What are AWS Cloud Design Patterns?

The AWS Cloud Design Patterns (CDP) are a collection of solutions and design ideas for using AWS cloud technology to solve common systems design problems. To create the CDPs, we reviewed many designs created by various cloud architects, categorized them by the type of problem they addressed, and then created generic design patterns based on those specific solutions. Some of these problems could also be addressed using traditional data-center technology, but we have included cloud solutions for these problems because of the lower cost and greater flexibility of a cloud-based solution.

Basic Patterns

Snapshot Pattern

Problem to Be Solved

More than anything else, it is important that your data is safe. This means that it is important for you to back up your data. For example, you may use a tape to backup data to a separate location. However, with a tape backup, you have the cost of changing and storing tapes, for example, and these are operations that are difficult to automate. While you can buy expensive equipment to provide some degree of automation, even when this is done, you are still faced with the fact that tapes is inherently have limited capacity, and complete automation is difficult.

Explanation of the Cloud Solution/Pattern

The AWS Cloud lets you use the limitless capacity of "Internet storage" (also known as "Web Storage"), doing so safely and with relatively little expense.

In the AWS Cloud, we often talk about a "snapshot," which is a backup copy of your data at a given point in time. The AWS Cloud lets you copy the data of a virtual server (including the operating system), along with other data, to Internet storage easily, making it easy for you to take these snapshots at regular intervals. You can take a snapshot with one click of a button in the Control Screen, or you can take a snapshot using an API. That is, you can automate it using a program. Because with Internet Storage you don't have to worry about capacity, you can automate the backup process by having a computer program take snapshots on a regular periodic basis.

When performing program update checks or creating temporary test environments, you need to create the environment using a specific data cross-section. In this case, you need to copy not just the data, but the OS as well. A snapshot is a perfect solution for this as well, because it copies each individual OS.

Implementation

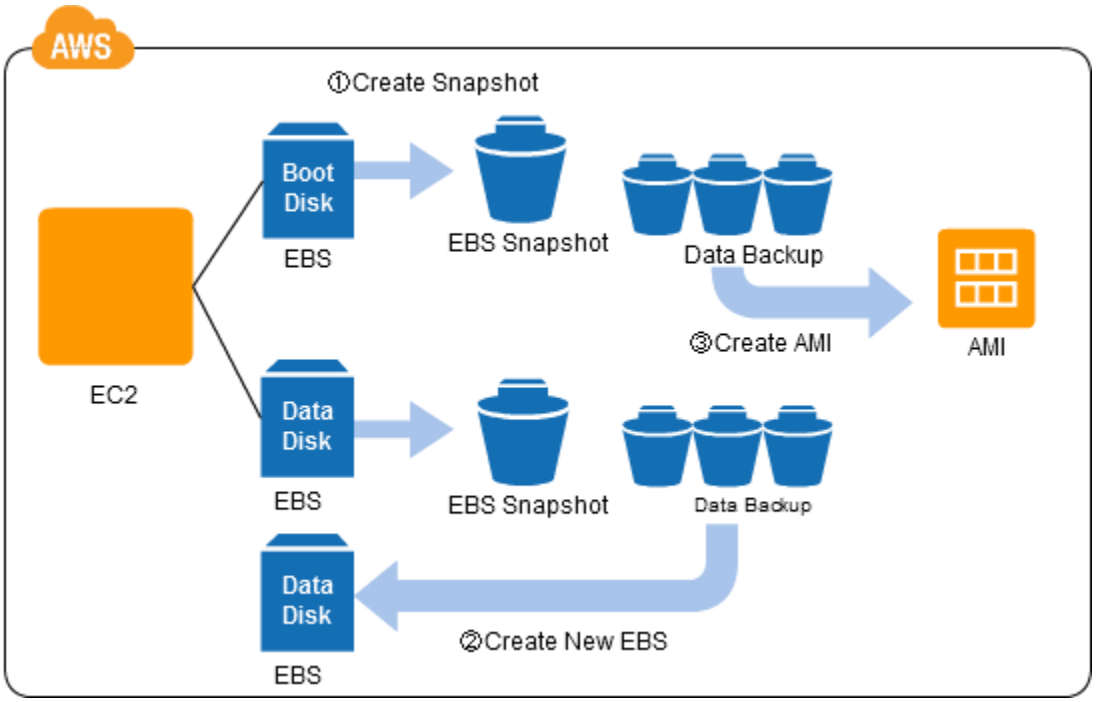
The Elastic Block Store (EBS), which is the virtual storage in AWS, has a snapshot function. Make a snapshot using this function. When you take a snapshot, it is stored in the Amazon Simple Storage Service (S3) object storage, which is designed to have 99.999999999% durability.

When the EBS snapshot function is used, all of the data required to recreate the EBS volume is copied to S3. A snapshot that has been stored in S3 can be recovered as a new EBS volume. Even if data is lost or corrupted in the EBS, you can recover the data from the time when the snapshot was taken.

When you use EBS as a data disk, you can backup your data at any time by taking a snapshot. You can take as many new backups as necessary, whenever necessary, without having to worry about storage capacity.

When you use EBS as a boot disk, you can make a copy for each operating system, and store it as an Amazon Machine Image (AMI). You can launch a new EC2 instance from that data.

Configuration



Benefits

- Taking backups can be controlled by a computer program. That is, you can automate the process, rather than having to make backups manually.
- You can use S3, which has high durability, as the backup destination.
- You can backup up all of the data on the EBS volume as a snapshot enabling immediate use of the snapshot to create a new EBS volume. This makes recovery easy in the event of a failure.

- You can make backups not just of user data, but of each individual operating system as well. Because the backup for each OS is stored as an AMI, you can launch new EC2 instances.
- You can take data under specific circumstances (for example, after replacing an application or after updating data), and can take multiple generations, without having to worry about storage capacity. This lets you rebuild the environment easily after a failure or a problem, enabling you to return to the environment of any given point in time.

Cautions

- You must maintain data consistency when taking snapshots. When you take a snapshot with the EBS volume mounted, make sure to take the snapshot in a state where logical consistency has been achieved, for example, after flushing the cache of the file system (EXT or NTFS), and after application transactions have been completed.
- Typically, the smaller the data size of the boot disk, the more rapidly the virtual server can be started. Note that disk checks that are performed periodically (fsck in Linux) also take time.

Other

- You may want to split the boot partition and the data partition into separate EBS volumes when making a backup, because you will probably want to backup your data parts more often than your boot parts.

Stamp Pattern

Problem to Be Solved

Setting up the OS and the applications required for a virtual server takes just as much work, time, and expense as for a physical server.

A virtual server is a server that does not depend on hardware, that is, it is a server in software. In practice, multiple virtual machines are emulated on a physical machine. Without any major changes to the physical machine, the virtual machine can function as if it were a physical machine. This is very convenient -- you can create or delete virtual machines immediately whenever the need arises.

Although opportunities for using virtual machines are increasing more than for physical machines, the setup required to use a virtual server (for example, setting up the OS and installing and setting up applications) still takes as much labor, time, and expense as for a virtual server.

Explanation of the Cloud Solution/Pattern

You can use the AWS Cloud to create of a machine image in a state where the operating system, middleware, and applications have already been set up on a virtual server, and use that image to launch a new virtual server. That is, once you have set up the operating system, middleware, and applications, you can copy them for use at any time. Copying the virtual servers like using a rubber stamp lets you generate virtual servers in large quantities,with the environments already set up and ready to go.

While in the past you would have had to do this by taking the physical disk where the environment has been set up and copying it to another disk, or by using separate backup software to copy from disk to disk, this would have been time-consuming and expensive, making it difficult to perform such copying quickly and at high volumes. When using the AWS Cloud, on the other hand, resources such as the servers, disks, and the like, are handled logically, so you can perform such operations easily.

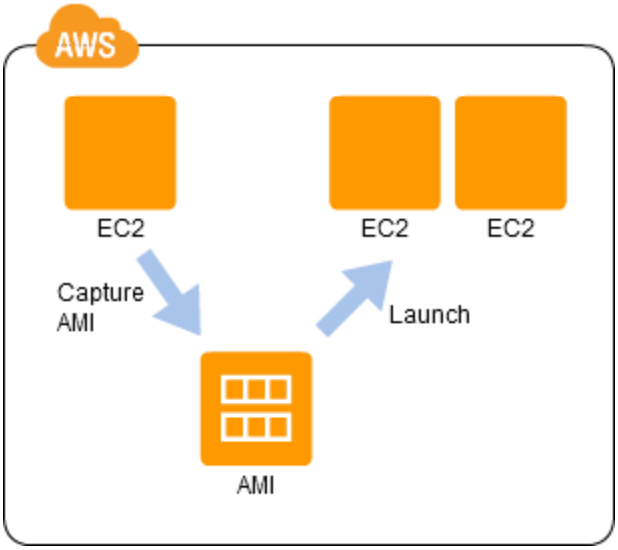
Implementation

If you create an Amazon Machine Image (AMI) from an Elastic Block Store (EBS) that includes the boot region of the operating system, you will be able to launch an EC2 instance from the AMI, making it possible for you to produce a large numbers of EC2 instances with identical settings.

(Procedure)

- Launch an EC2 instance and install the required software.
- Perform the necessary set up and create a state where it is running as a server.
- Capture and save the AMI after confirming proper operation.
- Use that AMI to create the required numbers of servers, when required.

Configuration



Benefits

- Using an AMI where the environment has already been constructed frees you from the operations required for setting up the EC2 instances launched based on that AMI.
- You can launch hundreds of EC2 instances with identical operating systems, data, and settings.
- Even when using a script to launch EC2 instances, set up operations are unnecessary if the environment has already been constructed for the AMI, making it possible to simplify the script.
- Not only can you use the AMIs that have been constructed, but you can share the AMIs with specified users, and, if necessary, you can publish the AMIs.

Cautions

- The point in time at which the snapshot should be taken and the timing with which the AMIs should be updated have to be handled on a case-by-case basis . The way to do so will depend on the system requirements.
- EC2 instances (virtual servers) that are exactly identical are replicated, so when there are items requiring changes in settings for individual virtual servers there is a need for some way to do so.
- Once you create an AMI, any patches or revisions to the base operating system will not be implemented automatically in the AMI. You will have to perform patching and upgrading operations on the individual AMIs.

Other

This operates the same way as the "Bootstrap Pattern" of the Operation & Maintenance Pattern. Because you can construct the operating system and middleware more flexibly in the Bootstrap Pattern than in the Stamp Pattern, you have to consider the trade-offs when determining which of the two patterns to use.

Scale Up Pattern

Problem to Be Solved

Typically it is difficult to estimate, during the system development phase, the server resources that will be required after deployment.

If the server resources are insufficient after deployment, the server will not perform well, or will be unable to keep up with batch processes. On the other hand, excessive server resources require excessive investments and mayactually produce losses.

While it would be nice to be able to adjust server resources after system deployment, this is difficult because the server resources depend on the specification of the physical machine.

Explanation of the Cloud Solution/Pattern

The AWS Cloud lets you switch the virtual server specifications (CPU, memory size, and the like) as needed. You can adjust the specifications even after launching of the virtual server.

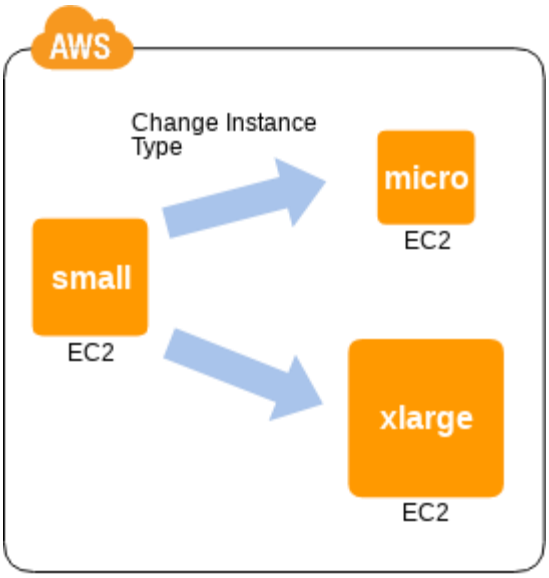
While in the past you would have to replace the physical server and reinstall the operating system if resources prove inadequate after deployment, there is no such need to do so in the AWS Cloud. You can deploy a system by launching a virtual server, and adjust the server specifications while monitoring the resource usage.

Implementation

(Procedure)

- Launch an EC2 instance and build the system.
- Monitorthe resource utilization using vmstat or a resource monitor, CloudWatch, or the like, and if the specification is inadequate (or excessive), stop the EC2 instance, and then restart after adjusting the instance type using the Change Instance Type menu of the AWS Management Console.

Configuration



Benefits

- This eliminates the need for precise estimation of server specifications at the time of system design/development.
- This reduces opportunity costs due to system stoppages and the inability to provide services to customers caused by inadequate resources.
- This enables reduction in waste, in terms of expenses, because you can switch to a reduced specification if you discover that the resources are excessive.

Cautions

- When you need to adjust a server specification, you will need to stop the EC2 instance temporarily. The system will go off-line at this time for a period of time from about 30 seconds up to several minutes (depending on the disk space and the setup of the server).
- Despite being able to adjust the server specifications, iyou are still limited by the upper limit for the instance type. As a result, if the resources are inadequate despite selecting the instance type with the maximum processing performance, you will have to consider the use of Scale-out Pattern, caching, substituting another service of AWS, for example.

Other

- If you can predict the processing peak, then you can also adjust the server specification automatically to match. For example, if closing processes with high overhead are performed at the end of the month, then you can put together a

schedule where the specification is adjusted upward at that time, and adjusted downward otherwise.

- You can handle Frequent processing delays through temporarily modifying the server specification, and then returning the server specification to the original after making improvements to performance of the application or database. This is used particularly often in areas such as consumer-oriented servers where the access frequency cannot be read, and in database servers, for example, which are difficult to scale-out.

Scale Out Pattern

Problem to Be Solved

To process high traffic volumes, you will need a high-specification server for the web server. The approach where a machine with a higher specification is used to increase the processing performance is known as "Scaling Up." However, this approach has problems. Typically, the higher the specification in a high-specification server, the higher the unit processing cost. Understand also that there is a limit to the server specifications – the specifications cannot be boundlessly high.

Explanation of the Cloud Solution/Pattern

The approach where multiple servers of identical specifications are provided in parallel to handle high traffic volumes is known as "Scale Out."

You launch multiple virtual servers, and use the Load Balancer to distribute the load to the individual virtual servers. Depending on your system, the traffic may increase drastically over weeks, days, or sometimes even hours. The AWS Cloud makes it easy for you to change dynamically the number of virtual servers that handle processing, making it possible to match such dramatic variations in traffic volume.

