Kubernetes and the Cloud

3.1 Introduction

If you have been keeping on eye on the cloud computing scene, you may have noticed that news relating to containerization technology seems to appear daily. You can be certain that news containing keywords like Docker, Kubernetes[*1], and CNCF falls into this category. And as the ecosystem continues to expand, with various products based on Docker and Kubernetes being created, it is not uncommon for stories to actually be about container-related technology even if they don't appear to be at first glance.

Although it is undeniable, to an extent, that industry players, particularly the mega cloud vendors, will be sorting out the cloud industry rules for a while to come, a look at the rise of containerization technology compels me to believe that the industry rules are likely to be rewritten by upcoming trends earlier than we had thought. Perhaps events set to have an even greater impact on the IT industry than the emergence of IaaS are afoot.

At IIJ, we have also started to make wide use of this technology to, for instance, enable rapid business deployments and the efficient, high-quality operation of large-scale systems, develop highly portable software, and optimize costs via the efficient use of infrastructure. I will discuss the IKE (IIJ Container Engine for Kubernetes) system that we use internally further below, but first I would like to explain why containerization technology has been thrust so rapidly into the spotlight and what impact this technology is likely to have on cloud computing.

3.2 Docker and Kubernetes

Products and services that encompass containerization technology are springing up like mushrooms, but only two products lie at the center of all this: Docker and Kubernetes. Catching up on developments surrounding these two products is a good way to familiarize yourself with the key trends in the rapidly advancing containerization space.

The relationship between the two is slightly complicated, but in the simplest terms, Docker is a container engine that starts programs and wraps them in containers, while Kubernetes is a container orchestrator that bundles together and controls multiple container engines. In general terms, an orchestrator is a controller that coordinates between multiple interrelated systems and integrates them into a single overall system. A container orchestrator like Kubernetes bundles together the multiple host nodes started by the container engine (e.g., Docker) and controls them efficiently and autonomously as a single large resource pool. Although Docker, as a product, competes with Kubernetes in some respects, it is less confusing to to think of the relationship between them as being one of a container orchestrator and a container engine (Figure 1).

That said, these two products are not always paired with each other. Docker is already widely recognized and used for the convenience it offers as a standalone system, but Kubernetes does not seem to be used quite that extensively in production environments. This is because Kubernetes is a platform designed to control container clusters of a fairly

[*1] Kubernetes is a container orchestrator that controls container engines, such as Docker, and manages container clusters composed of multiple nodes. It was created by Google and is currently an open source software project hosted by the CNCF (Cloud Native Computing Foundation). It is said to have originally been based on an internal Google system called Borg. Kubernetes is a runtime environment for containerized applications and makes possible infrastructure-independent, portable application packaging, provisioning, and operation. Kubernetes is sometimes referred to as an OS for the cloud era and is looked to as a potential unified operations interface for multicloud and hybrid cloud environments.

To use Kubernetes, you need network drivers, storage drivers, traffic managers, and so on to match your infrastructure, but Kubernetes basically does not include implementations of these elements. And to set up a Kubernetes environment properly, portal and monitoring tools for application management and account management are also essential, but these sorts of tools also need to be found elsewhere. This is the reason for the emergence of Kubernetes distributions, which package Kubernetes together with an environment for it to run in as well as an installer and other tools. IKE (IIJ Container Engine for Kubernetes), which I discuss below, is one such Kubernetes distribution.

decent size, so there are certain hurdles to overcome even if you just want to try it out, whereas Docker can also serve as a convenience utility in your local computing environment and is therefore a simple and easy choice even for minor use cases. A lot of engineers are probably making convenient use of Docker as a tool for setting up test environments, and as a means for distributing software. Docker is already on its way to becoming an essential part of the engineer's toolkit.

Yet, of the two, it seems to be Kubernetes that is causing the bigger stir in the IT industry at present. This is because through the use of Kubernetes, containerization technology, rather than merely providing convenient utilities, is expected to significantly change the face of server-side systems.

