Technology Reports (Special Articles) Public Clouds Operations Failure
Special Articles on Use of Public Clouds

System Operations on Public Clouds to Provide against Large-scale Failures

Innovation Management Department Hiroki Moriya

Public clouds as typified by AWS are being increasingly used not only by private enterprises but also by many organizations and groups such as government agencies and educational institutions. In this way, public clouds are becoming a social infrastructure, so the shutdown of a public cloud can have a major impact on society overall. As a result, system operations that make use of clouds are becoming increasingly important. NTT DOCOMO has been using public clouds on a large scale for many years and has accumulated system-design and operation know-how assuming the possibility of a large-scale failure. It has also developed tools that enable smooth information sharing within the company in the event of a large-scale failure. It has consequently become possible for NTT DOCOMO overall to perform system operations that make provisions for the large-scale failure of a public cloud.

1. Introduction

It's already been more than ten years since public clouds including Amazon Web Services (AWS)*1 began to flourish, and since then, many organizations and groups such as government offices and educational institutions in addition to private enterprises have undertaken the construction and operation of systems that use public clouds*2.

Public clouds are no longer simple cloud services

*1 AWS: A cloud computing service provided by Amazon Web Services, Inc.
*2 Public clouds: Cloud computing services that anyone can use over the Internet.

as they have come to function as an infrastructure supporting society overall. Once a large-scale failure occurs in a public cloud, a situation arises in which that failure can have a major impact throughout society.

NTT DOCOMO has experience in the long-term use of large-scale public clouds and has used that experience to accumulate know-how on system operations that leverage the features of public clouds and to develop tools that support those operations. In this way, we have been active in extending the use of public clouds throughout the company. This article describes these activities with the aim of providing a reference in the use of public clouds even for organizations outside NTT DOCOMO.

2. Approach to System Failures

2.1 High-availability System Configurations

Failures are inherent to systems. A system that never breaks down or a perfect system that suffers no failures does not exist. This principle applies to both systems constructed on-premise[*3] and systems constructed on public clouds. It is therefore important when constructing a system to give it a configuration that can restore system-wide availability even if only slightly in the event that a failure occurs in an element making up the system. For example, in the case that a single piece of equipment such as a server stops functioning, a technique long used to handle such a problem is to maintain system functions by switching to backup equipment. Likewise, in the application domain, many measures have been implemented to enhance the fault tolerance of the entire system. These include the adoption of a loosely coupled system through asynchronous processing that limits the range of failure impact and the use of retry processing that assumes the occurrence of failures. Additionally, in contrast to configuring a system as a single monolithic[*4] service, the trend in recent years has been to construct a system by dividing it into multiple microservices[*5] and to improve availability of the entire system by limiting the range of impact when a failure occurs.

2.2 Shared Responsibility Model on Public Clouds

The "shared responsibility model [1]" has been widely adopted in the use of public clouds. This model clarifies the range of responsibility of both the cloud user and the public cloud operator and aims to secure the availability of the entire system through the activities of both parties. For example, in the case of virtual machines[*6], the range of responsibility of the public cloud operator is generally the area up to the virtualization layer including the data center, physical servers, and networks. The operator therefore implements measures to maintain availability such as incorporating redundancy in infrastructure[*7] components, physical equipment, networks, etc. On the other hand, it is the responsibility of the cloud user to implement a virtual machine OS and applications running on that OS in such a way that maintains availability. This example concerns the use of virtual machines, but cloud services other than Infrastructure as a Service (IaaS)[*8] have become popular in recent years

*3 On-premise: An environment in which a company owns, maintains, and operates the hardware making up its system.
*4 Monolithic: A configuration that provides multiple functions as a single and large software unit.
*5 Microservices: A system development technique that creates a single service by combining a number of small services.
*6 Virtual machines: Computers such as servers constructed in a virtual manner by software.
*7 Infrastructure: Generic term for physical or virtual data centers, servers, networks, etc. for executing applications.
*8 IaaS: A service that virtually lends out hardware such as servers and networks. The user sets and runs an OS and application software on the borrowed servers or network.

such as Platform as a Service (PaaS)*9 and Software as a Service (SaaS)*10. But the concept of the shared responsibility model is equally applicable to these services if only because the range of responsibility differs between the user and public cloud operator.