Implementation

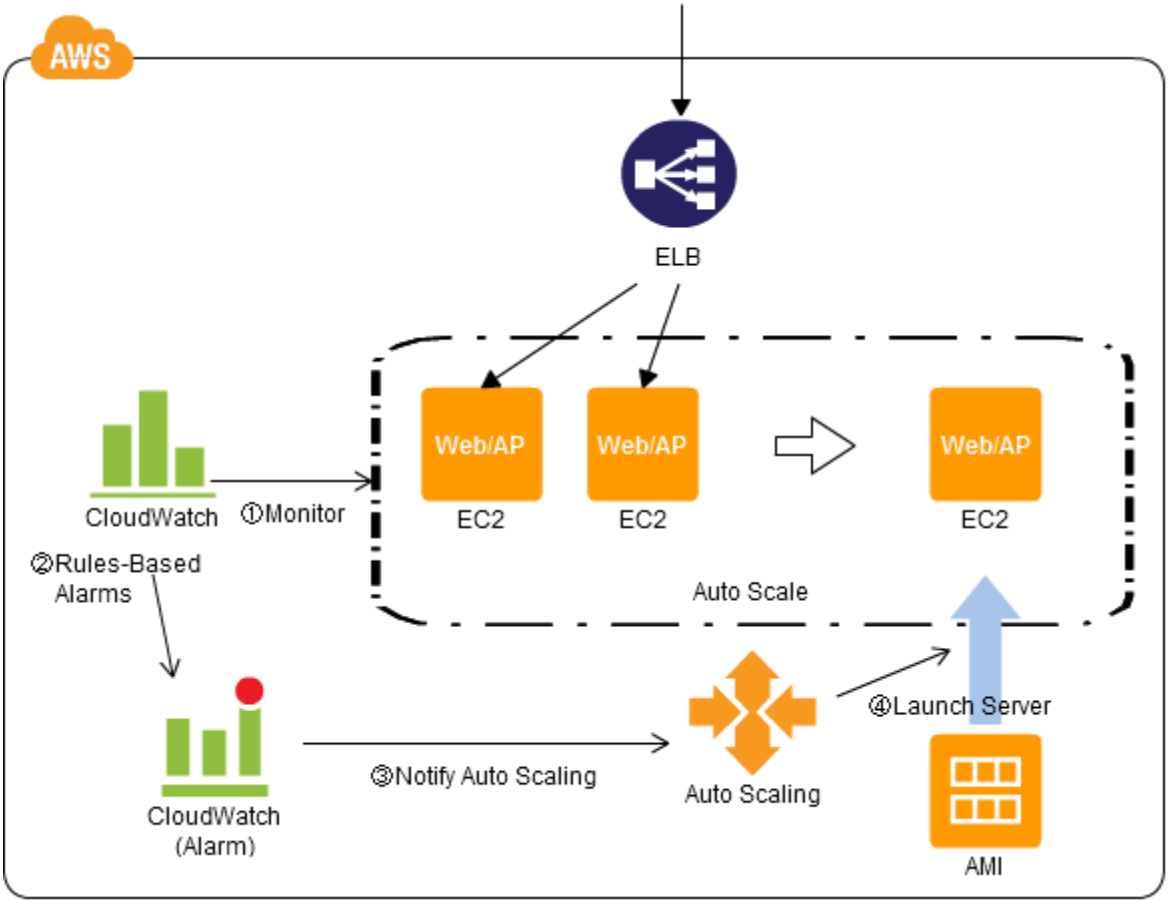
You can use a combination of three services: the load balancing service (Elastic Load Balancing (ELB)), the monitoring tool (CloudWatch), and the automatic scale-out service (Auto Scaling), to structure easily a system that scales up automatically depending on the load.

(Procedure)

- Set up multiple EC2 instances in parallel (as web/AP servers) under the control of ELB.
- Create an Amazon Machine Image (AMI) to be used when starting up a new EC2 instance.
- Define the conditions (metrics) to trigger an increase or decrease in the number of EC2 instances. The average CPU use rate of an EC2 instance, the amount of network traffic, the number of sessions, the Elastic Block Store (EBS) latency, and the like are often used.
- Use CloudWatch to monitor these metrics, and set up to issue an alarm if specific conditions are satisfied.

- Set up so that Auto Scaling will increase or decrease the number of EC2 instances when an alarm is received. <ref group="Related Blog"> Template:Related Blog </ref> * Completing the set up as described above makes it possible for you to, for example, start up two EC2 instances using an AMI that has been prepared in advance when the average CPU use rate has been greater than 70% for at least five minutes continuously. Of course, you can reduce the number of servers to match the circumstances.

Configuration



Benefits

- This provides service continuity because the number of EC2 instances is increased automatically in response to an increase in traffic.
- This allows you to reduce costs, because the number of EC2 instances is reduced when there is little traffic.

- This reduces your workload, as an administrator, because the number of EC2 instances is increased or decreased automatically in response to changes in traffic level.
- When compared to scale-up, the limit on processing capability is extremely high because the required number of EC2 instances can be provided in parallel, under the control of the ELB.

Cautions

- This cannot handle severe or rapid variations in traffic such as the traffic doubling or tripling over several minutes. This is because, in practice, the actual increase in the number of EC2 instances takes some time after the need to do so is identified. In this case, you should schedule an increase in the number of EC2 instances at specific times. Provide enough extra EC2 instances in advance to be able to handle the load, and then delete the unneeded EC2 instances.
- Consider whether to leave HTTP session management, SSL processes, and the like, to the ELB, or whether to process these in controlled servers.
- Because the ELB is not provided with a system for changing the amount of load distribution depending on the specifications, the controlled EC2 instance types should all be identical.
- When compared to scale-out on the web/AP server layer, typically scale-out on the DB server layer is more difficult. See the Relational Database Pattern.
- To increase the fault tolerance, you should distribute the scale-out across multiple Availability Zones (AZs). See the Multi-Datacenter Pattern. At this time, the incremental number of instances should be a multiple of the number of AZs to enable equal distribution to each of the AZs.
- When making the first SSH connection to a newly-launched EC2 instance, in some cases a fingerprint validation will be performed interactively to validate the host at the time of logging in, in which case functions such as automatic processing using SSH from the outside will be disabled.
- When an update is required within an EC2 instance, you will need to also update the Amazon Machine Image (AMI) that was the source for launching in Auto Scaling.

Other

- See the Clone Server Pattern, NFS Sharing Pattern, and NFS Replica Pattern regarding file sharing when performing Scaling Up.

- See the State Sharing Pattern in regard to session administration. See the Scheduled Scaleout Pattern.

Ondemand Disk Pattern

Problem to Be Solved

It is difficult to forecast in advance the disk capacity that will be used by the system. As a result, usually, at the time of system deployment, the designer provides a disk capacity that envisions what the need will be several years out, multiplied by a safety margin. Not only is that disk not fully used right away, but, in reality, might not be used for several years. Nevertheless, that disk still must be paid for.

Disk striping is effective when you need to improve the performance of the disk I/O. However, it is also difficult to estimate the number of disks that will satisfy the requirements when testing the striping, and an up-front investment in hardware is required when performing the testing.

Explanation of the Cloud Solution/Pattern

The AWS Cloud lets you use virtual disks. A virtual disk provides you with whatever capacity you need, whenever you need it.

When you use a virtual disk you will no longer need to make elaborate estimates in advance. Once the system is running, you can secure a disk with the required capacity on-demand while monitoring capacity usage.

Because there is a tendency for the cost of storage to fall year-by-year, securing capacity only as it becomes necessary is extremely beneficial in terms of cost. In the past, a situation where there is a temporary need for increased capacity would require you to buy a high-capacity disk, but with a virtual disk, you can add the disk can when necessary and then eliminate it after use.

Implementation

The virtual disk Elastic Block Store (EBS) in AWS allows you to secure the required capacity at any time, with whatever timing you need.

(Procedure)

- When using EBS, first secure a volume of the minimum required size.
- If you need more disk capacity, take a snapshot of the EBS and create a new EBS based on that snapshot.
- When creating the new EBS, specify a volume of a larger size than the original volume.

- Attach the new EBS to an EC2 instance.
- After attaching, use the Resize command in the file system that is being used (for example, resize2fs) to expand the region to the new capacity.
- When striping, attach multiple EBS volumes and use mdadm or operating system functions to create a software RAID configuration.

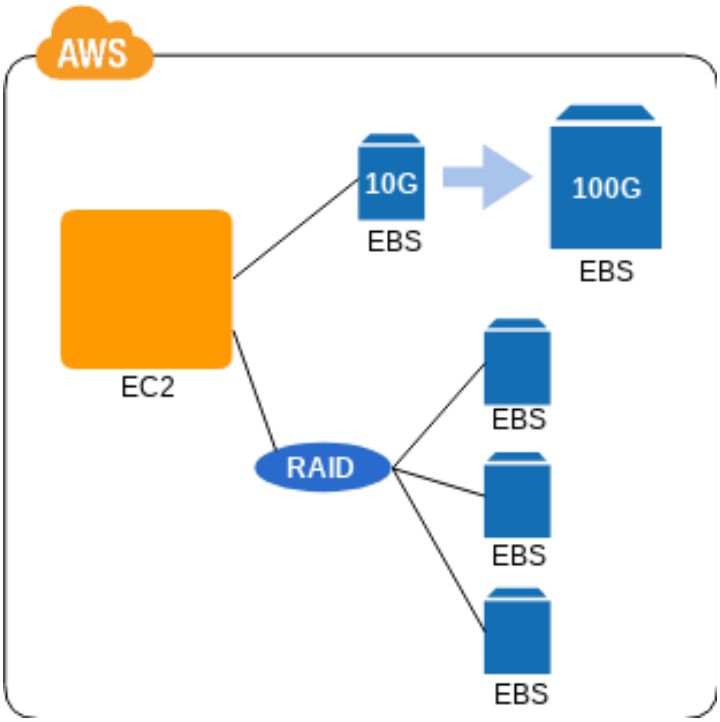
Implementation

The virtual disk Elastic Block Store (EBS) in AWS makes it possible to secure the required capacity at any time, with just the right timing.

(Procedure)

- When using EBS, first secure a volume of the minimum required size.
- If more disk capacity is required, take a snapshot of the EBS and create a new EBS based on that snapshot.
- When creating the new EBS, specify a volume of a larger size than the original volume.
- Attach the new EBS to an EC2 instance.
- After attaching, use the Resize command in the file system that is being used (for example, resize2fs) to expand the region to the new capacity.
- When striping, attach multiple EBS volumes and use mdadm or operating system functions to create a software RAID configuration.

Configuration



Benefits

- You can change the volume size later, freeing you from the work involved in making estimations.
- Being able to secure disk capacity when required has benefits in terms of cost.
- Striping can improve disk I/O performance.

Cautions

- Unlike S3, in EBS there is a charge for the disk capacity secured. For example, if a 100 GB region is secured, you will be charged for 100 GB, even if only 5 GB is used.
- The upper limit for disk capacity that the set in a single EBS (as of April 2012) is 1 TB. When capacity in excess of 1 TB is needed in a single partition, attach multiple EBSs and use software such as mdadm to combine them together into a single volume for use.

Other

An EBS is a disk volume that can be used over a network, so using an EC2 instance with a larger size increases the I/O performance. When striping, in particular, you should be aware of the EC2 instance size.

Multi-Server Pattern

Problem to Be Solved

You can improve availability in a variety of ways and in various different layers. On the disk layer, for example, you may use a RAID configuration, and on the network layer you may provide spare lines.

The same is true for servers as well, where you can use multiple physical servers to provide redundancy. Understand, however, that simply increasing the number of servers does not necessarily create a redundant structure. Because the materials required for providing redundancy, such as server machines and load balancers, involve considerable expense, sometimes when you take cost effectiveness into consideration, the best action is no action.

Explanation of the Cloud Solution/Pattern

Provide multiple virtual servers in parallel, using the Load Balancer provided by the AWS Cloud service to distribute the load appropriately. This is known as a "Multi-Server." While you can achieve this in an on-premises system, you can structure this environment substantially more easily in the AWS Cloud than ion-premises.

While n the past the load balancer itself has been an expensive appliance, with the AWS Cloud you do not need to go through the expense of purchasing or maintaining the load balancer. With the AWS Cloud, the Load Balancers are also on a pay-as-you-go basis. In consideration of redundancy, you can also switch the configuration from a single high-performance server to a Multi-Server arrangement, with each server having somewhat lower processing capacity.

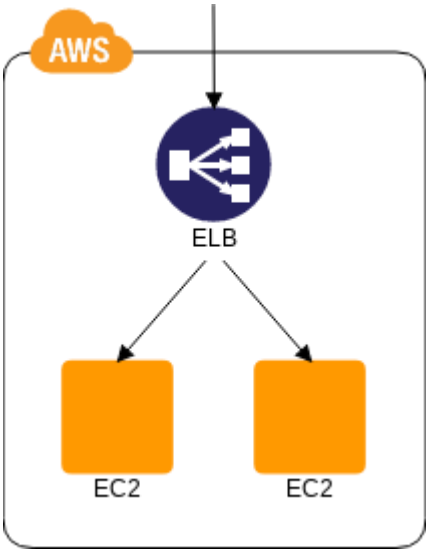
Implementation

Use the AWS Load Balancer service, "Elastic Load Balancing" (ELB). Once the ELB has received a processing request for an EC2 instance, the ELB distributes processes appropriately to the instances bound thereto. You can use the ELB health check function (a function for confirming whether or not an EC2 instance is operating properly), so as to not distribute a process to an EC2 instance that is not operating properly. As a result, you can make sure that the processing can continue using the remaining EC2 instances even if there is a failure in one EC2 instance.

(Procedure)

- Launch an EC2 instance and set up the operating system, and the like.
- Apply the Stamp Pattern, to launch multiple EC2 instances.
- Launch ELB and bind the multiple EC2 instances.
- Set up the options so as to use the health check function to check the bound EC2 instances.

Configuration



Benefits

- Even if there is a failure in one EC2 instance, the system as a whole can continue to operate.
- The combination of ELB and Auto Scaling makes it possible to run a constant number of virtual servers (EC2 instances), even when a failure occurs. For example, if the setup is so as to have a minimum of three virtual servers operating, then if one virtual server were to fail, another virtual server would be added automatically to always ensure three virtual servers. In this case, you must specify the Amazon Machine Image (AMI) in advance, and specify that the EC2 instance is to be launched from that AMI. <ref group="Related Blog"> Template:Related Blog </ref>

Cautions

- Because multiple EC2 instances are used in ELB, the cost is more than for a single-EC2 configuration.

- Use caution when there is data that must be shared on the middleware level or the application level. For example, because you have to share session information between multiple EC2 instances, you have to share sessions using a session database, or to use StickeySession. See the StateSharing Pattern.
- Always use an N+1 configuration (where one spare virtual server is always available). For example, if you need three virtual servers for the required processing capacity, add one spare server to use a four-server configuration to be able to handle one of the virtual servers going down.
- Note that when you provide redundancy in databases, you may have issues with data synchronization. See the DB Replication Pattern.

The Multi-Server pattern increases availability through providing a structure that always has redundancy, but there is also a technique where you can use the Floating IP pattern rapid recovery when there is a failure.

Multi-Datacenter Pattern

Problem to Be Solved

While you can apply the Multi-Server Pattern to improve availability when thinking about a server failure, the Multi-Server Pattern cannot handle a case where one envisions a data center-level failure (such as when there is a power outage, an earthquake, a network failure, or the like).

When you consider failures on the data center level, you will need to use multiple data centers. However, securing multiple data centers that are adequately far apart from each other, and purchasing physical hardware to provide redundancy in their systems, is extremely costly. On top of this, procurement and set up is also time-consuming, and from a cost effectiveness perspective, this is often difficult to achieve.

Another factor making this difficult to achieve is that improving availability involves more than merely setting up multiple data centers – you also need high-speed dedicated communication lines in consideration of data synchronization and communication between the data centers.

Explanation of the Cloud Solution/Pattern

Multiple data centers provide the Amazon Cloud services, and usually dedicated lines are connected between the individual data centers. You can specify the data center to be used, to structure systems (using virtual servers) in each of the data centers depending on your load. This allows you to structure systems that are distributed across multiple data centers more easily and less expensively than in the past. This makes it possible are you to create a system that can withstand even catastrophic events and data center-level failures.