Why is Kubernetes, which is by no means yet mature, garnering so much attention? Probably because Kubernetes originally came from Borg, which has supported Google's systems for over 10 years. Google's internal systems are almost never explained in any detail, but by revealing some of the ways in which it is used, the book Site Reliability Engineering (the SRE book, as it is commonly known) offered a glimpse into the surprising realities of Borg. No doubt this is what prompted more than a few people to take an interest in containerization technology. The revelation that Google's systems do not contain virtual machines and that all processes are essentially run as containers had a major impact on engineers, particularly those working in the cloud business. Kubernetes is sometimes referred to as the OSS version of Borg, but it is unclear how much the two actually have in common. As a relative newcomer, Kubernetes may appear to have a limited track record, but it is quite conceivable that its design incorporates best practices that have have been in use, and grown mature, at Google over a long period of time.

3.3 Best Practices for Harnessing IaaS

So then, how is Kubernetes set to change the face of server-side systems? It will facilitate highly portable, infrastructure-independent deployments, goes the narrative. It will offer large-scale cluster management capabilities and make dynamic use of computing resources to provide excellent scalability. The potential of Kubernetes is described in all sorts of ways, but they tend to be vague and somewhat nebulous. That is in some sense inevitable—the role of Kubernetes is akin to that of a computer OS. Try explaining what an OS does to someone with no idea what an OS is and you're likely to either find yourself telling a rambling, fragmented tale or delving into extremely technical detail.

Kubernetes is actually often referred to metaphorically as an OS for the cloud era. Drivers absorb any differences in infrastructure, the configuration of networks and storage is virtualized, and a unified interface is provided to any systems deployed on Kubernetes. Kubernetes makes it possible to design, build, and operate unified systems without relying on IaaS-specific interfaces. That said, Kubernetes is not an OS and does not provide an applications interface. The applications that run on Kubernetes are simply ordinary Linux and Windows applications.

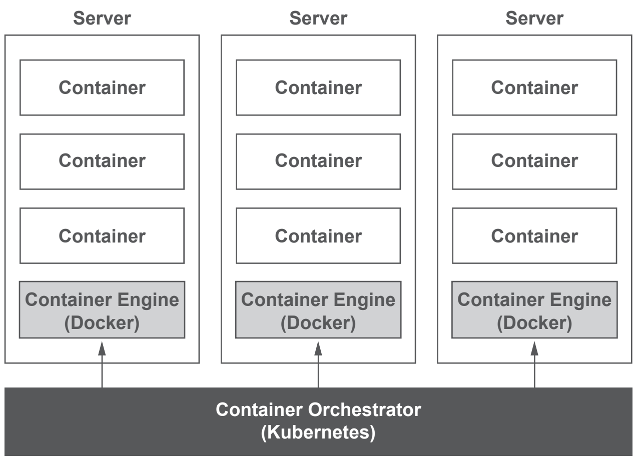
Server
Server
Server
Container
Container
Container
Container
Container
Container
Container
Container
Container
Container Engine (Docker)
Container Engine (Docker)
Container Engine (Docker)
Container Orchestrator (Kubernetes)

Figure 1: Container Engines and the Container Orchestrator

While they are comparable in some respects, the major difference between an OS and Kubernetes is that an OS controls a single computer housed within a physical box, whereas Kubernetes manages multiple networked computers as one large resource pool. Processes managed by an OS cannot leave the physical box, but processes running on a Kubernetes cluster (i.e., containers) can be on any of the nodes that make up the cluster. So when resources are depleted, all the system has to do is increase the number of nodes and reassign containers (this happens automatically), and if a container stops running because of a fault with a node, the container can be restored simply by restarting it on a different node (this also happens automatically).

Cloud services, in many cases, are regarded as having excellent stability, free from outages. But in reality, cloud services are many and varied, and since redundancy is not built into IaaS resources, in particular, such services can stop when faults occur, and they also experience planned outages for maintenance purposes. The spread of IaaS has made it possible to procure and build system resources and to recover from hardware faults in impressively short amounts of time, but for the most part, not that much has changed in the way systems are operated. Whether virtualized or not, if you're still dealing with servers, storage, and networks, very little changes in terms of operations. IaaS is not difficult to use, but the reality is that considerable effort is required to take advantage of the utility computing benefits that IaaS can offer.