Many public cloud operators provide functions to support user designs that maintain system availability. They are also committed to disseminating information in the form of documents and white papers covering commonly used configuration patterns as best practices. With these functions, it has become possible for users to construct systems for efficiently achieving high availability. Moreover, since configurations that achieve high availability are often already in use by public cloud operators for cloud services like PaaS and SaaS, users have come to favor such services for constructing their systems.

3. Recent Large-scale Failures on the Cloud Operator Side

The following provides some examples of large-scale failures in public clouds that have recently occurred.

1) AWS Failure

At AWS, on August 23, 2019, a failure in the control system for air conditioning equipment prevented the cooling system from operating normally causing some servers to overheat. This, in turn, caused failures to occur in a single Availability Zone*11 and specifically in Amazon Elastic Compute Cloud (EC2)*12 and Amazon Elastic Block Store (EBS)*13. Additionally, as EC2 and EBS are used as platform services for configuring other services, it was found that Amazon Relational Database Service (RDS)*14, Amazon Redshift*15, Amazon ElastiCache*16, Amazon WorkSpaces*17, and other managed services*18 were also affected. According to some cloud users, there were some cases in which adopting a redundant configuration across multiple Availability Zones beforehand was able to minimize the impact of this failure that occurred in a single Availability Zone, but some effects were nevertheless reported for some configuration settings.

Then, on April 20, 2020, failures such as an increase in processing errors and delays also occurred in the Tokyo Region in managed services such as Amazon Simple Queue Service (SQS)*19 and AWS Lambda*20. It was confirmed that users of these services were impacted by these failures [2].

2) Microsoft Azure Failure

At Microsoft Azure*21, authentication errors in Azure Active Directory (Azure AD) were observed around the world on September 29, 2020. Azure AD is a core service of Microsoft Azure for managing authentication and permissions in many services for developer and user access and inter-service operations. It was found that this failure impacted users by preventing them from using some services or applications such as Microsoft Azure and Office 365. As for the cause of this failure, it was announced that an update to an internal validation test was supposed to undergo deployment*22 only after multilayer testing. However, this update actually bypassed the testing process owing to a latent bug in the Safe Deployment Process (SDP)

*9 PaaS: A service that lends out a platform including an OS and middleware for running applications on the cloud. The user creates and uses application software on the borrowed platform.
*10 SaaS: A service that lends out applications on the cloud. The user can start using those applications immediately.
*11 Availability Zone: A single or group of data centers. Individual Availability Zones are physically independent of each other.
*12 EC2: A type of IaaS offered by AWS providing virtual machines.
*13 EBS: A type of IaaS offered by AWS providing block storage.
*14 RDS: A type of PaaS offered by AWS providing relational database functions.
*15 Amazon Redshift: A type of PaaS offered by AWS providing a data warehouse.
*16 Amazon ElastiCache: A type of PaaS offered by AWS providing in-memory cache.
*17 Amazon WorkSpaces: A type of SaaS offered by AWS providing Windows or Linux desktop environments.

system*23 and was directly deployed into the production environment as a result [3].

3) GCP Failure

A failure occurred on Google Cloud Platform (GCP)*24 on March 27, 2020 in Cloud Identity and Access Management (Cloud IAM), an access control and management service. Cloud IAM is a service used in common by many GCP services for access control, so this failure had a large-scale impact on other services. The official announcement stated that the effects of the failure lasted for a total of 14 hours. Its cause, as announced, was that cache servers within Cloud IAM ran out of memory due to an unexpected high number of modification requests made to the service, which caused those requests to time out [4].

As reflected by the above incidents, unexpected failures impacting users have even occurred in services provided by cloud operators with a proven track record on a global level—there will never be a total absence of failures.

4. Things for Users to Consider

Things that users should consider in relation to the occurrence of system failures in public clouds are summarized below.

