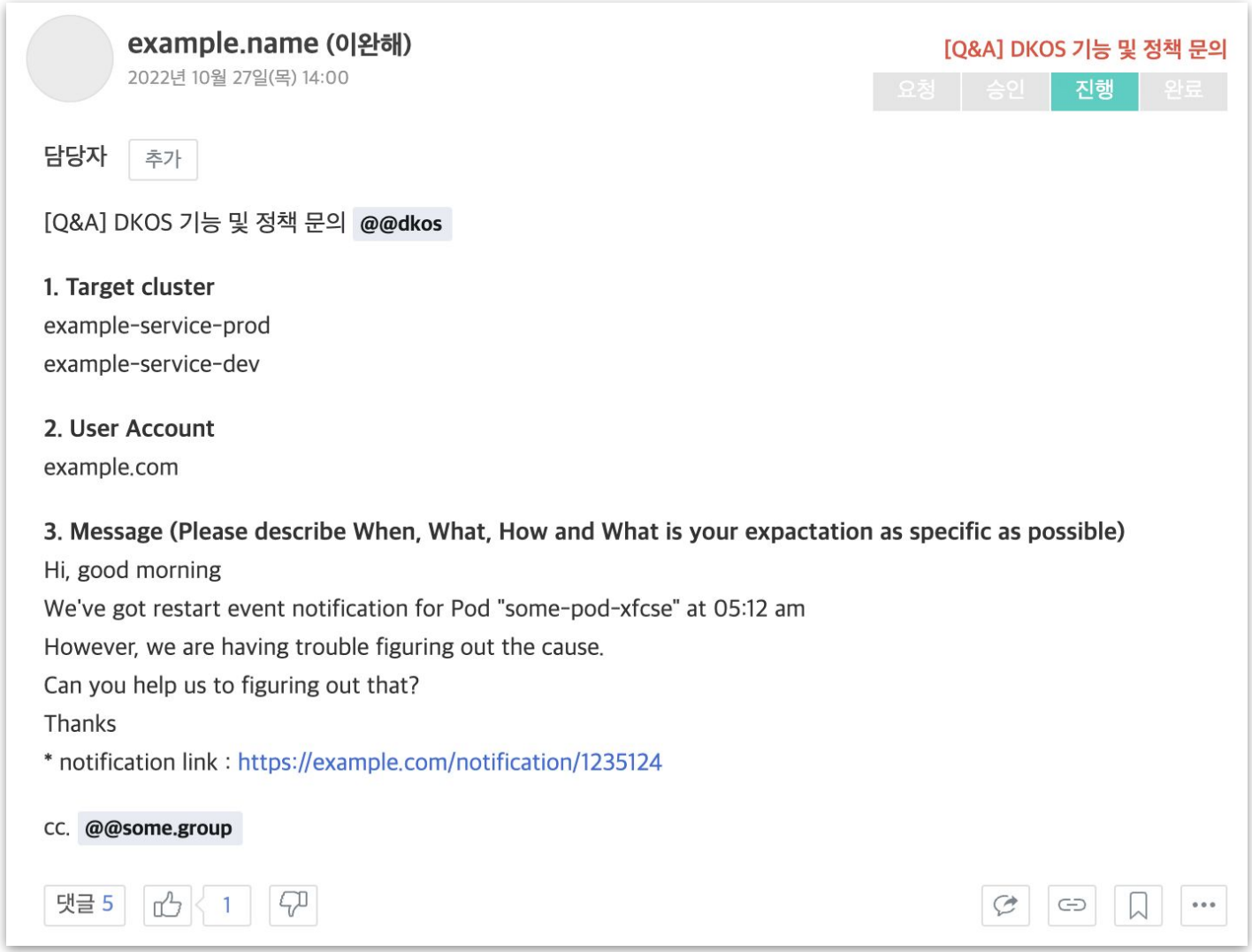
Have to consider multiple factors.

Problem 2. Lots of on-call Issues

in average,

10+ on-call issues per every working day.