Enter Kubernetes. A characteristic of IaaS systems is that they secure only the necessary resources as and when needed, and users only incur costs for the resources used. This dovetails well with Kubernetes, which makes it easy to dynamically manage and thereby efficiently use resources and to take advantage of scalability. Not only does Kubernetes combine multiple nodes to ensure availability, it also makes it possible to automatically recover, without the need for human intervention, from all but systemwide faults by having the task of restoring faulty nodes delegated to Kubernetes (Figure 2).

An oft-heard explanation is that containers do away with the thick management layer conventionally constituted by hypervisors and virtual machines and thus allow for a thin, lightweight management layer with containerized processes running directly on a single OS. But the effect of containerization technology would be very limited if that were all there was too it. Containerization is indeed more efficient than virtual machines in many cases, but that is simply a matter of means. The real effects are only realized once multiple servers are bundled into a single large resource pool, with operations automated by delegating management of configuration information to Kubernetes. In fact, a lot of containers right now probably run on an IaaS system implemented as a virtual machine. Kubernetes can be thought of as a package imbued with best practices for making better use of IaaS technology.

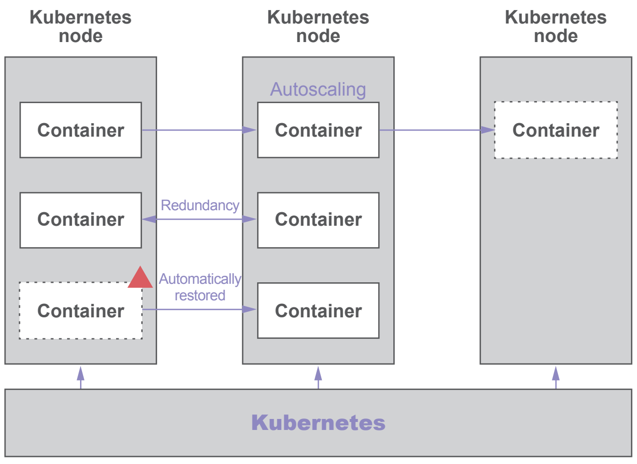

Figure 2: Kubernetes using IaaS

3.4 Realizing Hybrid Clouds with Kubernetes

While it is true that Kubernetes is a platform for better harnessing the advantages of IaaS, it's not the case that Kubernetes can only be used on IaaS. In fact, looking ahead, Kubernetes probably deserves more attention with respect to workloads for which an on-premise environment is key. We have been listening to our customers, and looking at cloud-related reports, for almost 10 years now. With the spread of cloud computing, although one can reason that a 100% on-prem setup is unrealistic, the response from the vast majority of customers has been that a 100% cloud setup would also be difficult to pull off. The fact that this is the majority opinion at a time when many engineers have become highly proficient with IaaS may offer some indication of what lies ahead.

What this means is that we need to think seriously about using hybrid clouds, which make use of both IaaS and on-prem environments as necessary. But, needless to say, this is no mean feat. Since IaaS is a closed system, it cannot be used in on-prem environments. Even if it were possible, however, operating a complex IaaS system in an on-prem environment could end up with you scratching your head and wondering what the whole point of using IaaS was in the first place. There certainly are some workloads for which an IaaS-equivalent setup in a private environment may be justified, but there are many tradeoffs to consider. However, wrapping both an IaaS and an on-prem environment with Kubernetes may be a realistic solution. Broadly, there are two challenges on the road to a hybrid cloud. These are the integrated management of IaaS and on-prem systems, and application and data portability (Figure 3).

You certainly want to keep the following within the on-prem environment: workloads and data that cannot be allocated to an IaaS system for compliance reasons, and workloads that use a fixed set of large-scale resources over an extended period with no increases or decreases in the amount of resources. And you may want to allocate all other workflows to an IaaS system. If it were clear, to an extent, which is the most appropriate from the get go, there would not be much to worry about, but it is not uncommon for the answer to the question of which is most appropriate to change over time, with, say, IaaS being preferred during a business's startup phase and on-prem being a better choice once the business has stabilized. If Kubernetes can solve the two previously mentioned challenges effectively, this market has the potential for sudden expansion.