4.1 Assume the Occurrence of Failures

Throughout the world, failures occur even in public clouds with a proven track record and wide usage—failures will never be completely eliminated even with the further evolution of public clouds. It is extremely important that users keep this in mind. Using a public cloud without facing this reality makes it difficult to design and operate a system that assumes the occurrence of failures, which can lead to major losses. It is imperative from the start that business owners and management in addition to development and operation managers understand this fact.

4.2 Apply Best Practices

As described above, it's impossible to operate a system with zero failures, but it is possible to reduce the impact of failures. Most public cloud operators provide users with recommendations on system designs for maintaining availability and reliability while simultaneously providing a variety of functions and options to make it easy for users to implement those designs. It is important that users use these functions to implement a design that will enable the system to continue operating as much as possible even if a failure should occur in the public cloud. Fortunately, there are already many users of public clouds throughout the world, and as a result, design patterns that have been adopted in all sorts of use cases have been released as best practices. For this reason, there is no need for the user to design from scratch since adopting those best practices in design and operation can reduce risk.

In addition, functions like PaaS and SaaS have grown in recent years, and services and platforms designed beforehand by public cloud operators based on best practices have come to be energetically adopted by users. This trend in constructing systems with high failure resistance is now gaining

*18 Managed services: Cloud services whose resource provisioning, operation, etc. are mostly the responsibility of the cloud operator. Among cloud computing services, these refer to PaaS and SaaS, for example.
*19 SQS: A type of PaaS offered by AWS providing a message queuing function.
*20 AWS Lambda: A type of FaaS offered by AWS providing an execution environment for application code. The user can execute an application by registering created source code.
*21 Microsoft Azure: A cloud computing service provided by Microsoft Corporation.
*22 Deployment: Installing applications by placing them in their execution environments.
*23 SDP system: A system used by Microsoft to manage the process of safely deploying software.
*24 GCP: A cloud computing service provided by Google LLC.

momentum. The "serverless" design pattern is a good example of this trend. It is a technique for designing and operating systems in which the user need not be concerned about physical servers or even virtual servers typical of IaaS. This article, however, omits details on serverless computing. Using such design techniques and services is advantageous not only in strengthening failure resistance but also in reducing the burden placed on users in system operation. There is no doubt that this trend will accelerate in the years to come.

4.3 Design Operations to Provide against Failures

Even if risk can be reduced through some set of measures, it will still be impossible to completely eliminate failures or their impact. Moreover, while it may be possible to develop measures that can sufficiently deal with failures that can be envisioned beforehand, there are many failures that originate in unexpected cases or events. It is therefore important to quickly identify the cause of a failure at the time of its occurrence and to design operations to enable a speedy recovery to be made. Several specific points in this regard are given below.

1) Enhance System Observability

System monitoring is extremely important even when using public clouds. The composition of an infrastructure particularly when using public clouds often spans IaaS, PaaS, and SaaS, and in the case of applications, the use of containers[*25] and microservices is increasing. As a result, systems that are distributed over many types of environments are coming to be deployed, which is making it more important than ever to enhance system observability. This does not simply mean life/death monitoring. Rather, it means the adoption of distributed tracing that tracks individual requests[*26] processed by applications and processes at the method[*27] level and of a platform service that can consolidate logs and tracking results in an integrated manner and visualize and analyze this information. These mechanisms come in various forms and may be provided by cloud operators or third party[*28] services or may be released as Open Source Software (OSS)[*29], so it is necessary to select and use the ones that fit the user's current objectives.

2) Lower the Risk of Unexpected Failures

Minimizing the risk of unexpected failures as much as possible is also important for improving operations. Making it a practice of routinely performing failover[*30] tests for the system on the cloud and recovery tests to provide against an outage of a specific cloud service is effective in lowering the risk of unexpected failures.