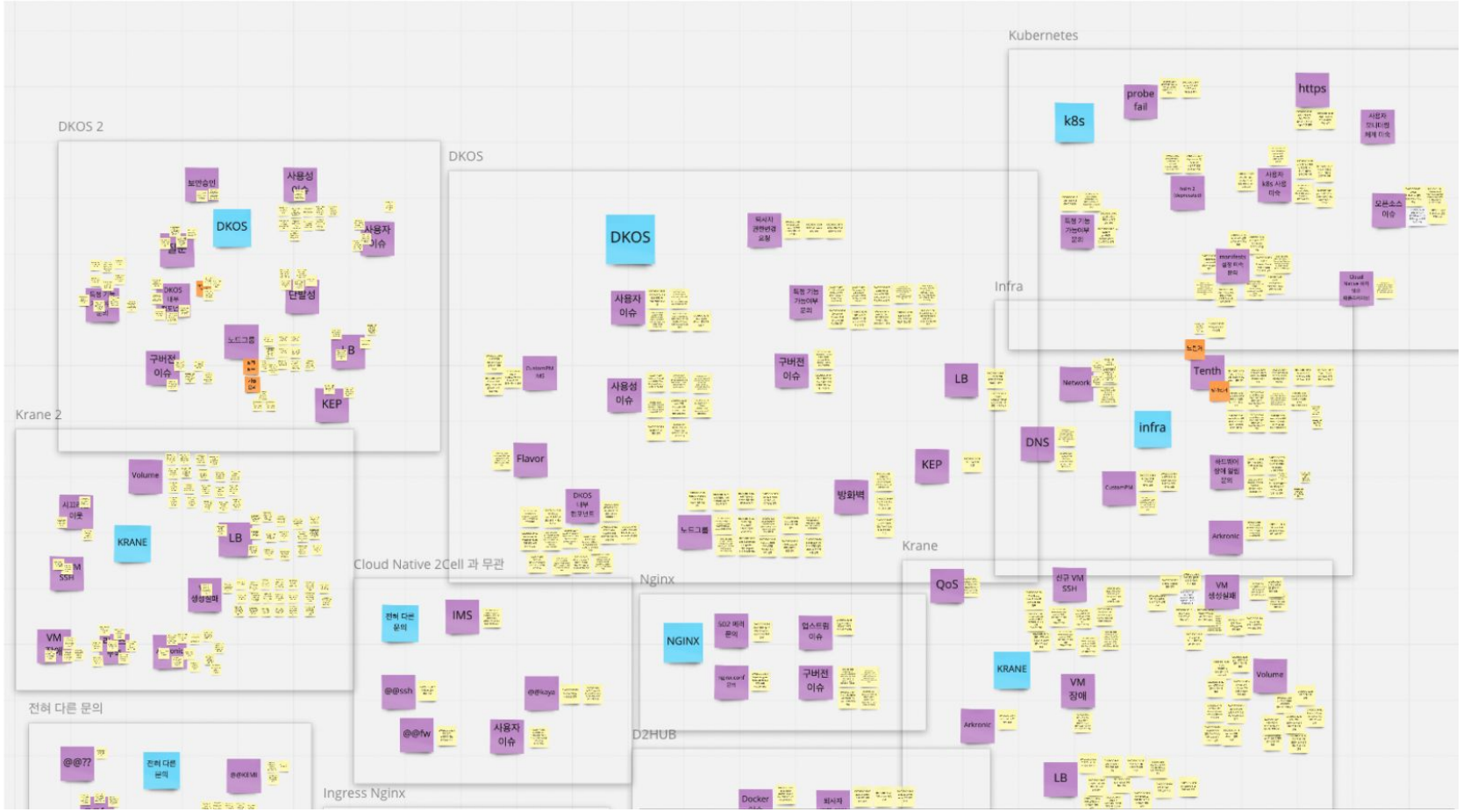
Most of them are inquiry from users.



Problem 2. Lots of on-call Issues

With a half year of data, We've collected 1,000+ inquiries from users

Conducted qualitative research based on Grounded theory.



Problem 2. Lots of on-call Issues

Not all developers do know well about Kubernetes.