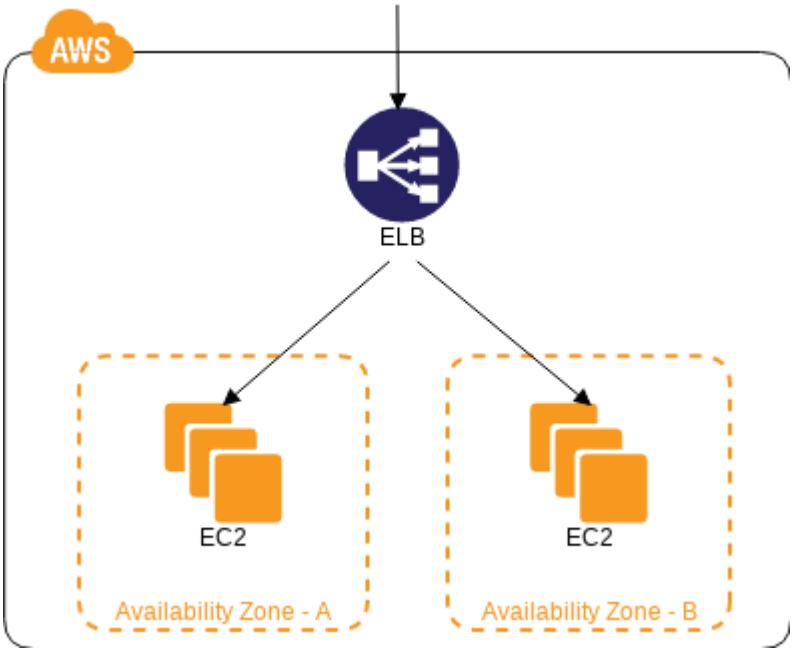
Implementation

Amazon Web Services (AWS) has multiple data centers, known as Availability Zones (AZ), in regions such as Tokyo and Singapore. Select the AZ to use and specify the AZ for locating each EC2 instance. The AZs are connected to each other through high-speed dedicated cables, making it possible to structure systems that span multiple AZs.

The implementation is essentially identical to that of the Multi-Server pattern. The difference is that when locating the EC2 instances, you can select different AZs when launching the instances. Because the Elastic Load Balancing (ELB) is structured automatically spanning multiple AZs, you do not need to worry about the load balancing.

Moreover, this exists not just on the web layer, but on the database layer as well, in a structure that spans multiple AZs in all layers.

Configuration



Benefits

- You can structure a system that will continue to provide service even when there is a major failure on the data center level.
- This makes it possible for you to configure, quickly and inexpensively, the type of Disaster Recovery (DR) structure that has been the goal since the Great Eastern Japan Earthquake.
- Because in AWS there are neither initial setup fees nor monthly use fees for each individual AZ, there is no difference in your cost regardless of whether you use a single AZ or multiple AZs.

Cautions

- Note that when databases are used in a master-slave configuration, the master will be located in one of the AZs. The AZs are connected to each other through high-speed dedicated cables, but the speed will be less than that of communication within and AZ. Use caution to prevent the speed of communication between AZs from becoming a bottleneck.

- If you are worried about the speed of communication between AZs, you can use an application or middleware such as HAProxy to communicate with an EC2 instance within the same AZ, and then, in the event of a failure of that EC2 instance, change the setup so as to communicate with an EC2 instance in another AZ. <ref group="Related Blog"> Template:Related Blog </ref>
- To achieve high fault tolerance, you need to locate the required numbers of servers redundantly in each individual AZ. For example, if you need three servers, then it will not be possible to handle the load when there is a failure in one of the AZs unless you locate three servers in each of the AZs.
- As of April 2012, Elastic Load Balancing (ELB) does not support redundant structures that span regions.

Other

- When you use multiple AZs you will have to pay for the communication between the AZs, but the cost is low (at US$0.01 per gigabyte as of April 2012), so fundamentally we recommend that you use the Multi-Datacenter Pattern when setting up a redundant structure that uses multiple servers.

Floating IP Pattern

Problem to Be Solved

You need to stop the server when applying a patch to a server, or when upgrading the server (to increase the processing capabilities). Because stopping a server stops the services it provides, you need to to minimize the downtime.

For web servers, you can use the Domain Name System (DNS) to swap the server. However, in this case as well, typically the swapping time cannot be shortened to less than the Time to Live (TTL) value, so this is not suited to instant swapping.

Explanation of the Cloud Solution/Pattern

With the physical servers of the past, you would have to provide a spare server to prepare for taking down a server. After you had taken down the actual server, you would have to start up the spare server to take over processing. You would have to set up the IP address (to that of the actual server).

You can achieve this easily and extremely quickly through the AWS Cloud. You can prepare a machine image that will enable you to start up the required virtual server whenever you need it. Moreover, because an API for specifying the IP address is also available, you can write a script to automate all the processes from starting up the server through setting the IP address.

Implementation

A static IP address used in Amazon Web Services (AWS) is known as an "Elastic IP Address (EIP)." You can use this to reassign IP addresses. You can detach an EIP from an existing EC2 instance and attach it to another ECP instance, to swap the virtual server that provides the services.

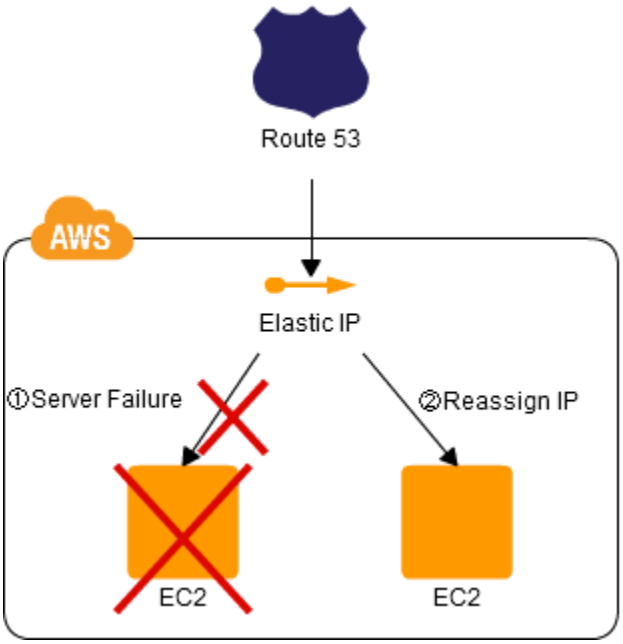
(Procedure)

- Assign an EIP to an EC2 instance.
- When there is a failure or when you are going to perform an upgrade, launch a new EC2 instance. You can use the Stamp Pattern when launching the new EC2 instance. You can perform the swap more rapidly by launching the other EC2 instance in advance.

- After launching the instance, detach the EIP from the current EC2 instance and attach it to the new EC2 instance.

One of the implementation example is http://stevemorad.s3.amazonaws.com/reInvent/demos/vip_template.htm. It was demoed during the AWS re:Invent CPN207 - Virtual Networking in the Cloud session will enable one EC2 instance to monitor another EC2 instance and take over a private "virtual" IP address on instance failure.

Configuration



Benefits

- You can swap servers by merely reassigning the EIP, unaffected by the TTL of the DNS.
- When performing an upgrade, you can achieve instantaneous fall-back through assigning the EIP back to the original server in the event of an error in the server to which the EIP had been swapped temporarily.
- Because you can apply EIPs across different Availability Zones (AZs), even if there were a failure on an AZ level, you can reassign the EIP to a server in a different AZ.

Cautions

- Switching an EIP normally takes several seconds.

- In a Virtual Private Cloud (VPC), you can create additional virtual network cards called an Elastic Network Interfaces, or ENIs. These will have a fixed IP addresses (note: the address is “fixed” (unchanging) from the infrastructure perspective, however from the operating system perspective it is a dynamic address). You can assign and then later move an ENI and its accompanying private address(es) and (if applicable) EIP(s) to another EC2 instance, enabling you to apply this pattern even within a private network. Note, however, that since their primary IP addresses are fixed, ENIs are by definition limited a single subnet inside of VPC.
- When you make a Secure Shell (SSH) connection to a new instance behind an EIP, a potential security issue warning will be issued, and login may become impossible.

Other

- You may use monitoring software such as HeartBeat, Nagios, or Zabbix to detect a failure. Because you can use a program to perform the EIP reassignment, you can automate this process through a combination with the monitoring software.
- You can use this in parallel with Server Swapping Pattern to not only reassign the EIP, but to swap the EBS as well, enabling data inheritance.
- There is also a technique for switching to a spare server using Elastic Load Balancing (ELB) without using the EIP, by adding or deleting a health check file.

Deep Health Check Pattern

Problem to Be Solved

You can use the health check function in the Load Balancer to evaluate the status of the servers bound to the Load Balancer when distributing processes.

In a configuration with a web server, a proxy server, an AP server, and a DB server, let's think about a case with a Load Balancer prior to the web server. The Load Balancer is able to evaluate the status of the web server, and cut off the web server if it is malfunctioning. However, the Load Balancer is unable to discern the status of the back-end servers, such as the proxy server, the AP server, and the DB server.

Explanation of the Cloud Solution/Pattern

You can use the health check function in the Cloud Load Balancer to set up a dynamic page in PHP, JavaServlet, or the like (that is, a program) to perform checks. The program is able to check the operations of the proxy servers, AP servers, and DB servers, and the like, to return the results to the Load Balancer. This makes it possible to check the health of the system as a whole.

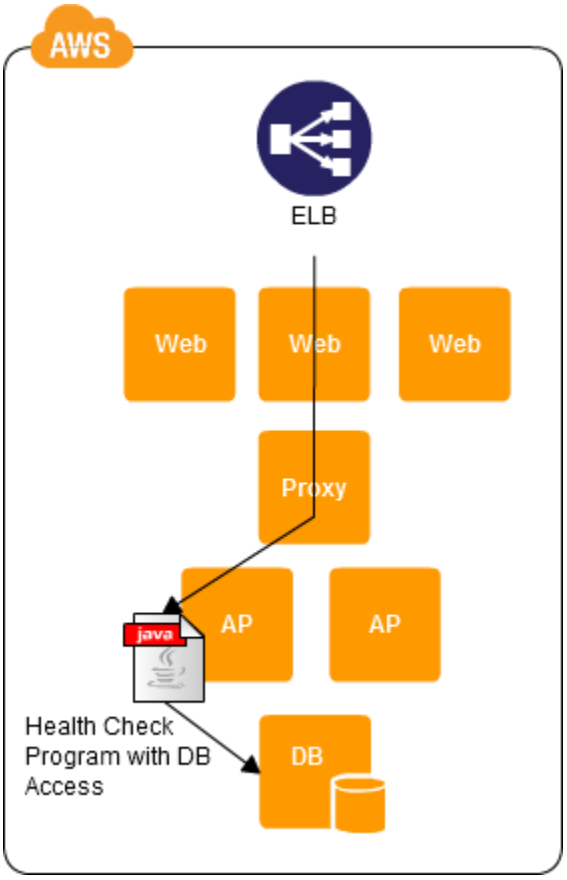
Implementation

The health check function of the ELB Load Balancer service in AWS performs a status check in terms of whether or not HTTP(S) access to a specified URL is possible. Using this, set the destination for the health check to a dynamic page. We will give an example of implementation for a system structured from a web server, a proxy server, an AP server, and a DB server.

(Procedure)

- Start the ELB and enable the health check function.
- Create the program to run on the AP server. Have that program involve accessing the database.
- Set the URL for the health check in the ELB to the program, and have the request to that URL activate the program.
- Execute the health check from the ELB.

Configuration



Benefits

- This makes it possible for you to check all of the servers required for system operation.
- Depending on how the program that responds to the health check is made, it may perform a Close Process (that is, not accept the request), or return customized error information, depending on the detail of the failure.

Cautions

- If there is a large number of servers, then the health checks themselves will contribute to the traffic, so you must carefully consider the timing for the health checks.
- If the DB server has become a single point of failure (SPOF) and has gone down, there may be an overreaction that can take all of the servers down, depending on how the back-end server check program is written.
- The DB Replication Pattern should be used in parallel so that the DB server part does not become a SPOF.

Clone Server Pattern

Problem to Be Solved

A scale-out structure is a common technique, but in systems that start small often the structure is not one where it is possible to provide Multi-Server services using multiple servers at all. In such a case, establishing measures to handle increased load may be time-consuming.

Explanation of the Cloud Solution/Pattern

This pattern allows you to take a system that has not taken distribution of load into account, and easily make it into a system where load distribution is possible. A server that already exists is used as a master, and a machine image is prepared for a server to be added. In the machine image, content synchronization and coordination of database connections has been performed in advance. This allows you to achieve load distribution through scale-out by merely starting up the machine image.

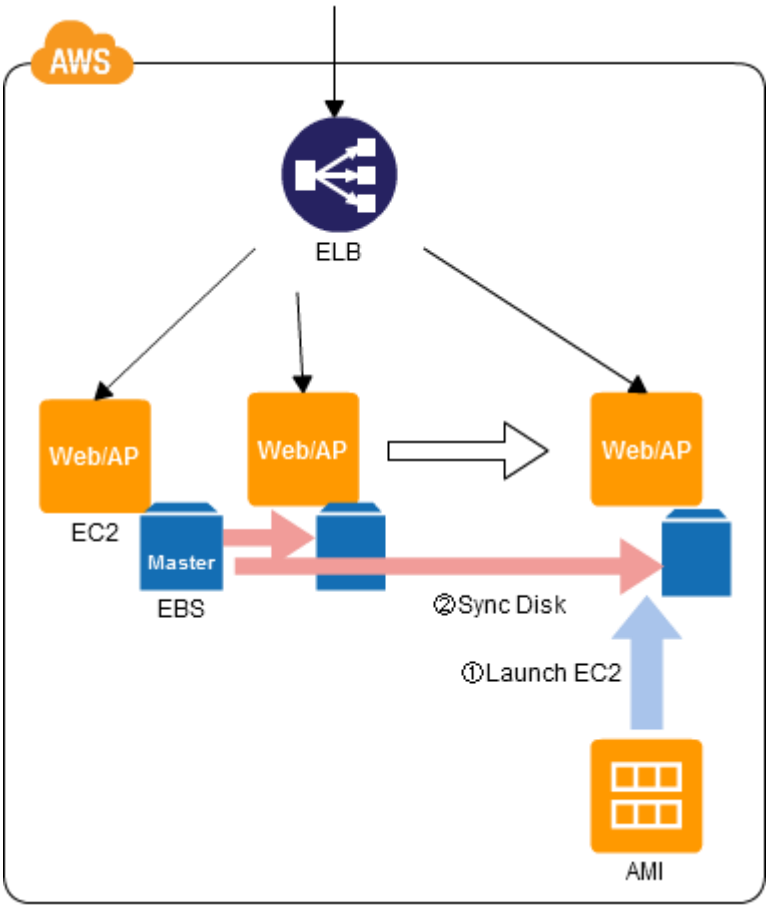
Implementation

Use the Load Balancer service (Elastic Load Balancing (ELB)) and an Amazon Machine Image (AMI). Produce an AMI for cloning where content synchronization, and the like, have been adjusted to enable load distribution. If the load becomes too great, start up an EC2 instance from this AMI for cloning. Having the EC2 instance subject to load distribution by ELB makes lets you perform scale-out with essentially no changes to the existing system.