The following introduces chaos engineering as a technique for reducing the risk of failures characteristic of public clouds. Chaos engineering intentionally causes failures to occur in the system's production environment and measures their impact on the entire system. In this way, it becomes possible to observe whether effects that are not expected to be system wide at the time of a failure behave as such and to therefore improve the system's failure resistance. This technique is ideal for a cloud that can manage an infrastructure via an Application Programming Interface (API) and restructure it in any number of ways. Netflix in

*25 Containers: As one type of computer virtualization technology, a method for creating a dedicated area called a container on one host OS and running necessary application software within that container.

*26 Requests: Operation requests made to an application.

*27 Method: An HTTP method such as GET, POST, PUT, and DELETE.

*28 Third party: Refers to a third party manufacturer or developer.

*29 OSS: Software whose source code is released free of charge for anyone to reuse or modify.

*30 Failover: A mechanism for automatically switching over to a redundant standby system when a failure occurs in the main system.

the United States has put into practice such operation on a daily basis to lower the risk of suffering effects from unexpected failures [5]. Additionally, since chaos engineering is applied in a production environment (actual operation environment), it is not a technique to be executed blindly. Rather, it should be applied after putting the documented system design into practice and after sufficient confidence in the failure resistance of a system has been obtained. This point requires special attention.

3) Understand the Configuration of Individual Cloud Services

Of unexpected importance when using a public cloud is the need to understand the configuration of individual cloud services. Of course, details on the inner configuration of a cloud service in relation to security and compliance are not released, and depending on the cloud service, neither are details on the locations themselves of data centers. On the other hand, there are services whose logical configuration at least in part is released so that the user can take that information into account at the time of system design and operation. Obtaining a good understanding of that configuration can aid in identifying the cause of a failure and achieving a smooth recovery.

For example, in the case of AWS, a group of data centers is called an Availability Zone that is physically independent from any other Availability Zone. Understanding this concept in itself is essential to high-availability design. Additionally, in terms of PaaS/SaaS, it is often the case that a service will be rolled out to users after it is configured by AWS itself across multiple Availability Zones from the same perspective as a user. Understanding this makes it possible to grasp whether the occurrence of a failure in a particular Availability Zone will impact individual cloud services. Moreover, on the user side, if an option is provided for cutting off traffic to a specific Availability Zone at the time of a failure, deciding to do so can minimize the impact of that failure. In addition, the EC2 service providing virtual servers has become the foundation for many PaaS/SaaS services, and understanding this makes it possible to predict whether a failure in EC2 "holds the possibility of impacting other services" and to at least stand ready for such an outcome. In addition to the above, there are cases in which the configurations of certain public cloud services are actually released, so responses to the occurrence of failures can be smoothly put into practice the more that these configurations are understood.

5. Support Visualizer Development and Provision

Finally, this section describes a measure that NTT DOCOMO has put into practice in the event of a failure when using AWS, the most used cloud service within the company. NTT DOCOMO manages more than 900 AWS accounts (as of December 2020) that use AWS under various workloads*31. In 2019, however, a failure in the AWS Tokyo Region had not a small impact on NTT DOCOMO and a number of problems came to light as a result. As a measure taken to solve these problems, we constructed a system called Support Visualizer

*31 Workload: An indicator of the size of a system's load, such as the CPU utilization rate. In particular, in a public cloud environment, the workload may represent the system itself, including the OS and application code running on the cloud. In this article, we use the term in this latter sense.

that consolidates information on AWS support cases (described later).

5.1 Problems Identified from AWS Large-scale Failure

As described above, NTT DOCOMO uses AWS under a variety of workloads. Needless to say, system requirements as well as the number of application users, data traffic, etc. differ depending on the workload, so it is not rare for system design and operation to be system dependent and for different types of cloud services to be used. Consequently, if a failure should occur in a cloud service, the extent to which the effects of that failure will be felt will depend on the system. Up to now, there have been systems that suffered no effects at all from the occurrence of a large-scale failure as well as systems in which effects were felt by cloud service users and application users. The following problems came to light with respect to the management of individual systems and the company overall during the occurrence of such failures.