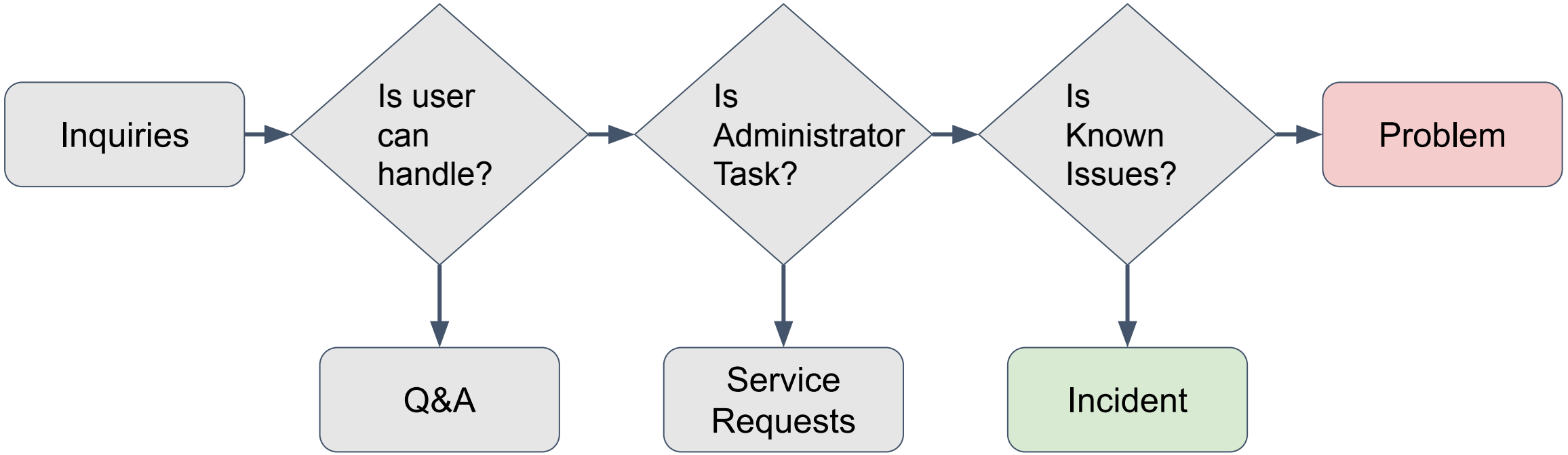
Most of their issues come from **not knowing the details of Kubernetes.**

- Use local directories with “hostPath” (which is not persistent) (but requires persistent)
- Service denial when rolling out Pods (no concern of the graceful shutdown)
- Use “latest” container image tag and asking “why the old image is deployed on cluster”
- Taints some node and forget
- Forget their TLS Certification is expired.
- Not set resource limits and requests, and got Node OOM.
- Can not access to their service, because of minor typo in ingress manifest.

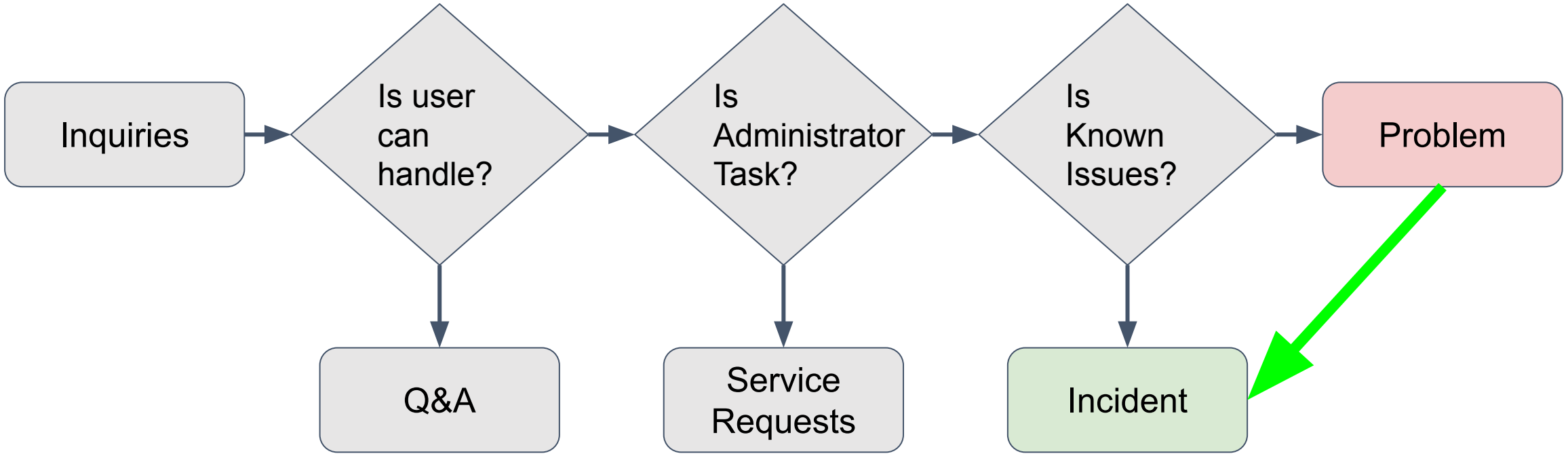
However, all of these **can be fixed, just by someone points out.**