(Procedure)

- Startup the ELB (for a structure with only one EC2 instance), and place the EC2 instance under the control thereof.
- Create, from the EC2 instance that is currently running, an EC2 instance for cloning.
- Periodically use rsync, or the like, as necessary, to perform file synchronization with the master EC2 for the EC2 for cloning.
- Depending on the load (or when a high load is anticipated), start up the required numbers of EC2 instances for cloning that are required, and add to the ELB.

Configuration



Benefits

- This lets you perform load distribution through Scale Out easily, without modifying the existing system.

Cautions

- The master EC2 instance becomes a single point of failure (SPOF).
- If a database is running in the master EC2 instance, do not run the database in the clone EC2 instance, but rather have the master EC2 instance as the destination for the database connection.
- If there will be file uploads or writing, do so using the master EC2 instance (such as through using mod_proxy of Apache to proxy from a clone virtual server of only the applicable URL to the master virtual server, or the like).

NFS Sharing Pattern

Problem to Be Solved

Content synchronization is necessary when distributing loads over multiple servers. While periodic one-way synchronization from master servers to slave servers is easy, there is a problem of a time delay when using periodic synchronization. Moreover, if a record or file is written in the slave server, there is still the problem of how to reflect the changes to the master server and the other servers.

Explanation of the Cloud Solution/Pattern

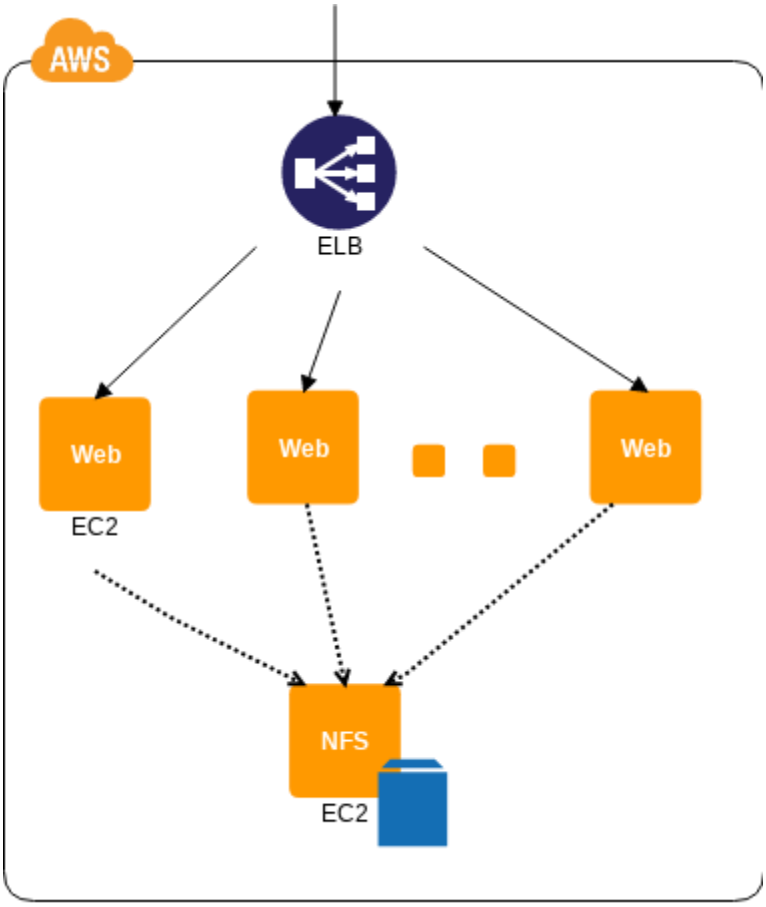
This pattern lets you write identical content between multiple servers in real-time. The master virtual server that stores the shared content is defined as the Network File System (NFS) server, and the slave server is defined as an NFS client. This lets you update the content from any server, enabling sharing in real-time.

Implementation

(Procedure)

- Build an NFS server on an EC2 instance.
- Place the content to be shared on the NFS server.
- Have the scale-out servers reference the content on the NFS server.

Configuration



Benefits

- Placing the shared content on the NFS server enables real-time synchronization.
- Content can be shared by merely mounting the NFS server, simplifying setup.

Cautions

- The synchronization of disks as explained in the Clone Server Pattern can coexist with NFS. Content that is updated frequently should be shared through NFS.
- The NFS server requires administration.
- You have to take into consideration the NFS access performance when the number of EC2 instances is large.
- To prevent the NFS server from becoming a SPOF, think about solutions such as GlusterFS.

NFS Replica Pattern

Problem to Be Solved

When files are shared by multiple servers using NFS, the loss in performance due to the NFS part will become significant when the number of servers sharing the files increases and the access frequency gets high.

Explanation of the Cloud Solution/Pattern

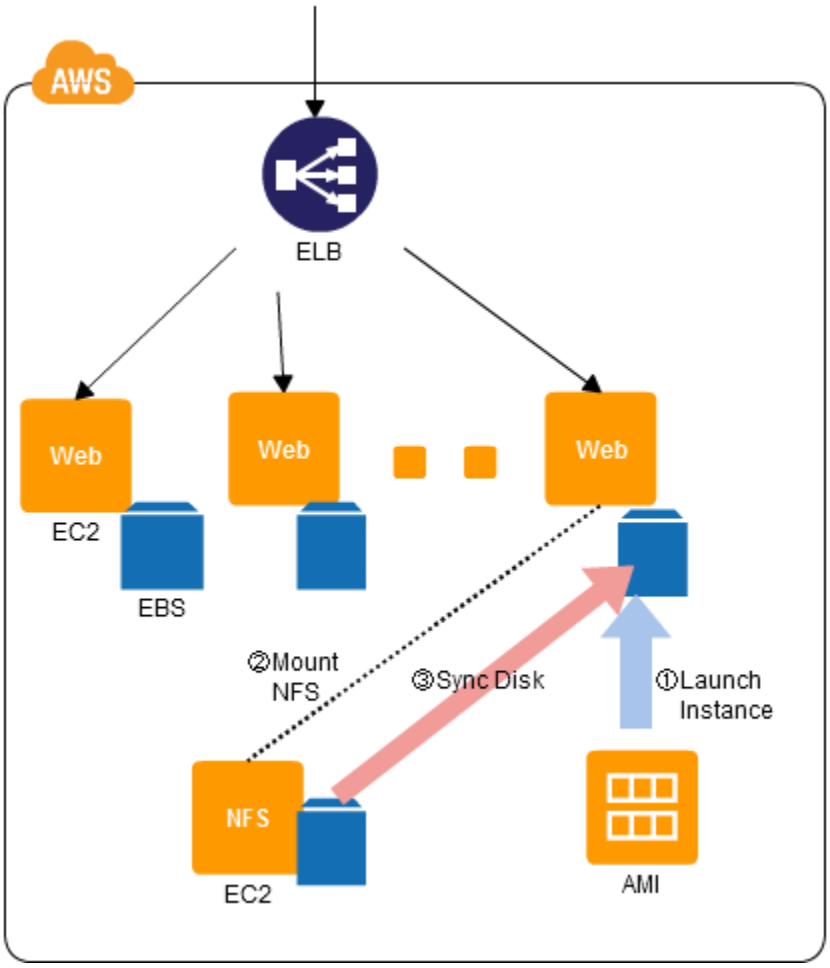
This pattern allows you to improve the referencing performance, in particular, in response to a drop in performance of a network file system (NFS) that stores shared files. Prepare virtual disks individually on each server and copy the shared files of the NFS server.This lets you use the virtual disk as a reference-only replica of the NFS for each server.

Implementation

Copy the files of the NFS server to the Elastic Block Store (EBS) that is the virtual disk for each individual EC2 instance.You can read in the EBS files for each individual EC2 instance, enabling higher performance than when accessing the NFS server.

- Structure the NFS server on an EC2 instance, and place the shared files there.
- Start up an EC2 instance (a web server) in Auto Scaling first. When it has been started up, mount the NFS server, and then copy the content of the NFS server to the EBS.
- Set up the applications on each of the EC2 instances to have the EBS as the destination for referencing.

Configuration



Benefits

- When a shared file on the NFS server is updated, that file is used by the EC2 instances started up thereafter.
- There is no need to access the NFS server, so performance is unlikely to become an issue, because the shared files exist on the EBS for the individual EC2 instances.
- Even if, for example, the NFS server goes down, the content is stored on each individual EBS, so the NFS server does not become an SPOF.

Cautions

- When updating shared files, updating the files on the NFS server alone does not reflect them to the individual EC2 instances. You must perform synchronization using rsync, or the like.

Other

- You can improve performance and reduce costs through using a local disk of an EC2 instance, known as Instance Storage (an ephemeral disk), instead of using EBS as a local disk.

State Sharing Pattern

Problem to Be Solved

Often state information that contains information that is unique to the user (HTTP session information) is used when generating dynamic content. However, when multiple web/AP servers are running under the control of the Load Balancer, if you have each individual web/AP server possess state information, this could cause the state information to be lost if there were a server failure or if you intentionally reduce the number of servers.

Explanation of the Cloud Solution/Pattern

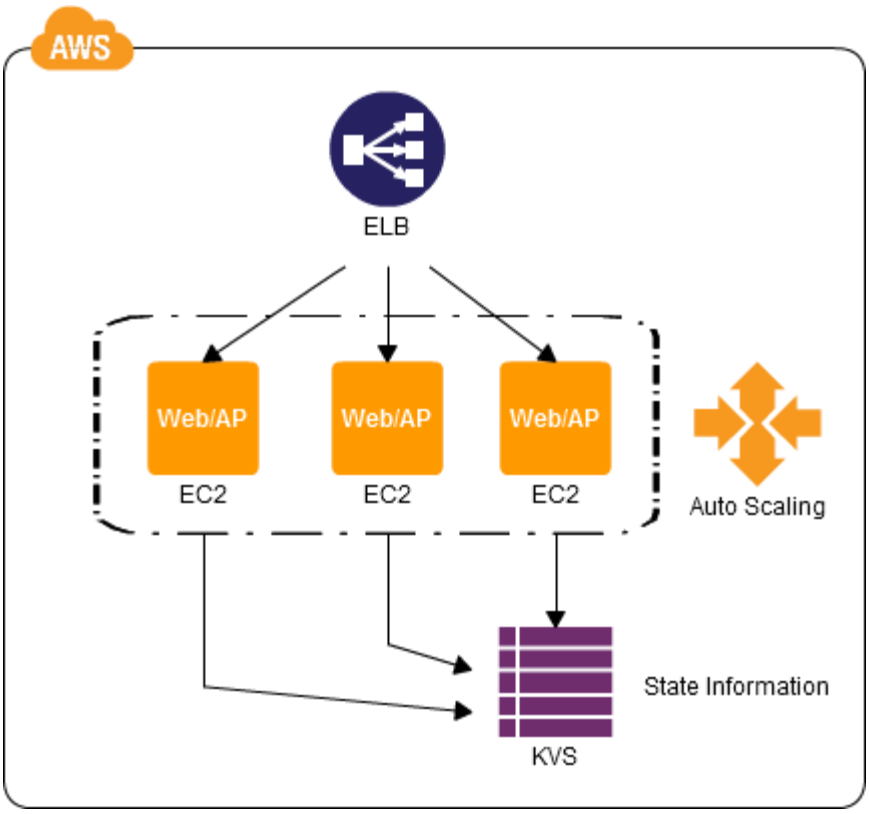
This pattern lets you maintain state information in a scale-out structure. This ensures inheritance of state information when the number of servers is increased and prevents loss of state information when the number of servers is decreased (including when there is a failure). The state information is placed in a high-durability shared data store (memory/disk), to be information that is referenced by multiple servers. This makes the server "stateless," having no state information. Even if a new server is added, it is the shared data store that is referenced to cause the state information to be inherited.

Implementation

The AWS data stores include "ElastiCache," "SimpleDB (KVS)," and "DynamoDB (KVS)." All of these let you store state information. Select One of these, depending on the requirements.

- Prepare a data store for storing the state information.
- Use, as a key in the data store, an ID that identifies the user (a session ID or user ID), and store the user information as a value.
- Store, reference, and update the state information in the data store, instead of storing it in the web/AP server.

Configuration



Benefits

- This lets you use the scale-out Pattern without having to worry about inheritance or loss of state information.

Cautions

- Because access to state information from multiple web/AP servers is concentrated on a single location, you must use caution to prevent the performance of the data store from becoming a bottleneck. If the performance requirements are challenging, consider selecting DynamoDB, which rarely becomes a bottleneck.
- Depending on the requirements, the Amazon Relational Database Service (RDS) (a relational database management system) or the Amazon Simple Storage Service (S3) (Internet storage) may be used for the data store.

URL Rewriting Pattern

Problem to Be Solved

When a web service is provided by a virtual server, if the number of accesses becomes too large, the load is handled by increasing the number of virtual servers or upgrading the specifications of the virtual servers. However, often the majority of the accesses are requests for static content, so how to distribute access to static content has become an important issue.

Explanation of the Cloud Solution/Pattern

You can use Internet storage to distribute access of static content. This makes it possible to handle the load without increasing the number of virtual servers.

Using this method requires that the URL for the static content be changed to a URL for the Internet storage. But this does not imply that you always have to adjust the static content directly; you can use the the Filter function of the web server to change the URL at the time of delivery. Instead of delivering from Internet storage, you can also deliver the content from a content delivery server.

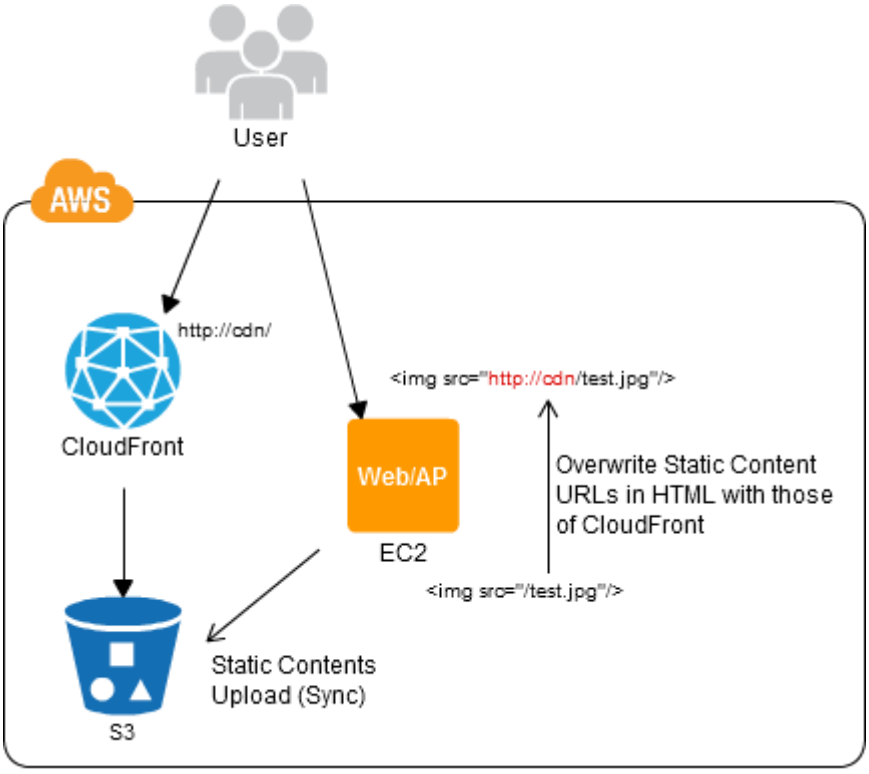
Implementation

In AWS, you can use the Amazon Simple Storage Service (S3) to deliver static content. You can deliver content globally without delay by using content located on Amazon Simple Storage Service (S3) as the original, and using the CloudFront content delivery service.