1) Collecting Information from the Entire Company

The impact of a failure on individual systems is not necessarily the same, so it is incumbent on the company to determine without delay the extent of that impact on individual systems and to make appropriate announcements. In actuality, however, consolidating information quickly is difficult given that the operation of each system is independent and that on-site personnel at the time of a failure are busy trying to track down the cause and perform recovery operations. Of course, speedy restoration of the system and minimizing the impact on users should be given top priority, but amid all this, it is also important that the company disseminate appropriate information as their social responsibility. Up to now, it's been very difficult to satisfy both of these needs at the time of a large-scale failure, so the problem of collecting information from the entire company remained.

2) Identifying and Dealing with the Range of Impact

Problems in individual systems also came to light. Since system failures themselves are not limited to the use of public clouds and can occur at any time, a large-scale failure will likely be dealt with in the same way as a system failure. At this time, however, it is not known whether the problem is caused by a failure in the cloud itself or is peculiar to that individual system, so identifying the cause and dealing with it appropriately takes time.

In relation to this phenomenon, most public cloud operators release a service status, so checking on that status at the time of a failure can provide information on that failure and its approximate range of impact. On the other hand, our experience with cloud failures to date has revealed that such service status data does not necessarily list all failures that have occurred. In actuality, there are cases in which a failure becomes known from reports submitted by cloud users and cases in which the cloud operator reports a failure after resolving it.

5.2 Support Visualizer

In AWS, if a user's system happens to be affected by a failure in a service provided by AWS itself, the user may issue a report to AWS in the form of an inquiry ticket[*32] called a "support case."

*32 Inquiry ticket: A unit for managing individual inquiries and their replies.

This operation makes it possible for AWS on its side to collect information on a failure and to eventually isolate its range of impact.

As a user, however, NTT DOCOMO manages each AWS account independently and cannot, as a result, grasp the impact of a failure on other projects on the basis of a support case. There is therefore a need for projects to exchange information directly with each other. However, as described above, on-site personnel are working as hard as possible to recover their system at the time of a failure, so the reality is that they have little time to exchange information. The same can be said of consolidating information from the entire company. In response to this problem, we developed Support Visualizer as a system that consolidates information on support cases issued within the company and that anyone in the company can use to peruse that information.

1) Architecture

The architecture of the Support Visualizer system is shown in **Figure 1**. Although support-case information in AWS accounts is independent of each other, AWS provides an API that can obtain that information. Support Visualizer uses this API to consolidate support-case information, which is indexed and stored in a database to enable it to be searched and analyzed by Amazon's Elasticsearch service. Here, the Kibana*33 dashboard*34 can be used by the Cloud Center of Excellence (CCoE)*35 to peruse and analyze this consolidated information, and in the event that a highly urgent support case