All we need is **someone** other than us 😂

Problem 3. Known issues being forgotten

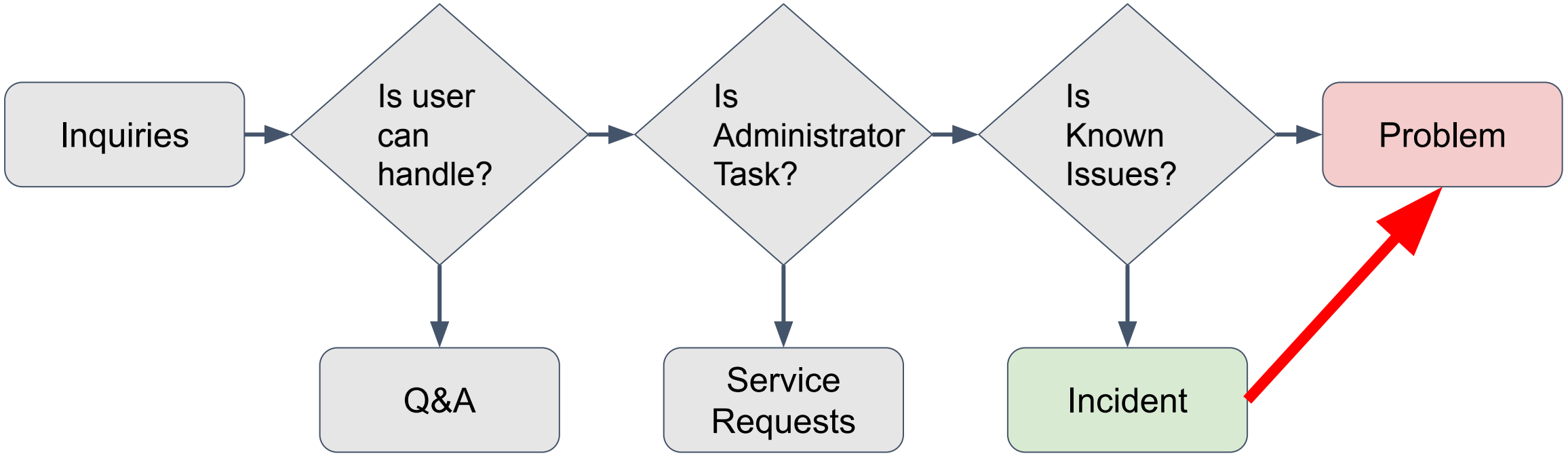


Problem 3. Known issues being forgotten



Even though we know some of issues right now,

Problem 3. Known issues being forgotten



Even though we know some of issues right now,

We forget after a few years :-(

Automation for Operation

We have a

- Chatbot
  doing routine tasks without bunch of CLIs

- Event Monitoring Tools
  watching Kubernetes events and give notifications.

- Resource Monitoring Tools
  watching resource usage of each node, and give notifications before dead.

- Component Monitoring Tools
  use "/healthz" or equivalent API.