(Procedure)

- Upload (synchronize) a portion of the static content (JavaScript/CSS/images, etc.) of the EC2 to S3.
- If necessary, generate a CloudFront using, as the original, the S3 where the static content has been uploaded (synchronized). (When CloudFront is used, it is also possible to use an EC2 instance directly as the original server, rather than S3.)
- Overwrite the static content URLs in the HTML tags with those of the S3 or of CloudFront.
- You can use the Apache filter module or Nginx, prepared as a proxy, to perform the overwriting dynamically.

Configuration



Benefits

- You can provide greater robustness to load and can also reduce EC2 costs through distributing the access of static content to S3/CloudFront.
- You can use CloudFront as a way to handle latency caused by distance in worldwide delivery.
- When mod_ext_filter/mod_sed, Nginx, or the like is used, you can use this pattern by inserting a filter, without modifying the original HTML file. You can easily fall back to the state where CloudFront is not used, by simply turning the filter off.

Cautions

- When CloudFront is used, there may be some cases where deletion or updating may be delayed due to content caching.

Rewrite Proxy Pattern

Problem to Be Solved

There is a technique for handling load where static content is located on Internet storage or a content delivery service. However, this technique requires you to change the access destination for the static content to the Internet storage, and requires modifications to the existing system such as overwriting the URLs within the content or setting up filters to a web server.

Explanation of the Cloud Solution/Pattern

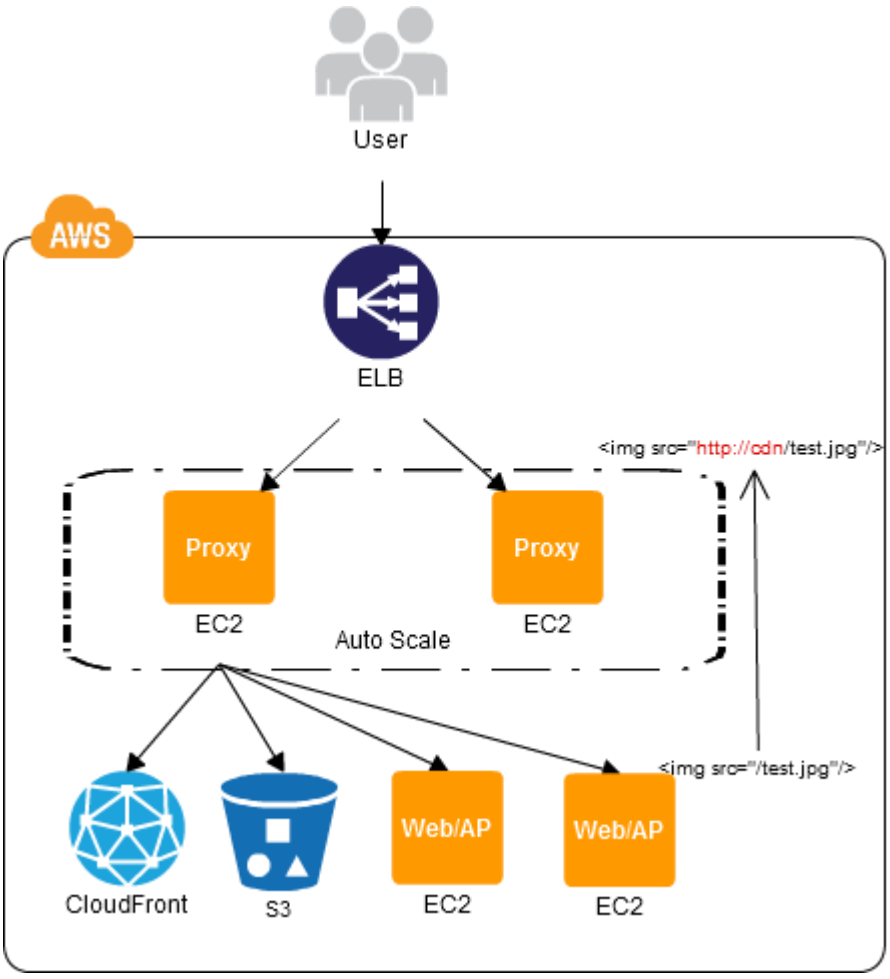
You can change the access destinations without modifying the existing system through providing proxy server. Providing a proxy server in front of the server for storing the content changes the access destinations of the static content to the Internet storage or content delivery service.

Implementation

Construct a proxy server using popular software such as Apache or Nginx, and locate the proxy server in front of the existing system.

- Locate a proxy server able to overwrite the content, such as Nginx, between the Elastic Load Balancing (ELB) and the Amazon Simple Storage Service (S3) (which stores the static content), running the proxy server on an EC2 instance.
- Add to the proxy server rules for overwriting URLs within the content.
- As necessary, apply Auto Scaling to the proxy servers.

Configuration



Benefits

- Overwriting the access destinations using a proxy server makes it possible to distribute the load of static content without modifying the existing system.

Cautions

- You must endure redundancy of the proxy server so as to not create an SPOF (single point of failure).
- The web/AP server is located indirectly in ELB, and thus cannot be attached directly to ELB even if there is an increase or decrease in the number of web/AP servers (EC2 instances) through auto scaling.

Cache Proxy Pattern

Problem to Be Solved

When you use multiple web/AP servers as a way to handle high loads, it multiplies the expense. If your budget is tight, you need to think about techniques that will not increase the numbers of web/AP servers.

Explanation of the Cloud Solution/Pattern

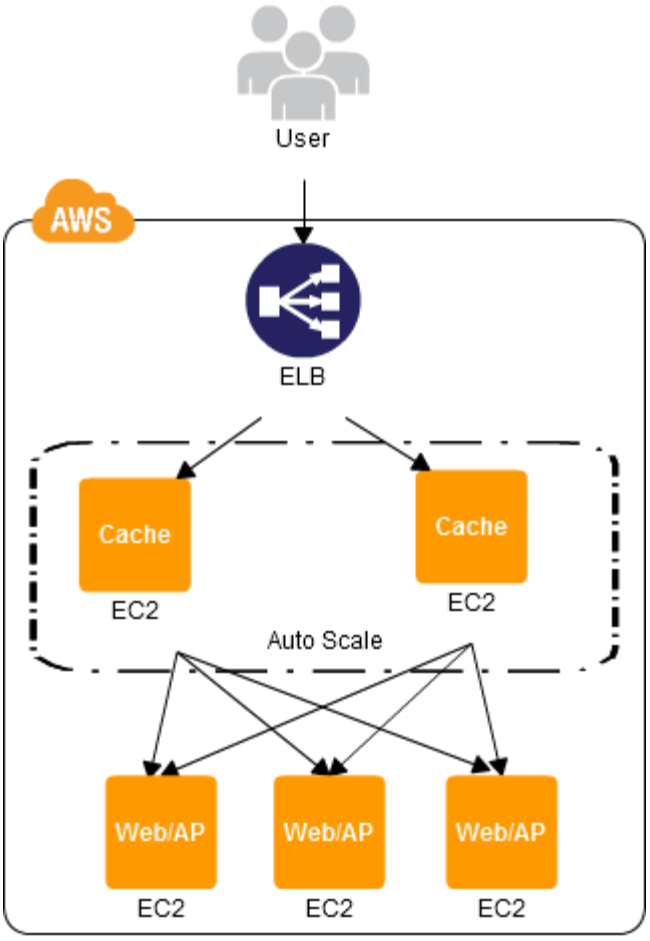
One way to increase the performance of a web system is to cache your content. This is a technique where static content or dynamic content that does not change (much) is cached upstream of the web/AP server and that content is delivered by the upstream cache server, which has high delivery performance, until the cache expires. You can construct virtual servers easily in the AWS Cloud, enabling you to implement cache servers easily into your system.

Implementation

On an EC2 instance, install a common cache server software, such as Varnish, and use it as a cache server.

- Place the cache server software, such as Varnish, in front of the web/AP server.
- Place the cache server in front of the web/AP server.
- In the cache server, set up the server to host the original data, the cache expiration, and the like.

Configuration



Benefits

- This lets you deliver content using the cache, without modifying the web/AP servers.
- For dynamic content in particular, this greatly reduces the load of content generation.
- You can set up a flexible cache by either caching or not caching HTTP headers, URLs, cookies, and the like.

Cautions

- You need to ensure redundancy in the cache server as well so as to not create a single point of failure (SPOF).
- Because the EC2 instance that is used as the web/AP server is located indirectly in Elastic Load Balancing (ELB), you need to be clever in in how you attach to the cache server when the number of web/AP servers has been increased or decreased through Auto Scaling.

Scheduled Scale Out Pattern

Problems to Be Solved

The Scale-Out Pattern is effective when handling high levels of traffic in a web service that is structured in a Cloud environment. However, when monitoring the load status to manually add virtual servers, or when automatically adding instances depending on the load statuses of the virtual servers, the creation of instances may not be able to keep up with the demand when there is a sudden increase in accesses (such as a case where traffic doubles in less than five minutes).

Explanation of the Cloud Solution/Pattern

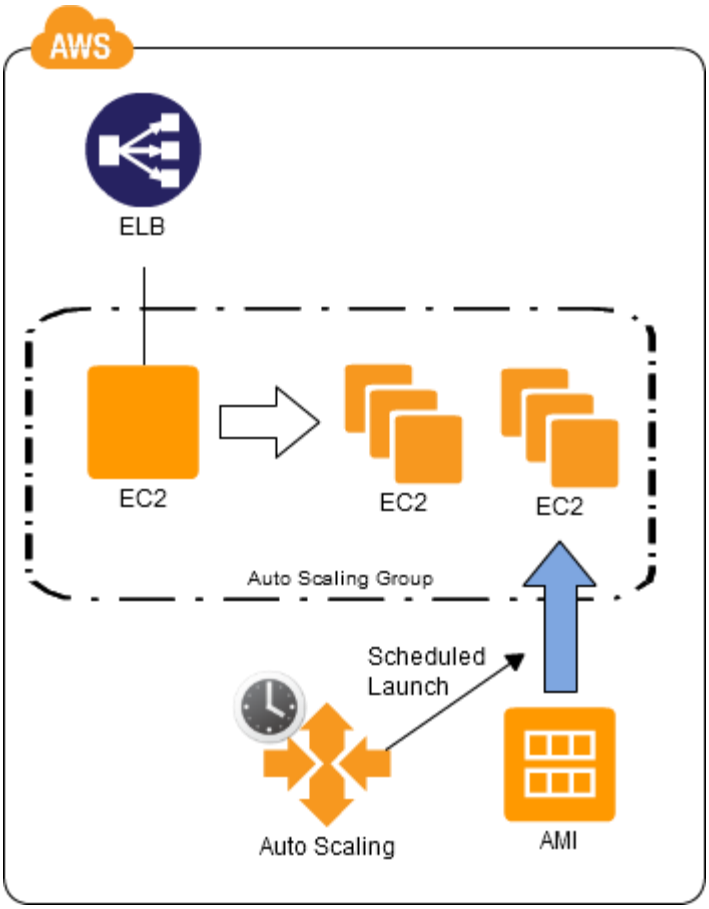
Use scale-out through scheduling when the timing when there will be an sudden increase in traffic is understoodi n advance. While the fundamental structure is similar to that of the Scale-Out Pattern, the key difference is that of performing the scale-out through specifying the timing for doing so. You can handle a rapid increase in traffic robustly by completing the scale-out in advance, and you can minimize wasted costs by performing the scaling immediately prior to the rapid increase in traffic.

Implementation

Auto Scaling in AWS has a function for specifying a time for changing the settings. You can use this function to configure scheduled scale-out. You can also perform Scale-in through specifying a time band when the traffic is anticipated to settle down.

- Reference the Scale-Out Pattern to set up Auto Scaling (including scale-out triggers and scale-in triggers).
- Specify the timing with for increasing the number of EC2 instances and change the setting for "minimum number of instances (– – – min-size)" to the number of instances to be provided. * At the specified time, new EC2 instances will be launched, up to the minimum number of instances specified.
- If the minimum number of instances again is reduced with the timing with which the load is to settle down, then scale-in will follow the trigger that has been set.

Configuration



Benefits

- This allows you to increase the number of EC2 instances following the schedule with which the traffic volume is anticipated to increase.
- This reduces your cost because the number of EC2 instances is reduced when there is little traffic.
- When compared to scale-up, the limit on processing capability is extremely high because the required number of EC2 instances can be provided in parallel, under the control of the ELB.

Cautions

- Be aware that the specified time is in terms of UTC.
- When there is to be a sudden increase in traffic, you must scale-out the ELB in addition to scaling-out the EC2 instances. In this case, you should perform pre-warming.

Web Storage Pattern

Problem to Be Solved

Delivery of large files, such as video files, high-definition image files, and zip files, from a single web server can become a problem in terms of network load. In such a case, you can distribute the load through multiple web servers to reduce the load on the network– but this involves locating the large files on multiple servers, which is a problem in terms of cost.

Explanation of the Cloud Solution/Pattern

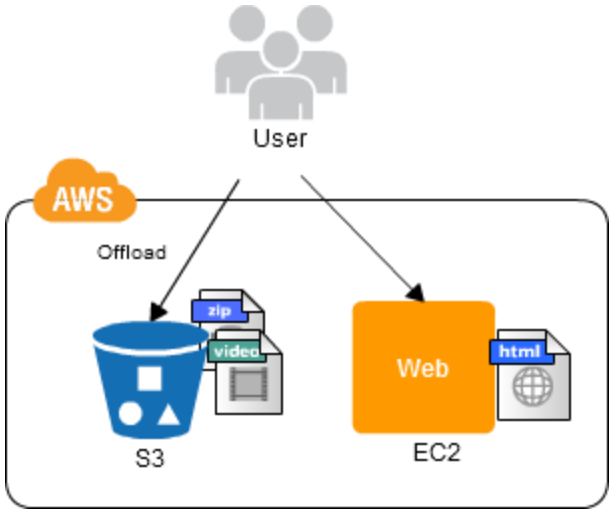
You can solve the problems with the network loads and disk capacities of the web servers by locating the large files in Internet storage and delivering the files directly from there. Objects that are stored in Internet storage can be accessed directly by users if set to being public. This enables delivery from Internet storage, thereby reducing the network load of the web server, and also eliminates the need to copy data between virtual servers in synchronizing the delivery files.

Implementation

Place the content to be delivered onto the Amazon Simple Storage Service (S3), and enable user downloading directly from the S3.

- Create a "bucket" on the S3 Internet storage, and upload the static content (image/video/compressed files, or the like) to be published.
- Set this content to be public, enabling user access. When set to be public, a URL will be issued for each individual content object. Provide, to the user, a URL that has been issued, such as http://(bucketname).s3.amazonaws.com/(filename)*, or create a link on a webpage.

Configuration



Benefits

- The use of S3 eliminates the need to worry about network loads and data capacity.
- S3 performs backups in at least three different datacenters, and thus has extremely high durability.
- Because a URL is issued for each content object, the files can be used for a broad range of purposes, such as file sharing, merely through placement on S3.