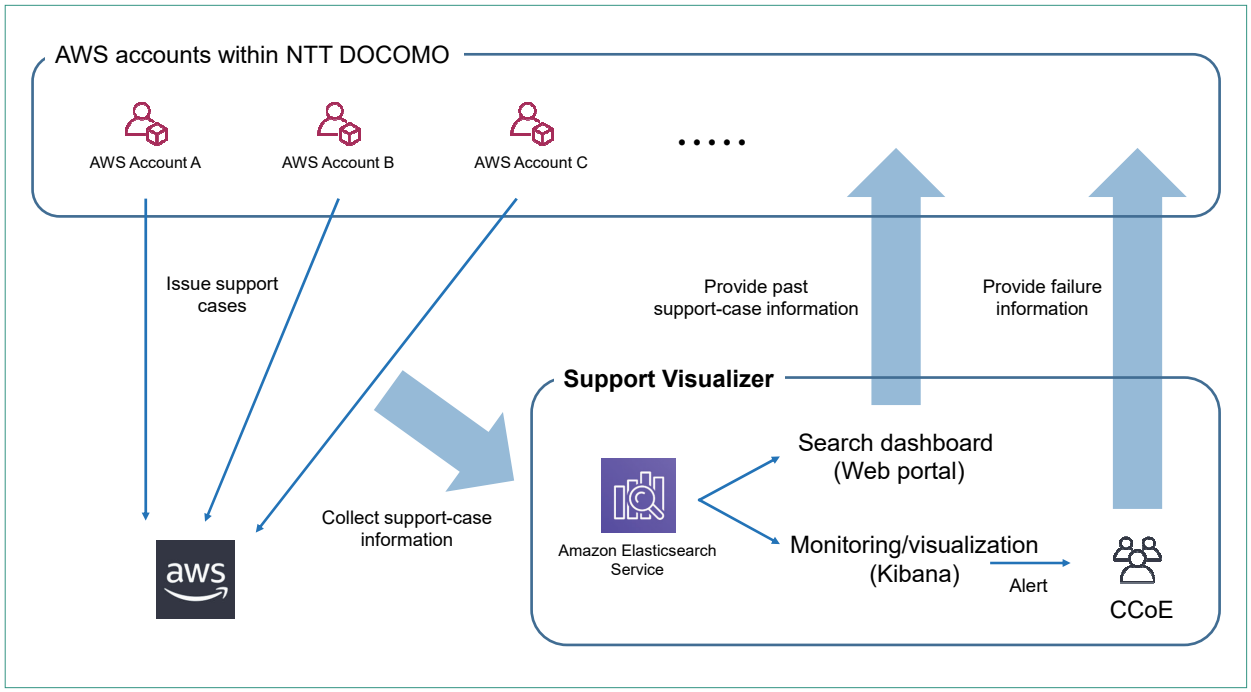

Figure 1 Architecture of Support Visualizer

*33 Kibana: An open-source data visualization tool developed by Elastic.
*34 Dashboard: A screen that consolidates information.
*35 CCoE: An exclusive team within an enterprise that establishes best practices and creates essential systems and governance to make cloud usage successful.

comes to be issued, the CCoE can be notified using Slack*36 or a similar tool.

We have also deployed a portal to enable users within the company to peruse this in-house consolidated support-case information and to filter that information by keywords or services, degree of urgency, update time of support-case information, etc. (Figure 2).

2) Problems for which Solutions Can Be Expected

Given the occurrence of a system failure on the cloud and the issuing of support cases from separate systems, this deployment of Support Visualizer enables even other users within the company to check the content of those support cases. This is especially effective at the time of a large-scale failure on the cloud—even if system operators are busy responding to the failures in their own systems, support cases issued by those systems will still be consolidated automatically. In this way, the company can easily obtain an overall view of the extent to which a failure is impacting its systems. At the same time, operators of individual systems can determine whether the effects on their systems are separate phenomena or a phenomenon occurring on the cloud overall. Additionally, if a similar phenomenon has already occurred on other systems,