Useful tools, but not enough

We need a “Detection as a Code”

We need a “Detection as a Code”

to examine the factors (to delete unused cluster)

to let users know what could be a problem (without human intelligence)

to find a known issue (without waste of time)

detek:
detecting Kubernetes known issues

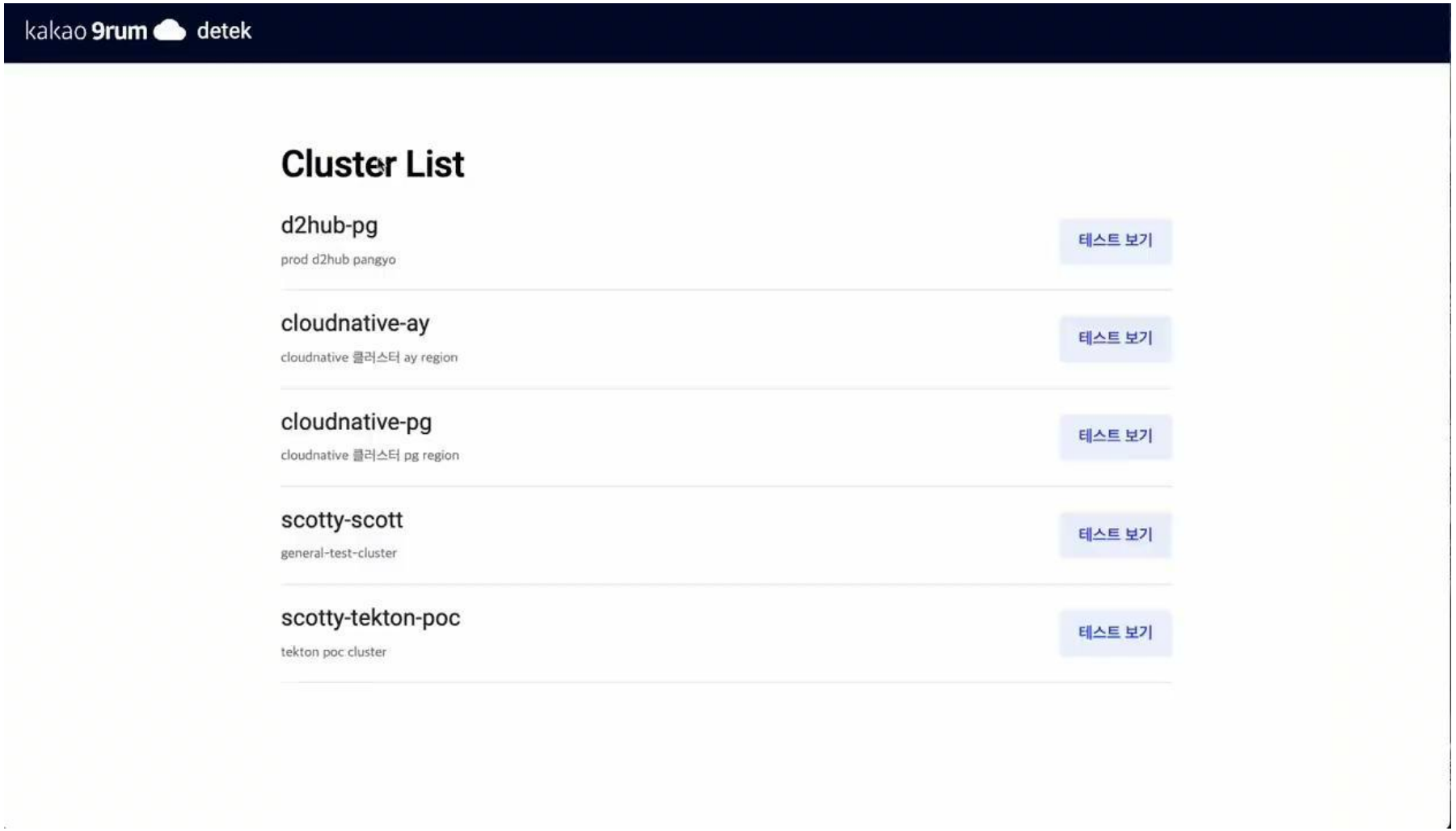
Extensible problem-detecting CLI tool

for **reliable Kubernetes cluster operations** and **rapid problem detecting**.