  Cautions

  Because the content that is delivered requires independent Domain Name System (DNS) naming in S3, the DNS name of the main site cannot be used as-is. For example, if the main site is "www.my-site.org," then the content on S3 requires a different DNS name, such as "data.my-site.org," or the like. Because of this, you may need to change link destinations in the HTML files that have already been constructed. However, in this case you may be able to use the URL_Rewriting Pattern to handle this through performing batch overwriting of the modules of the web server.

Direct Hosting Pattern

Problem to Be Solved

You are not able to keep up by increasing the number of machines when there is a sudden increase in the number of accesses in a short period of time. While you might consider handling this by increasing the number of servers by forecasting the increase in accesses, increasing the number of servers wastefully is a problem in terms of cost.

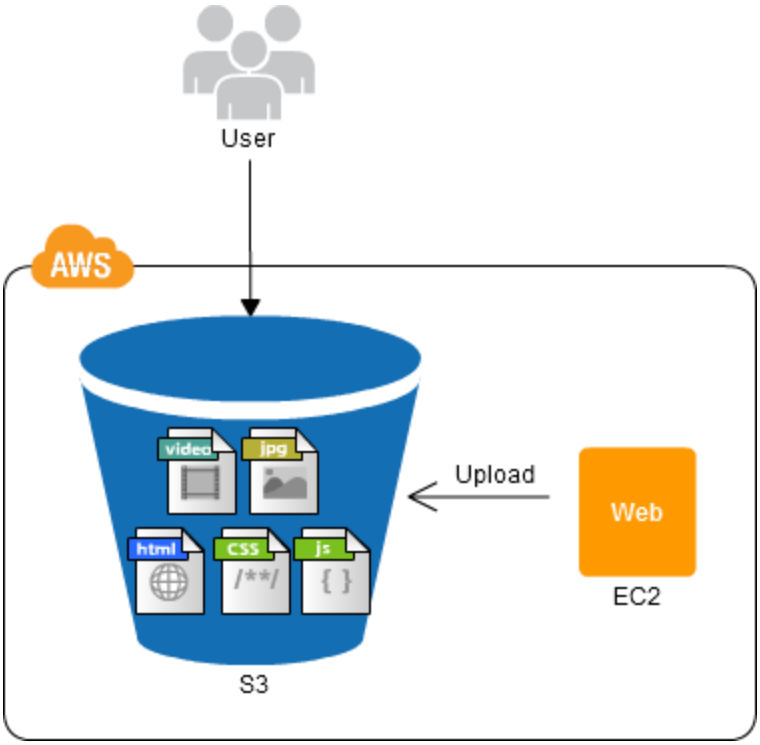
Explanation of the Cloud Solution/Pattern

In this pattern, Internet storage provided by the AWS Cloud is used as a web server, to not only host large static files such as images and video, but to host HTML as well. Because Internet storage was originally designed envisioning use as shared storage, it has no problems in terms of capacity. Even if the number of accesses for a particular service were to increase suddenly, Internet storage as a whole would be able to perform the processing without problems, enabling use as a web server without you having to take any special measures to handle the load.

Implementation

- Upload, onto Amazon Simple Storage Service (S3) Internet storage, the static content to be published (HTML, CSS, JavaScript, images, video, or the like).
- Set up the S3 bucket to publish the content. Set up the bucket policies to permit public access of the content of the bucket.
- Turn on the website hosting function of S3 and set up an index page and an error page, to host the website on S3 alone.

Configuration



Benefits

- Access to the static content is left to S3, enabling you to increase the the availability and durability of the web system with ease.

Cautions

- It is not possible to run a server-side program on S3, so it is not possible to output different pages for, for example, each user that has logged in.
- When JavaScript is embedded in the content that is delivered from S3 and data is to be obtained from another server through asynchronous communication, the server and DNS name for the location from which to acquire the data will be different, and thus there will be the need to communicate through JSONP, due to the constraints on closed main communication in JavaScript.

Other

- This pattern can still be used even in a dynamic site, such as the Amazon Content Management System (CMS). For example, if you use MovableType, when you post a blog, the blog engine writes out a static HTML file, which you can host in S3.
- S3 also has a function for issuing signed URLs. You can use this to issue a signed URL for allowing access to limited users only. You can also set an expiration for the signed URL. See the Private Distribution Pattern.

- A system for performing access validation, known as a "bucket policy," is also provided in S3. You can use this to enable access to specific users only, or to limit access to HTTPS only.
- As of March 2012, over 900 billion objects (files) are saved in S3, and, at peak times, over 700,000 requests per second have been handled.

Private Distribution Pattern

Problem to Be Solved

Internet storage has both high availability and high durability, and targets delivery of large content files and frequently accessed content files. However, when delivering content to only specified users, cooperation with the validation systems of the applications that created the content is indispensable. This makes it difficult to achieve access control with Internet storage alone.

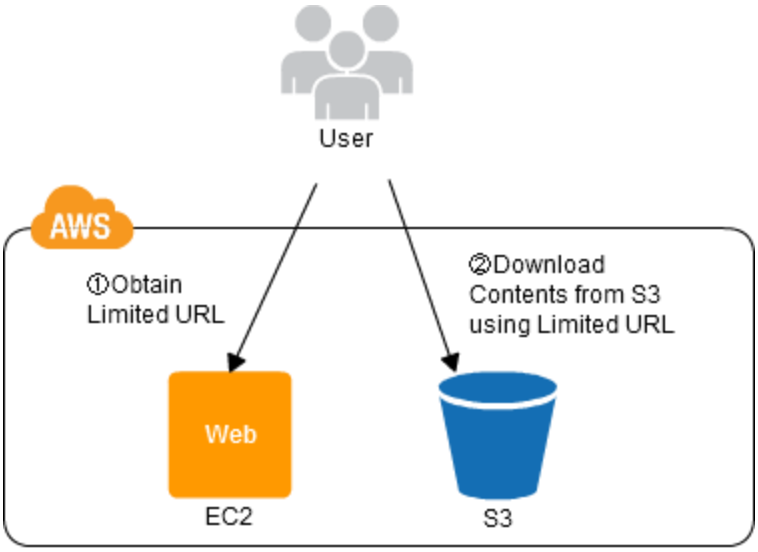
Explanation of the Cloud Solution/Pattern

You can use a function for publishing limited URLs, provided by Internet storage, to set up access-origin IP addresses and access-enabled time intervals. You can issue URLs for individual users, and have content that is downloadable by only those limited URLs, to prevent downloading if a link has expired or if an individual with a different IP address attempts access. Essentially, this lets you provide content to specified users only.

Implementation

- Prepare the S3 apitool or the AWS SDK.
- After execution of user validation on the local system, use an API to generate a limited URL for each content object published to that user.
- Use the URL table that is generated to generate webpages dynamically.
- Use the limited URLs as the link (download) destinations in the content, such as the HTML, that is generated.

Configuration



Benefits

- This enables delivery of private content through time-limited use by specified users only.
- Because the actual content download is performed directly from S3 (rather than passing through an EC2 instance), the properties of S3 of being robust to load and to failures are directly applicable.

Cautions

- You must provide a validation system and a server for issuing time-limited URLs.
- Even if the user validation has not expired, the term of effectiveness of the URL will expire, preventing downloading.

Other

Generally you would use this pattern in combination with an application validation system. In the case of content that can be accessed by all logged-in users, you can create webpages using URLs generated manually, using, for example, a third-party tool (such as CloudBerryExplorer), if not generating the limited URLs automatically.

Cache Distribution Pattern

Problem to Be Solved

Given the rising popularity of computers and mobile devices, ever more users are accessing content on the Internet from an increasingly broad range of locations. On top of this, image and video data are now of higher definition, so the data volume has also become extremely large.

From the perspective of the user experience, you must provide the data to the user more quickly and with greater stability, but, given the technology of today, there is, for example, a communication delay of at least 200 ms when accessing a server on the East Coast of the United States from Japan. For this reason, having only a single transmission origin for the content will negatively affect the user experience.

Explanation of the Cloud Solution/Pattern

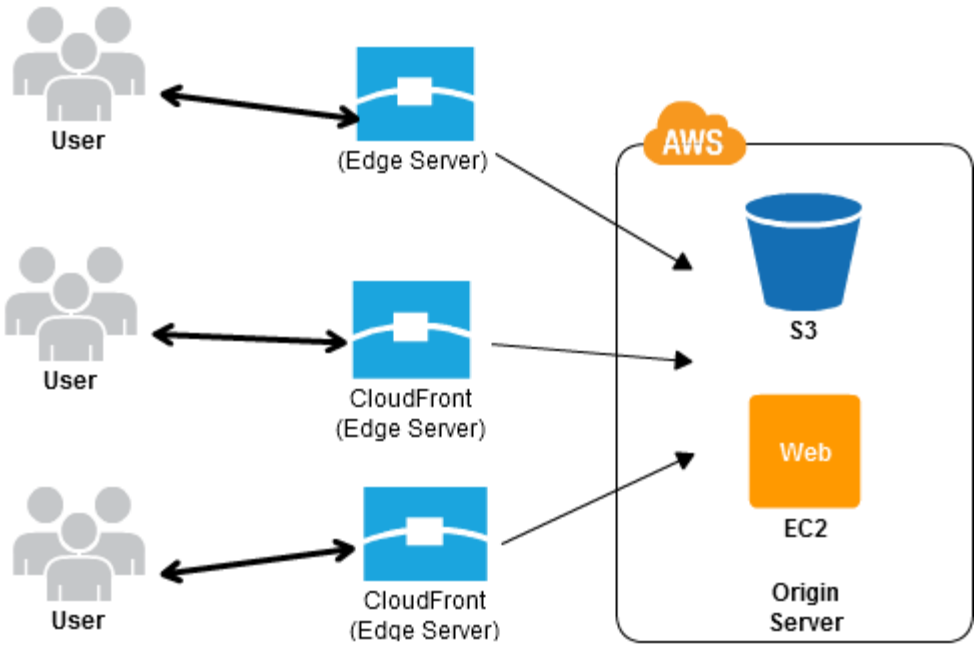
Cache data for content that is delivered from a content delivery origin is located at locations that are provided in various regions throughout the world. This enables you to deliver the content from a location that is physically near to the user, enabling you to resolve the geographical/physical constraints. You can use this pattern to reduce the distance between the user and the content, thus improving the responsiveness to the user.

Implementation

You can use CloudFront in AWS to use cache servers ("edge servers") throughout the world.

- Determine the origin server that is to be the delivery origin for the content, and place the content there.
- Set up CloudFront to use a master server. When performing the setup, a Domain Name System (DNS) name of "xxxx.cloudfront.net" (where the xxxx portion is generated at random) will be issued.
- You may use the DNS name that is issued automatically, or you may use an independent domain name. At this time, set the CloudFront DNS name that has been issued in the CNAME record for the DNS name of the origin server.

Configuration



Benefits

- This makes it possible for you to provide a better user experience to users in geographically distant places.
- This lets you distribute the file download processes, which is useful in load distribution as well.
- You can use an existing server (a server other than an EC2 instance, such as one for an on-premises system or for hosting) as the origin server, enabling you to use this pattern while taking advantage of existing servers.
- You can also use S3 directly as the origin, using it as an origin server.

Cautions

- Fundamentally, the cache server performs caching based on the cache timeout setting of the master server that is the transmission origin. Because of this, there may be cases where the cache server will not be updated even if the file on the master server is updated prior to the cache timeout. You must consider the nature of the content when setting the cache timeout. You may also consider using the Rename Distribution Pattern.

Rename Distribution Pattern

Problem to Be Solved

If you are delivering content using the Cache Distribution Pattern, when a file on the master server is updated, the data on the edge server (the cache server) is not updated until it times out. This cannot handle a case when you want to update a file with a given timing.

Explanation of the Cloud Solution/Pattern

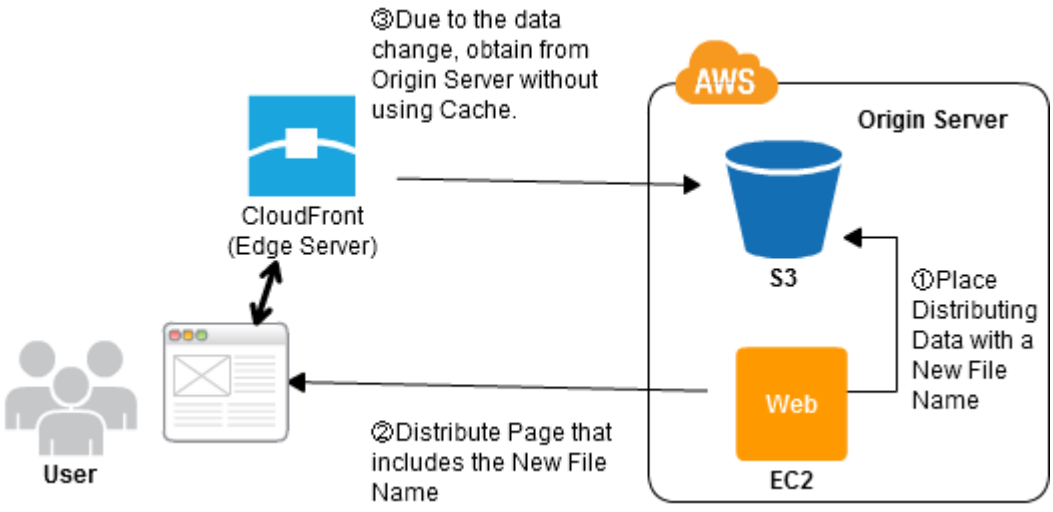
For cache data on an edge server, the URL that accesses that data is used as a key. If there is a file you want to update, you can locate it under a different filename and then change the access URL itself, enabling you to deliver the new content regardless of the cache timeout on the edge server.

Implementation

(Procedure)

- Create base content (such as an HTML file that includes the access URL) separate from the content.
- For the base content, either shorten the cache timeout or always distribute from the master server.
- When updating content distributed by CloudFront, provide that content on the master server, under different names.
- Update the URLs in the base content to the URLs of the new content.

Configuration



Benefits

- This lets you deliver new content without waiting for the cache timeout, when content has been updated on the master server.

Cautions

- Because this would be ineffective if the cache timeout for the base content itself were too long, you should set the cache timeout for the base content to a short value. This reduces the caching effect for the base content, however.
- Because the old file will remain on the edge server until the cache timeout, specifying the URL directly may cause the old file to be downloaded. If necessary, you may delete the old file or use an invalidation function.

Patterns for Uploading Data

Write Proxy Pattern

Problem to Be Solved

Typically Internet storage has extremely high capacity in regards to reading, along with extremely high data durability. However, to maintain redundancy, not only is data written to multiple locations, but also communication with the client is through the HTTP protocol; as a result, the writing speed is relatively slow. Because of this, there is a performance problem when writing large amounts of data to Internet Storage.