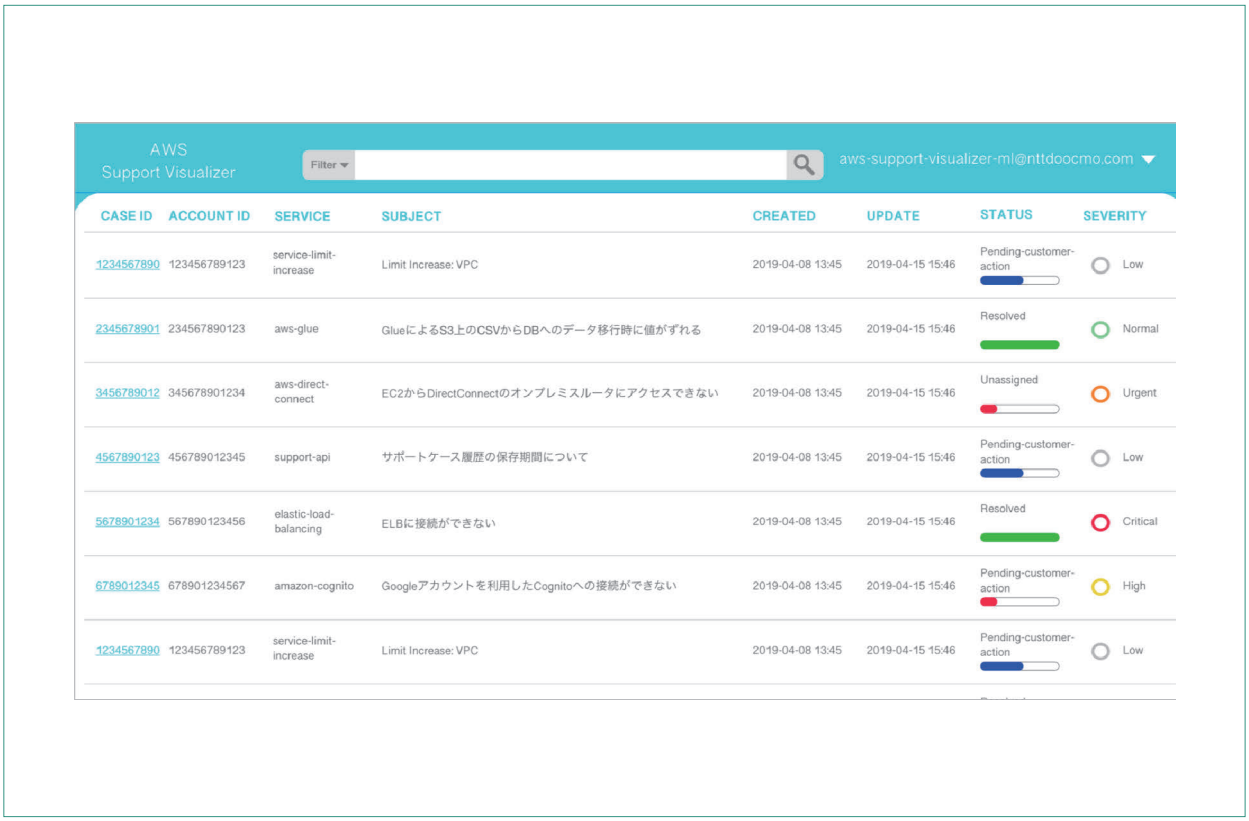

Figure 2 Screenshot of dashboard for Web portal of Support Visualizer

*36 Slack: A business chat tool provided by Slack Technologies, Inc.

it may be possible to check what measures were taken to solve that problem. NTT DOCOMO manages more than 900 AWS accounts of various workloads, so we can expect Support Visualizer to enable users to grasp the impact of a failure at actual sites, which is something that cannot be obtained only on the basis of status information released by the cloud operator.

3) Side Benefit

A side benefit of Support Visualizer is that technical inquiries on development and operation in everyday use of a cloud come to be consolidated, which means that technical know-how on using a cloud from within the company can also be consolidated. Going forward, we aim to incorporate a mechanism for analyzing trends in consolidated support cases and actively consolidating know-how from the results of that analysis. In this way, we can feedback analysis results to each system and promote more efficient cloud usage.

6. Conclusion

This article described system operations that anticipate the occurrence of large-scale failures in public clouds. Despite the ongoing evolution of technologies and architecture such as clouds, containers, and microservices, it is impossible to construct a system with zero failures. We will continue in our efforts to minimize the impact of failures as much as possible and to cultivate best practices so that NTT DOCOMO can make productive use of public clouds throughout the company.

REFERENCES

[1] Ministry of Internal Affairs and Communications: "Cloud Characteristics and Security," (In Japanese). https://www.soumu.go.jp/ict_skill/pdf/ict_skill_2_3.pdf

[2] AWS: "Summary of the Amazon EC2 and Amazon EBS Service Event in the Tokyo (AP-NORTHEAST-1) Region," Aug. 2019. https://aws.amazon.com/jp/message/56489/

[3] Microsoft Azure: "RCA – Authentication errors across multiple Microsoft services and Azure Active Directory integrated applications (Tracking ID SM79-F88)," Sep. 2020. https://status.azure.com/status/history/

[4] Google Cloud: "Google Cloud Infrastructure Components Incident #20003." https://status.cloud.google.com/incident/zall/20003

[5] Netflix: "The Netflix Simian Army," Medium, Jul. 2011. https://netflixtechblog.com/the-netflix-simian-army-16e57fbab116