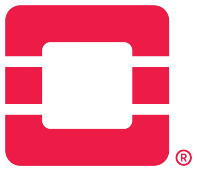
made by Go

https://github.com/kakao/detek

detek: How we use it



detek: Internal Structure

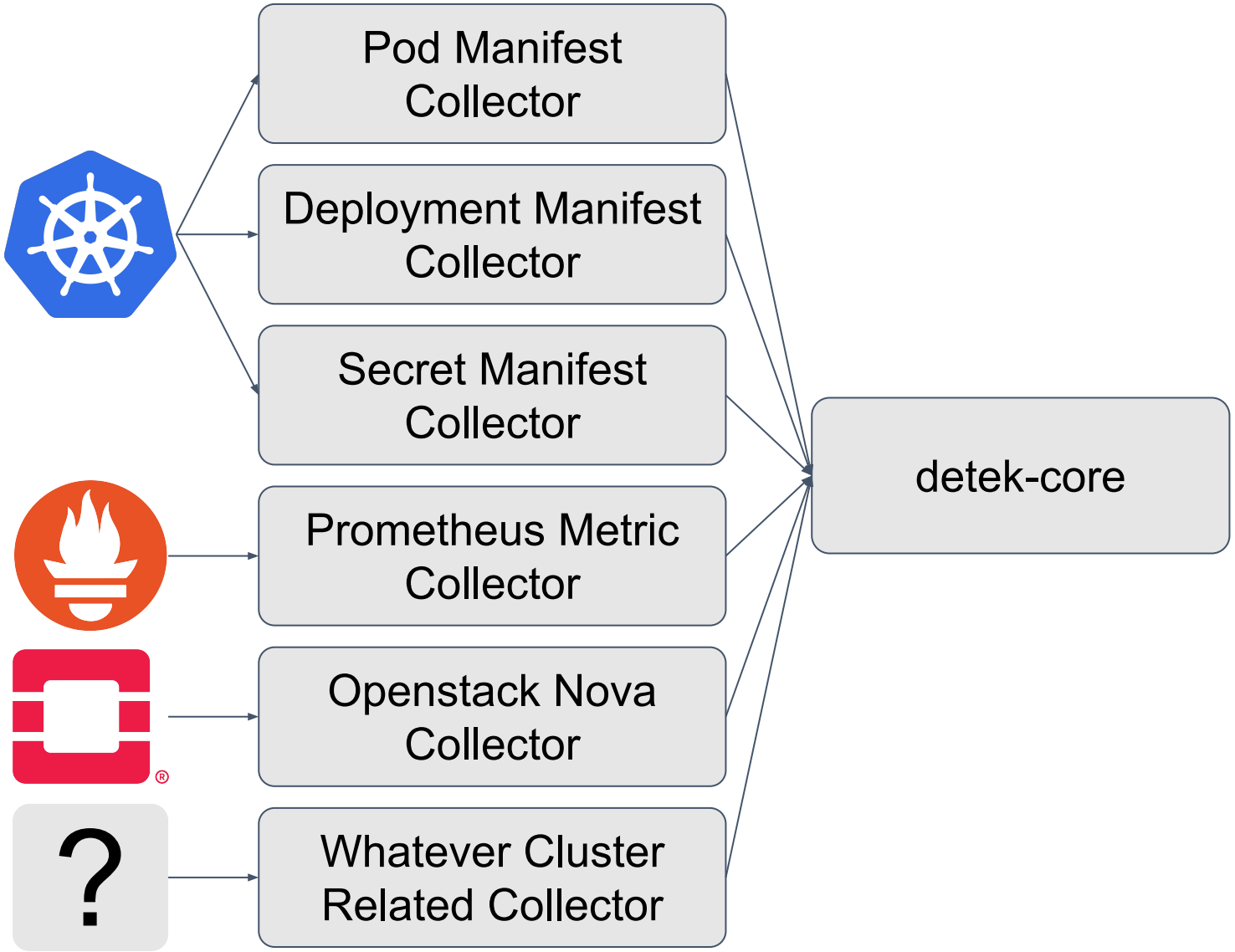


From a variety source of targets

detek: Internal Structure

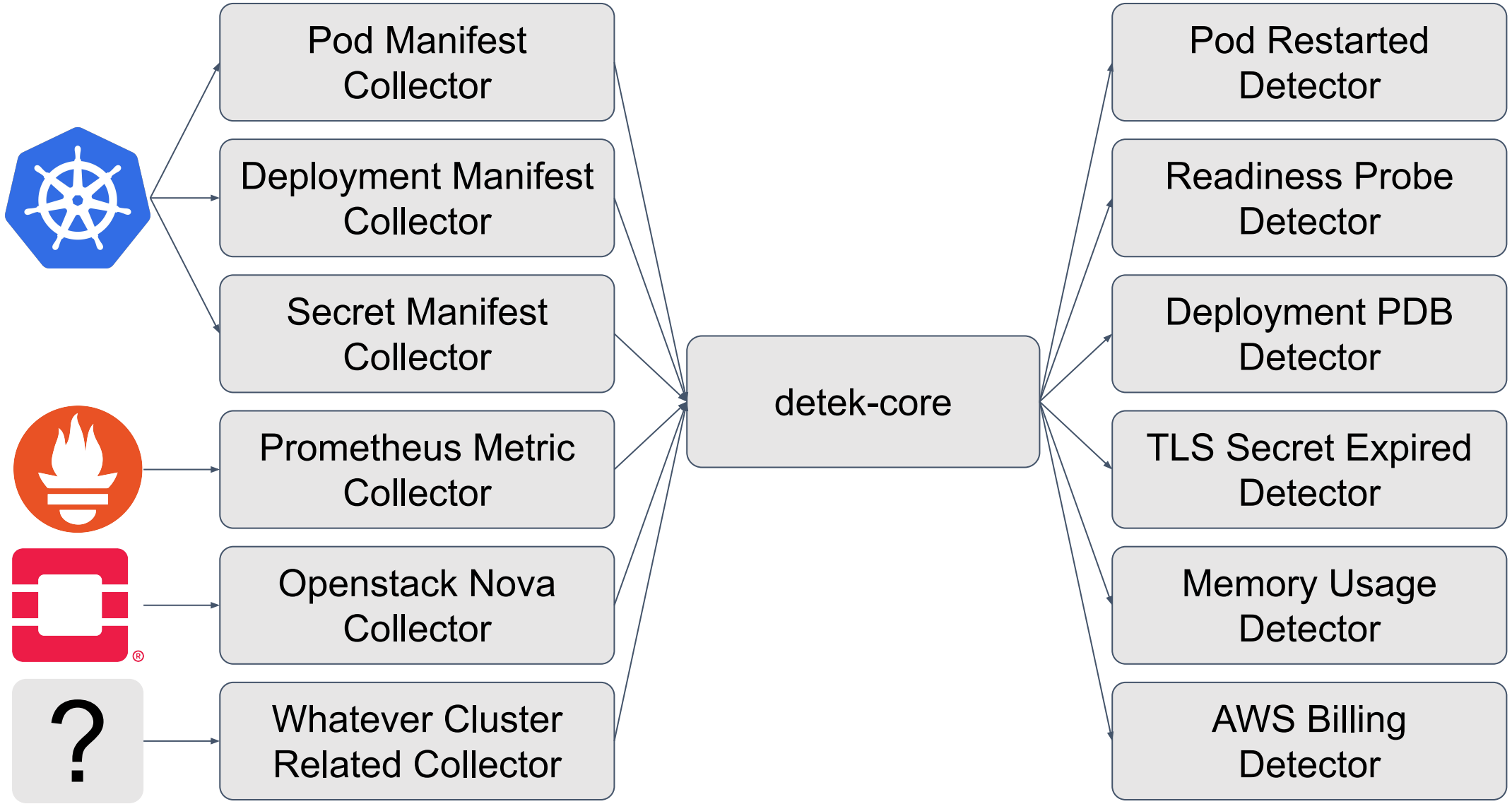
“Collectors” collect data

detek: Internal Structure

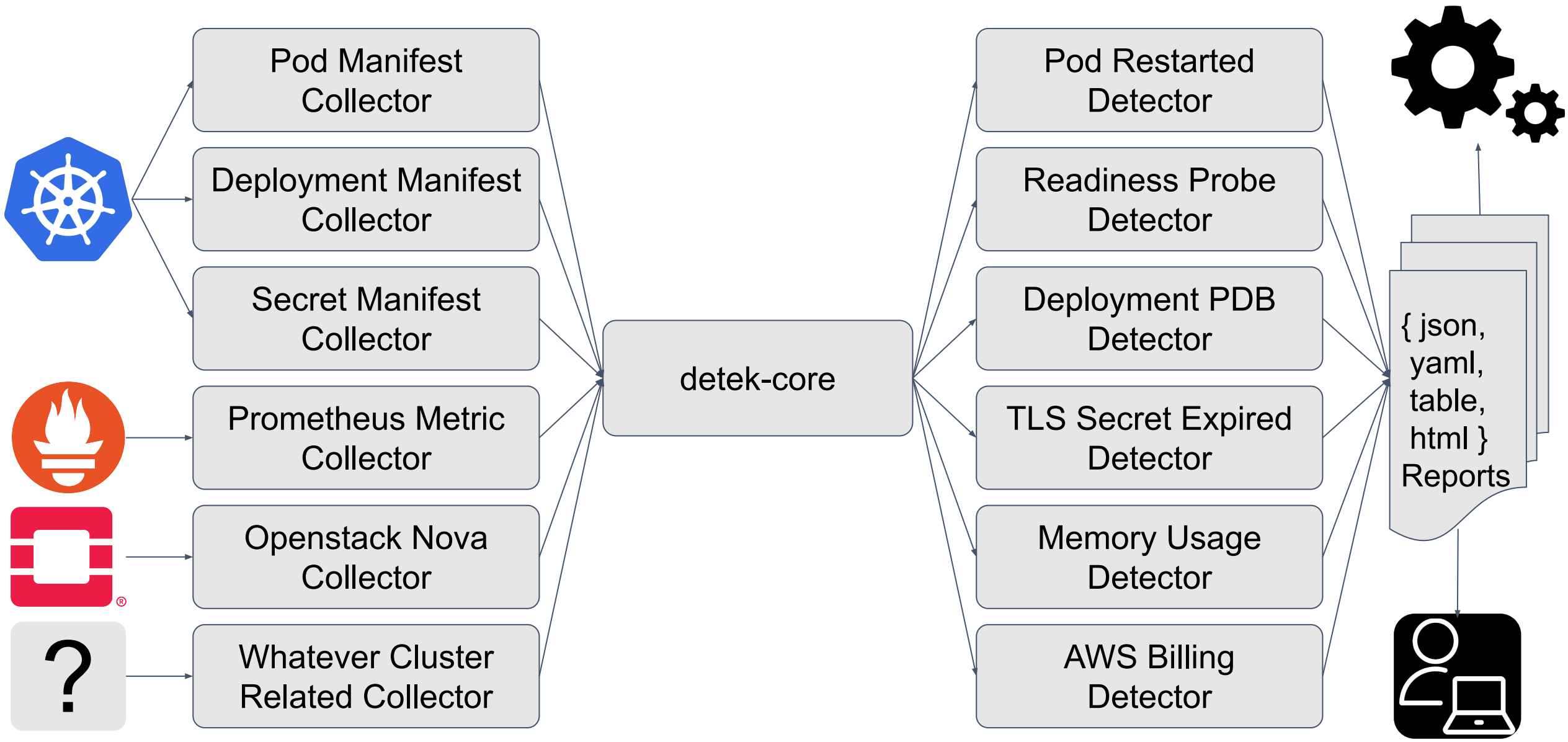


Save it to KV storage in detek-core

detek: Internal Structure



detek: Internal Structure



detek: Internal Structure

- Pod Manifest Collector
- Deployment Manifest Collector
- Secret Manifest Collector
- Prometheus Metric Collector
- Openstack Nova Collector
- Whatever Cluster Related Collector

<- Collector / Detector ->

Extensible Components

- Pod Restarted Detector
- Readiness Probe Detector
- Deployment PDB Detector
- TLS Secret Expired Detector
- Memory Usage Detector
- AWS Billing Detector

Container Image from unknown registry

Connect via SSH and check Kernel Params

Admission Webhook with outdated Certification

Check all kube-apiserver functional

Simple Structure, but Effective

Wrong Pod IP (IPAM edge case)

Check OOMKilled Pods

Deprecated version warning

Non functional Ingress Resource

Check services have a proper endpoints

detek is an open source!

Kubernetes is a **de-facto standard** for today's development and operations.
But, **this does not mean that everyone can use it well.**

Hope this helps K8s users to use K8s more effectively.

Whatever kind of contribution is welcomed :-)

https://github.com/kakao/detek

Q&A

Seok-yong Hong & Wanhae Lee, Kakao Corp