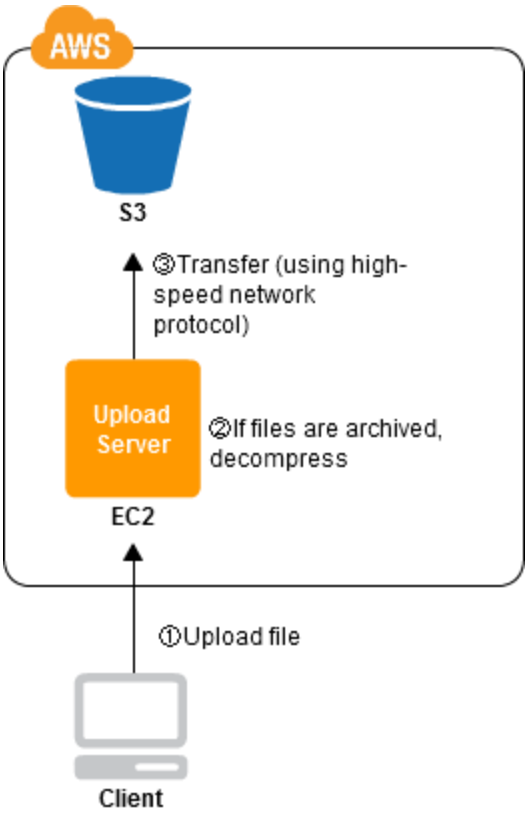
Explanation of the Cloud Solution/Pattern

Rather than passing data directly to Internet storage from the client, configure so that the data is received by a virtual server, and then is forwarded to Internet storage from the virtual server. The transfer from the client to the virtual server can use a protocol that is faster than HTTP (for example, a protocol based on UDP). when there is a large number of small files, they may first be archived on the client side and then decompressed after transfer to the virtual server, followed by forwarding to Internet storage. If the virtual server and the Internet storage are in the same region, then the connection will be through a dedicated line, which can reduce the total transfer time through the virtual server substantially when compared to that of transferring to Internet storage directly.

Implementation

- Start up an EC2 instance for receiving the data. Start up the EC2 instance in the same region as the Amazon Simple Storage Service (S3) that is the ultimate data storage destination.
- Install, on the EC2 instance, an FTP server or web server, UDP transfer software such as Aspera or TsunamiUDP, or software able to accelerate the transfer speed. (This server is known as the "upload server.")
- Transfer the data from the client to the upload server. If there is a large number of small files, first combine them into a single file on the client.
- After completion of the transfer to the upload server (or as a sequential operation), transfer from the upload server to S3. If archived on the client, transfer to S3 after decompression on the upload server.

Configuration



Benefits

- This can increase the speed of transfer to S3.
- In particular, you can expect a dramatic increase in the transfer speed when uploading to a S3 in a region in a foreign country.

Cautions

- Because in some cases the speed of writing to the EC2 instance that is the upload server (which is typically the speed of writing to the EBS) may become the bottleneck, you may need to perform disk striping (see Ondemand Disk Pattern), to increase the writing performance.
- Because the pipeline for a small EC2 instance is relatively narrow, use a large instance if high performance is required.

Other

- Solutions for increasing the speed of data transfer using UDP include TsunamiUDP, Aspera, SkeedSilverBullet, and so forth.

- You can use a technique where the file is divided into parts and written in parallel (known as the "multi-part approach") as a way to increase the performance of writing to S3.
- This enables you to not just increase speed, but increase convenience for the user as well through, for example, uploading to an EC2 instance through FTP and then automatically synchronizing to S3 as-is.

Storage Index Pattern

Problem to Be Solved

Due to the distribution of locations of data, Internet storage is excellent in terms of both durability and availability. However, because access is via the Internet, typically responsiveness is poor when compared to on-premises access. In some cases, high-speed search functions are not provided, requiring you to take some action on the application side when retrieving a data table for a specific user or when retrieving data for a specific range of dates.

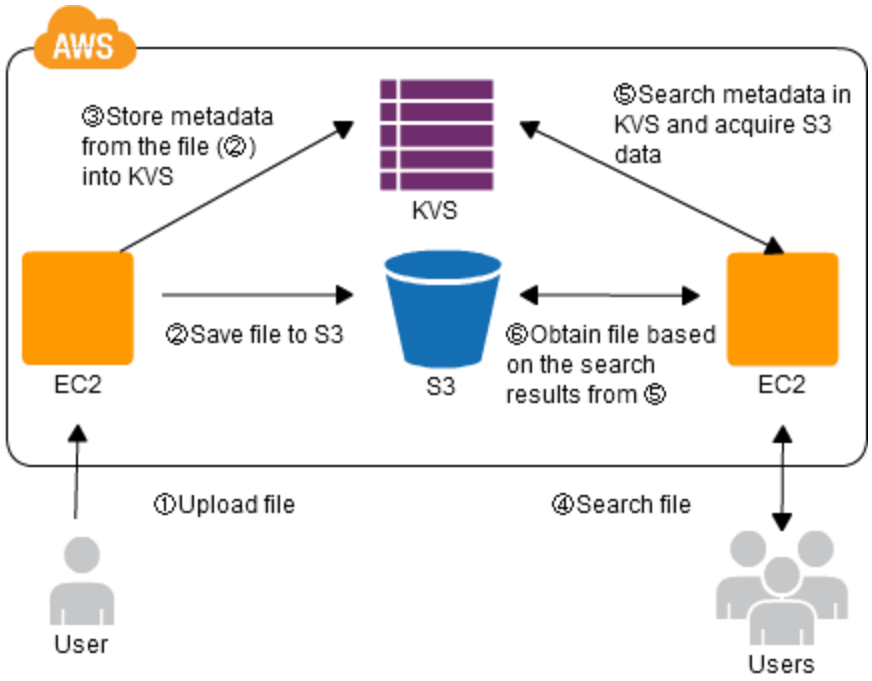
Explanation of the Cloud Solution/Pattern

When storing data into Internet storage, metadata is stored at the same time into a KVS that has excellent searching performance, and then that data is used as an index. When searching, the KVS is used and access the Internet storage based on the result returned.

Implementation

- After storing data to Amazon Simple Storage Service (S3), store S3 metadata (keys, paths, data sizes, storage times, and the like) in SimpleDB or DynamoDB.
- When performing searches or tabulation, perform the processing using SimpleDB or DynamoDB.
- Acquire the S3 data based on the result of processing in SimpleDB or DynamoDB.

Configuration



Benefits

- This enables you to use robust high-capacity storage functions along with high-performance searching.

Cautions

- Correct search results will not be produced if there is a mismatch between the data in S3 and the KVS metadata. It is imperative that the data recording and the metadata recording be performed simultaneously.

Direct Object Upload Pattern

Problem to Be Solved

Large data files from a large number of users are uploaded to photograph and video sharing sites. In some cases the upload process involves a high server-side load (particularly in terms of the network load), requiring a virtual server that is dedicated to uploads, even in sites that only are of a moderate scope.

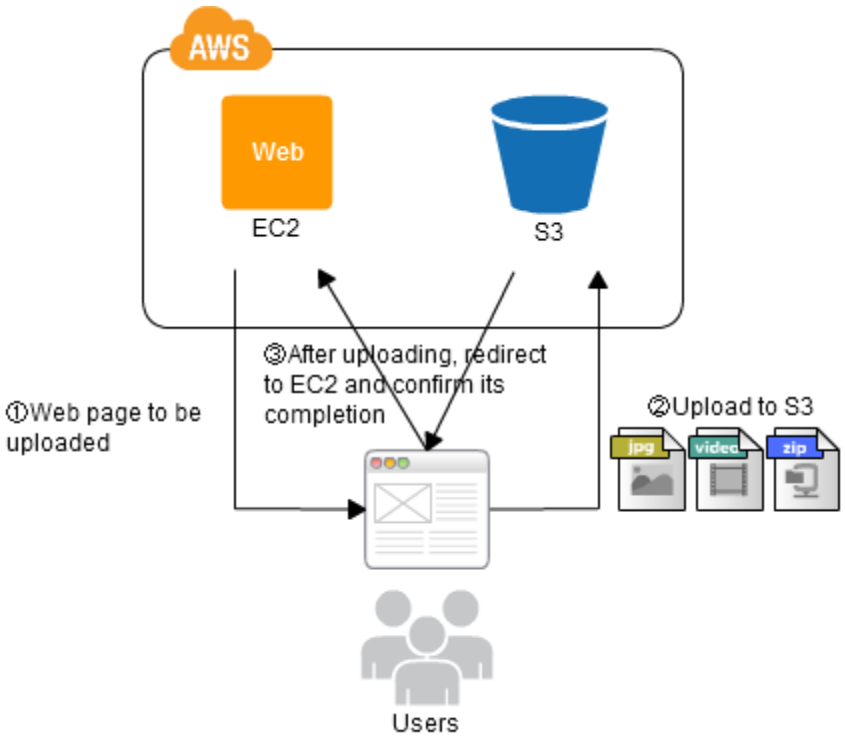
Explanation of the Cloud Solution/Pattern

Leave the upload process to the Internet storage. That is, rather than having the upload from the client go through a virtual server, upload directly to the Internet storage. This lets you ignore the web server load caused by the upload process.

Implementation

- Generate an HTML form, on the web server (the EC2 instance) for performing uploading to the Amazon Simple Storage Service (S3).
- Use the upload form to upload the file directly from the user side to S3. Because there will be a redirect to the URL specified in the form after completion of transference of the file to S3, perform a process to confirm the completion of uploading on the server that is the destination of the redirect.

Configuration



Benefits

- There eliminates the labor or cost of preparing an EC2 instance for uploading.
- This enables distribution of the load of the upload processes by taking advantage of the scalability of S3. The data is uploaded to S3, making it easy to share between EC2 instances.

Cautions

- Because implementation must be performed in coordination with S3, this approach is complex when compared to uploading using an EC2 instance alone.

Other

- See the following regarding a function for uploading to S3 using an HTML form: http://doc.s3.amazonaws.com/proposals/post.html
- Because the HTML form can be created manually, both the distribution of the HTML form and the uploading can be performed using S3 alone.

DB Replication Pattern

Problem to Be Solved

The basic technique for saving data that is important to the system is that of storing it in a database. Recently the use of database replication functions has been becoming increasingly widespread. While replication is performed relatively commonly, in the past this has usually been limited to being within a given datacenter, due to the costs involved. However, it is important to think about cases where the entire datacenter may be damaged. (Major disasters, such as the Great Eastern Japan Earthquake, have actually caused such cases.)

Explanation of the Cloud Solution/Pattern

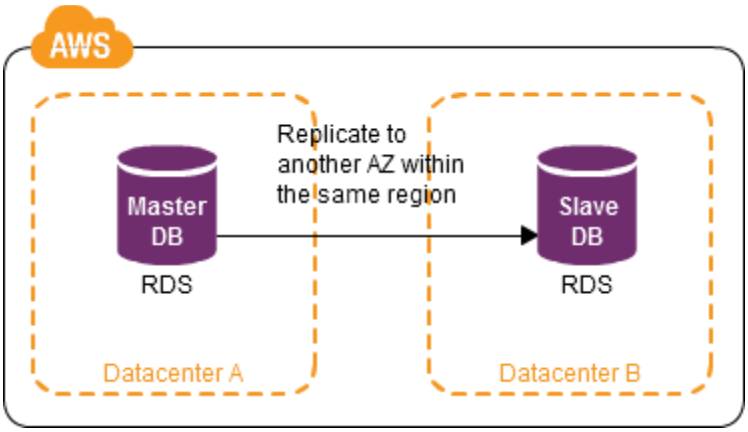
This is a pattern that lets you perform replication crossing geographic locations. This pattern lets you prevent data loss and maintain data access availability. While this is a pattern that was used even before Cloud technology, the use of cloud technology enables inexpensive use of multiple geographic locations, making such distribution a practical option.

Implementation

In AWS, there are the concepts of "regions" and "Availability Zones" (AZs). A "region" is the broader concept, where there are multiple AZs within the Tokyo Region, for example. In consideration of this point, you can locate EC2 instances transparently in different datacenters, enabling database replication to those different datacenters. You can use a Multi-AZ Amazon Relational Database Service (RDS) to achieve replication across AZs with ease. Of course, databases may be installed in EC2 instances to achieve this instead.

- Locate two EC2 instances in AZs of different geographic locations.
- Install Relational Database Management Systems (RDBMS) in each of the EC2 instances and set up the replication.

Configuration



Benefits

- This makes it possible to continue the operations without loss of data, even in the case of a disaster or failure.
- Switching the access destination to the replicated database lets you apply a patch to a database without shutting down the system.

Cautions

- While this makes fail-over to the slave possible when a failure has occurred in the master database, be aware that the fail-over will require some downtime.

Other

- For the purposes of disaster recovery, set up replication for databases to locations that are far apart geographically (in different regions).
- When replicating to a database in another region, there may be some loss in performance when synchronous replication is used. In this case, consider asynchronous replication or periodic replication.

Read Replica Pattern

Problem to Be Solved

Often server specifications are upgraded (that is, the server is scaled-up) when the resources of a database server are overwhelmed when the frequency of access to the database is high. When scaling-up becomes difficult, you can perform scaling-out ito horizontally distribute the database servers; however, typically this is difficult as well. Normally the proportion of reading from a database is high when compared to writing to the database, so you should distribute the reading processes to the performance of the system as a whole.

Explanation of the Cloud Solution/Pattern

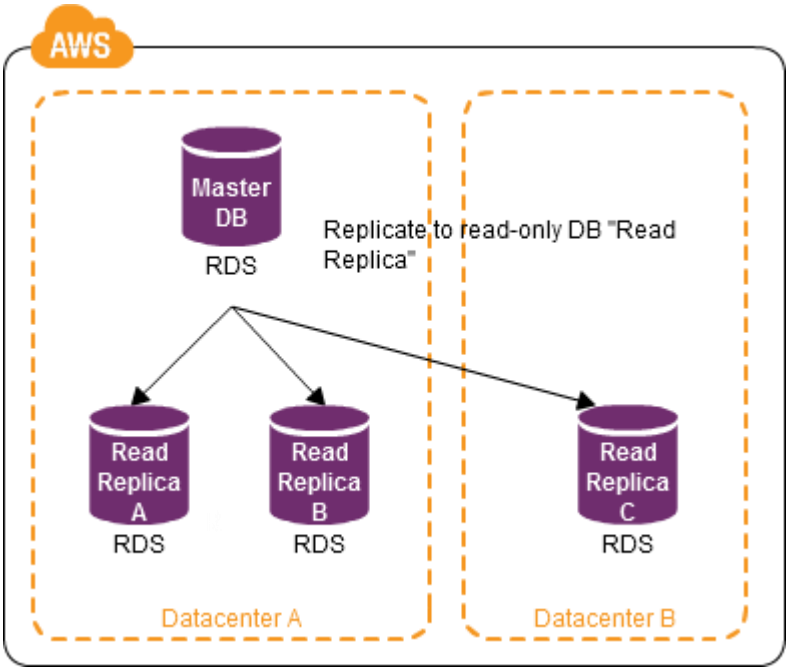
There are several ways to improve the reading performance. You can use this pattern to improve the overall performance through distributing the reading into multiple "Read Replicas" (that is, replicas for reading). A Read Replica follows the writing to a master, reflecting the data to itself. You can reduce the load on the master through using primarily Read Replicas for reading.

Implementation

The AWS Relational Database Management Systems (RDBMS) service, the Amazon Relational Database Service (RDS), has a function enabling the easy creation of a read-only database known as a "Read Replica." An EC2 instance can also be used instead to create a read-only database.

- Create a read-only replica of the master database. For a database handled by RDS, create the read-only replica using the Read Replica function.
- When data is read by an application, set the access destination to the Read Replica.
- While multiple Read Replicas can be used, they must be assigned on the application side. In doing so, middleware such as HAProxy or MySQL Proxy may be used as well.