At present, however, neither hybrid clouds that use Kubernetes nor on-prem environments that use Kubernetes can really be described as mature when compared with comparable IaaS systems. IaaS allows for software-based control of all resources via APIs, whereas on-prem environments

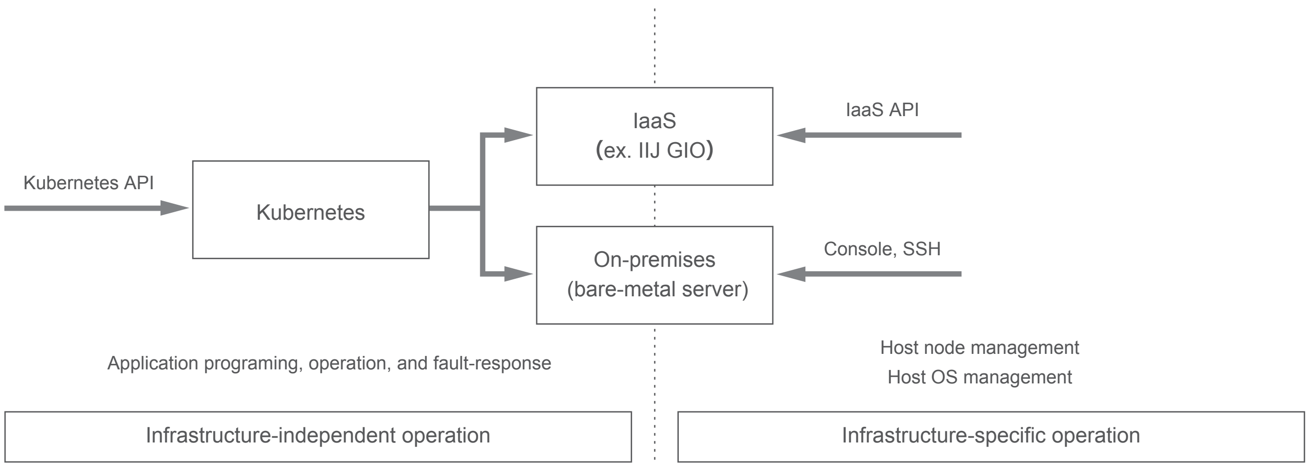
Kubernetes API
Kubernetes
IaaS (ex. IIJ GIO)
IaaS API
On-premises (bare-metal server)
Console, SSH
Application programing, operation, and fault-response
Host node management
Host OS management
Infrastructure-independent operation
Infrastructure-specific operation

Figure 3: Kubernetes Enabling a Hybrid Cloud

do not allow integrated software control, so compromise and cooperation between both Kubernetes and devices is necessary in order to build the environment. Also, once technologies such as SDN (software-defined networking) and SDS (software-defined storage) become easier to use in on-prem environments, this is likely to spur on the spread of Kubernetes in on-prem environments.

To realize a hybrid cloud, we need a common system for managing IaaS and on-prem. Kubernetes is without doubt a strong candidate for such a system.

3.5 IKE (IIJ Container Engine for Kubernetes)

Kubernetes has the future potential to dramatically improve efficiency by fundamentally changing the design and operation of server side systems and the distribution and provisioning of applications, and at IIJ, we have also developed and begun using a container cluster system. Named IKE (IIJ Container Engine for Kubernetes), this system was created to serve as a common platform for services and as an operating environment for our internal systems.

Packages like IKE that provide a container cluster environment around Kubernetes are called Kubernetes distributions. Above, I likened Kubernetes to an OS, but it is actually like an OS kernel. An OS cannot do much with a kernel alone. It needs the tools and device drivers provided by a distribution (e.g., RedHat or Ubuntu) before it can be of any use. Similarly, simply installing Kubernetes will not get you far. At a minimum, you need network drivers and storage drivers to suit your infrastructure; for a more pleasant container cluster experience, you need management tools to control Kubernetes; and a full support environment that facilitates the monitoring of applications deployed on Kubernetes, the display of alerts, the collections of logs, and so on is also indispensable. Packages that provide the ecosystem for setting up a Kubernetes cluster along with a mechanism for installing that environment as appropriate for the specified infrastructure are called Kubernetes distributions. IKE is one such distribution.

IKE was not developed for the purpose of providing services to customers, so it is only designed to work in a somewhat limited operating environment, but it can be installed on IIJ GIO, our cloud service, as well as in on-prem environments. We also plan to make it installable on other vendors' IaaS platforms and make it possible to provide a common environment on any sort of infrastructure.

IIJ has several reasons for implementing a Kubernetes distribution, and making use of IaaS is not the only objective. Our foremost objective is to speed up business, and our second objective is to enhance operational specialization so as to handle increasingly sophisticated and complex systems. This may all sound a little abstract and leave you with the impression that our objectives are vague, but what we ultimately aim to achieve is an environment in which, for example, teams responsible for developing services can concentrate on development while operations teams can focus on operations. If we can achieve this, I think we will naturally also fulfill those objectives (Figure 4).

In any case, containerization is a fascinating area. Simply enclosing each process in a container, rather than jamming a full server environment into a single container like with a virtual machine, makes it possible to realize infrastructure-independent server-side system distributions and all-purpose operation systems, giving rise to products that influence the entire IT industry. And this is probably only the beginning. It's exciting to think that more surprising and unexpected ideas still lie ahead.

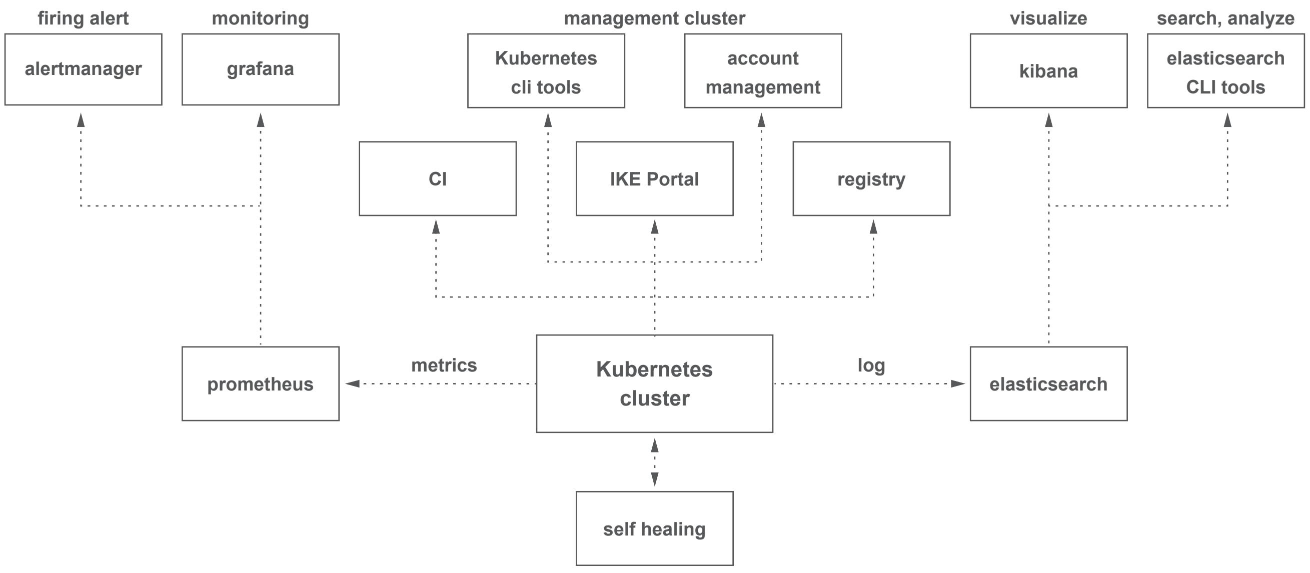

Figure 4: IKE

Keisuke Taguchi
Technology Strategy Office, Service Administration Division, IIJ.
As an engineer, I divide my working life equally between freelancing and IIJ. I started out as an applications engineer, but as soon as I encountered cloud computing, I ended up giving half of myself over to the infrastructure side of things.
I'm currently absorbed in the area of containerization technology, which I could go on and on about, so I'll stop here. Tags that apply to me: container, cloud, road bike, udon.