Configuration



Benefits

- If the load in reading from a database is high, you can use this to distribute that load.
- This is also effective if you want to perform a process without placing a load on the master database, in, for example, a data analysis application.
- Because the purpose of the Read Replica is not that of a redundant configuration, if the point is to increase database durability, consider database replication itself, rather than a Read Replica. Of course, you can use a Read Replica and database replication in parallel.
- Typically, a Read Replica is an asynchronous replication, so be aware that there will be a slight lag between the master and the Read Replica.
- Disabling the automatic backup n an RDS (that is, setting the "Backup Retention Period" to 0), will prevent the Read Replica from that RDS.

Inmemory DB Cache Pattern

Problem to Be Solved

The majority of the database load usually is involved with reading. Because of this, you can improve the performance of the system as a whole by improving the performance of reading from the databases.

Explanation of the Cloud Solution/Pattern

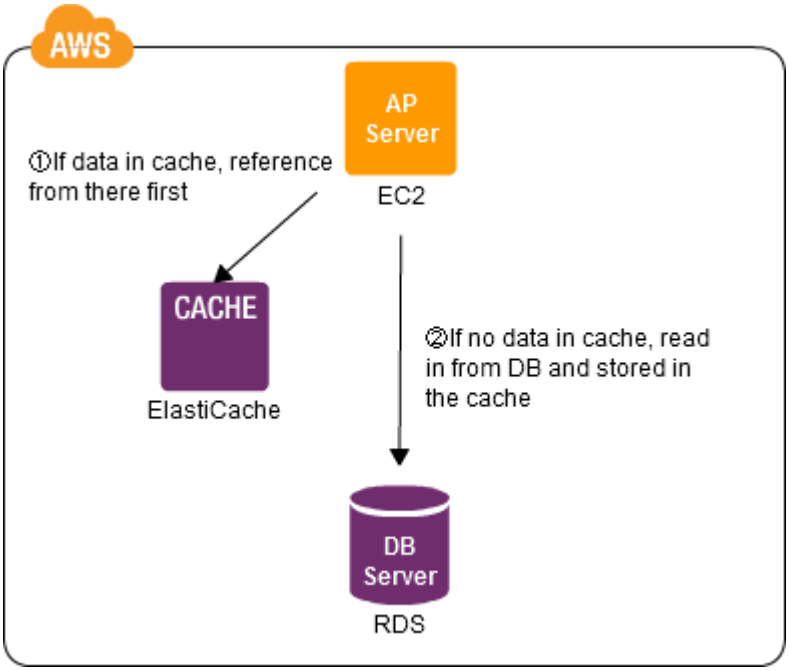
You can use this pattern to improve the performance of reading from databases by caching in memory data that is read frequently. This is a technique where data that has been used is placed in a cache, so that it can be read in from memory (rather than from the disk) the next time it is used. Typical examples of data that are cached are results of time-consuming queries, results of complex calculations, and so forth, in database operations.

Implementation

"ElastiCache" in AWS is a memory cache service. This service is provided with an automatic recovery function for the case of a failure.

- Prepare the memory cache. You may use ElastiCache, or you may use the open source memcached in an EC2 instance.
- When data is read in, data in the memory cache is referenced first. If the data is not there, it is read in from the database and stored in the cache.

Configuration



Benefits

- You can use high-speed memory for the cache to reduce the load of reading from the database, improving overall system performance.
- You can use ElastiCache to streamline the operation, and ElastiCache is robust to failures.

Cautions

- You should consider the trade-offs when caching query results. The proportion of reading in performing specific queries and writing the queries to the related tables is important. For example, if references are extremely frequent (many times per minute), but updates to the data are not so often (daily or even hourly), caching will be of value. On the other hand, you must be clever to prevent old data from remaining as-is in the cache. Reference Information: http://highscalability.com/bunch-great-strategies-using-memcached-and-mysql-better-together
- Using the cache may require you to modify the program that accesses the database.

Sharding Write Pattern

Problem to Be Solved

It is extremely important to increase the speed of writing to a relational database management system (RDBMS), but is a very difficult problem. Of course, you can use multiple database servers to produce performance beyond that which can be achieved through scaling-up, but how to to do so is an extremely important issue.

Explanation of the Cloud Solution/Pattern

"Sharding" is a technique for improving the performance of writing with multiple database servers. Fundamentally, databases with identical structures are prepared and divided using appropriate table columns as keys, to distribute the writing processes. The use of the RDBMS service provided in AWS Cloud lets you perform this sharding, to achieve an increase availability and operating efficiency.

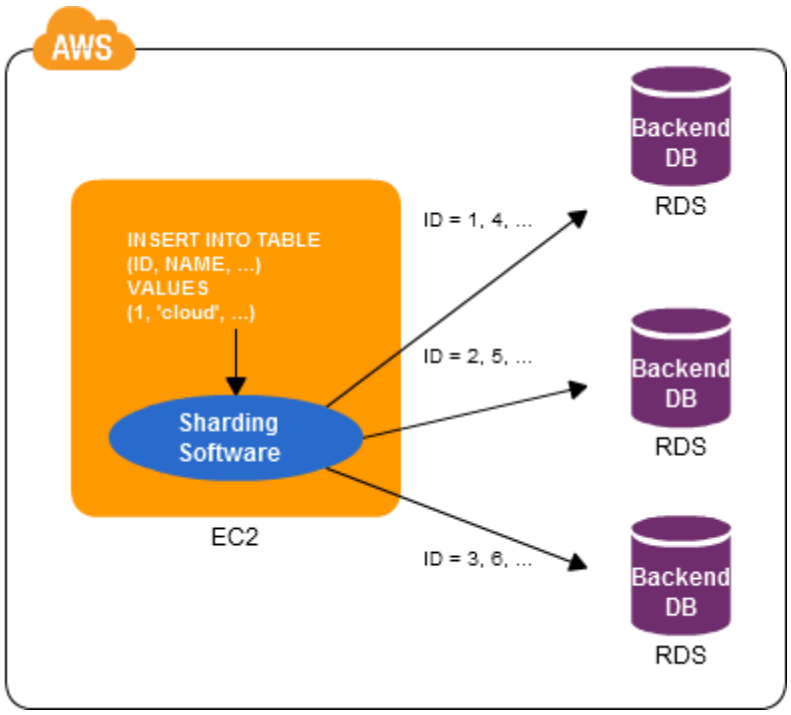
Implementation

You can use the AWS RDBMS service (RDS) in sharding backend databases to improve availability and operating efficiency.

Install sharding software, such as a MySQL server combined with a Spider Storage Engine on an EC2 instance.

- Prepare multiple RDSs and use them as the sharding backend databases. You can distribute the RDSs to multiple regions.

Configuration



Benefits

- You can achieve higher availability through using RDSs as backend databases for sharding (in Multi-AZ).
- You can distribute the backend databases to multiple regions to provide improved performance in various regions of a worldwide system.

Cautions

- When the backend databases are distributed to multiple regions, encryption may be required when communicating with the sharding software.

Patterns for Batch Processing

Queuing Chain Pattern

Problem to Be Solved

If performing sequential processing where processes running on multiple systems are linked together (for example, in the case of image processing, the sequential operations for uploading, storing, and encoding the image, creating a thumbnail, and the like), there will be a tendency for performance bottlenecks to result from having the systems tightly linked to each other. This tight linkage also complicates the recovery operations when there has been a failure. You can derive benefits in terms of performance and maintenance by loosely coupling the systems together whenever possible.

Explanation of the Cloud Solution/Pattern

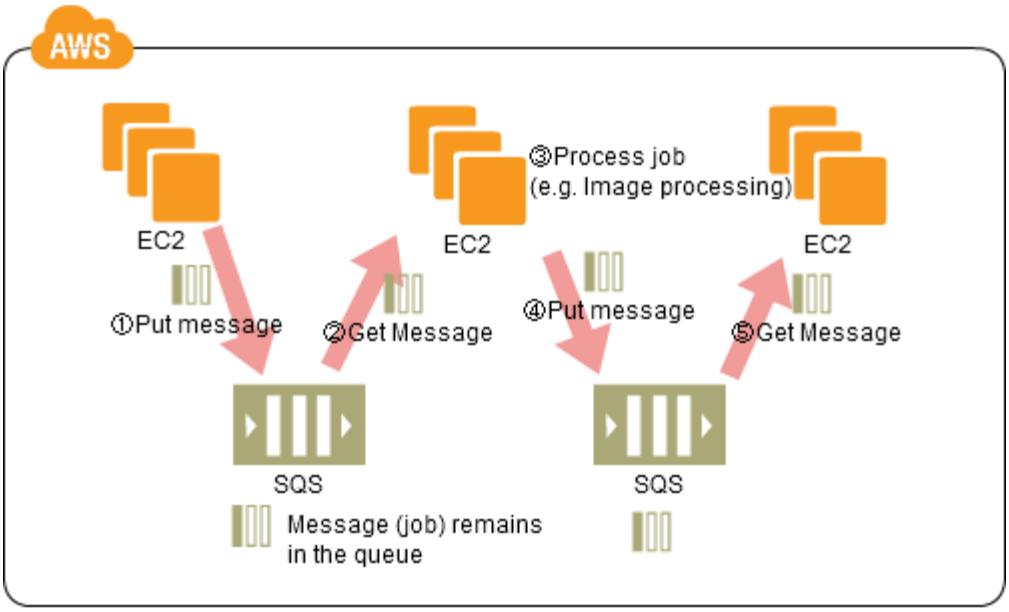
You can achieve this loose loose coupling of systems by between systems using queues, and then exchanging messages that transfer jobs. This enables asynchronous linking of systems. This method lets you increase the number of virtual servers that receive and process the messages, enabling parallel processing, when necessary, letting you resolve bottlenecks easily. Even if there happened to be a failure in a virtual server, the unprocessed processed messages would still remain in the queue, making it easy to restart processing when the virtual server is restored.

While you can use this pattern even without Cloud technology, the queue itself is provided as an AWS Cloud service, making it substantially easier for you to use this pattern than it has been in the past, given the nature of Cloud technology in enabling flexible provisioning of virtual servers.

Implementation

Use the the Amazon Simple Queue Service (SQS) to transfer a process from an EC2 instance for one process to the EC2 instance for the next process. SQS is the AWS queue service. The processing in EC2 is as follows: a job (message) is received --> the job is processed --> the job (message) is transmitted --> repeat. Depending on the nature of the job, you can run the processes on multiple EC2 instances.

Configuration



Benefits

- You can use asynchronous processing to return responses quickly.
- You can structure the system through loose coupling of simple processes (EC2 instances).
- You can handle performance and service requirements through merely increasing or decreasing the number of EC2 instances used in job processing.
- Even if an EC2 were to fail, a message (job) would remain in the queue service, enabling processing to be continued immediately upon recovery of the EC2 instance, producing a system that is robust to failure.

Cautions

In systems where processes must be performed in strict sequence, you need to be aware that, in SQS, the sequence of messages when drawing from the queue is not completely guaranteed.

Other

- You can combine this pattern with the Priority Queue Pattern.
- You can use the Amazon Simple Workflow (SWF) to provide complex workflows relatively easily, rather than just using simple queues.

Priority Queue Pattern

Problem to Be Solved

There are cases where a large number of batch jobs may need processing, and where the the jobs may need to be re-prioritized.

For example, one such case is one where there are differences between different levels of services for unpaid users versus subscriber users (such as the time until publication) in services enabling, for example, presentation files to be uploaded for publication from a web browser. When the user uploads a presentation file, the conversion processes, for example, for publication are performed as batch processes on the system side, and the file is published after the conversion. Is it then necessary to be able to assign the level of priority to the batch processes for each type of subscriber.

Explanation of the Cloud Solution/Pattern

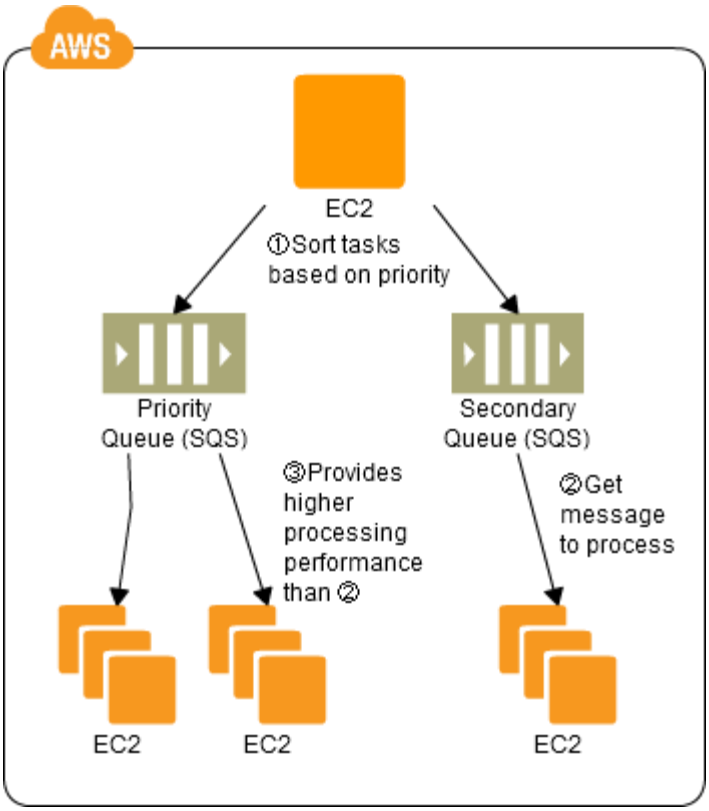
A queue is used in controlling batch jobs. The queue need only be provided with priority numbers. Job requests are controlled by the queue, and the job requests in the queue are processed by a batch server. In Cloud computing, a highly reliable queue is provided as a service, which you can use to structure a highly reliable batch system with ease. You may prepare multiple queues depending on priority levels, with job requests put into the queues depending on their priority levels, to apply prioritization to batch processes. The performance (number) of batch servers corresponding to a queue must be in accordance with the priority level thereof.

Implementation

In AWS, the queue service is the Simple Queue Service (SQS). Multiple SQS queues may be prepared to prepare queues for individual priority levels (with a priority queue and a secondary queue). Moreover, you may also use the message Delayed Send function to delay process execution.

- Use SQS to prepare multiple queues for the individual priority levels.
- Place those processes to be executed immediately (job requests) in the high priority queue.
- Prepare numbers of batch servers, for processing the job requests of the queues, depending on the priority levels.
- Queues have a message "Delayed Send" function. You can use this to delay the time for starting a process.

Configuration



Benefits

- You can increase or decrease the number of servers for processing jobs to change automatically the processing speeds of the priority queues and secondary queues.
- You can handle performance and service requirements through merely increasing or decreasing the number of EC2 instances used in job processing.
- Even if an EC2 were to fail, the messages (jobs) would remain in the queue service, enabling processing to be continued immediately upon recovery of the EC2 instance, producing a system that is robust to failure.

Cautions

Depending on the balance between the number of EC2 instances for performing the processes and the number of messages that are queued, there may be cases where processing in the secondary queue may be completed first, so you need to monitor the processing speeds in the primary queue and the secondary queue.

Job Observer Pattern

Problem to Be Solved

There is a technique for load distribution in batch processing where you can control job requests using queues, where the job requests that are queued are processed in parallel by multiple batch servers. However, because, if Cloud technology is not used, the number of batch servers that are provided must be up to handling the peak requirements, there will be excessive batch server resources in time bands other than the peak, which is harmful in terms of cost effectiveness. On top of this, the responsiveness may suffer if there is an unanticipatedly large load on the batch system.

Explanation of the Cloud Solution/Pattern

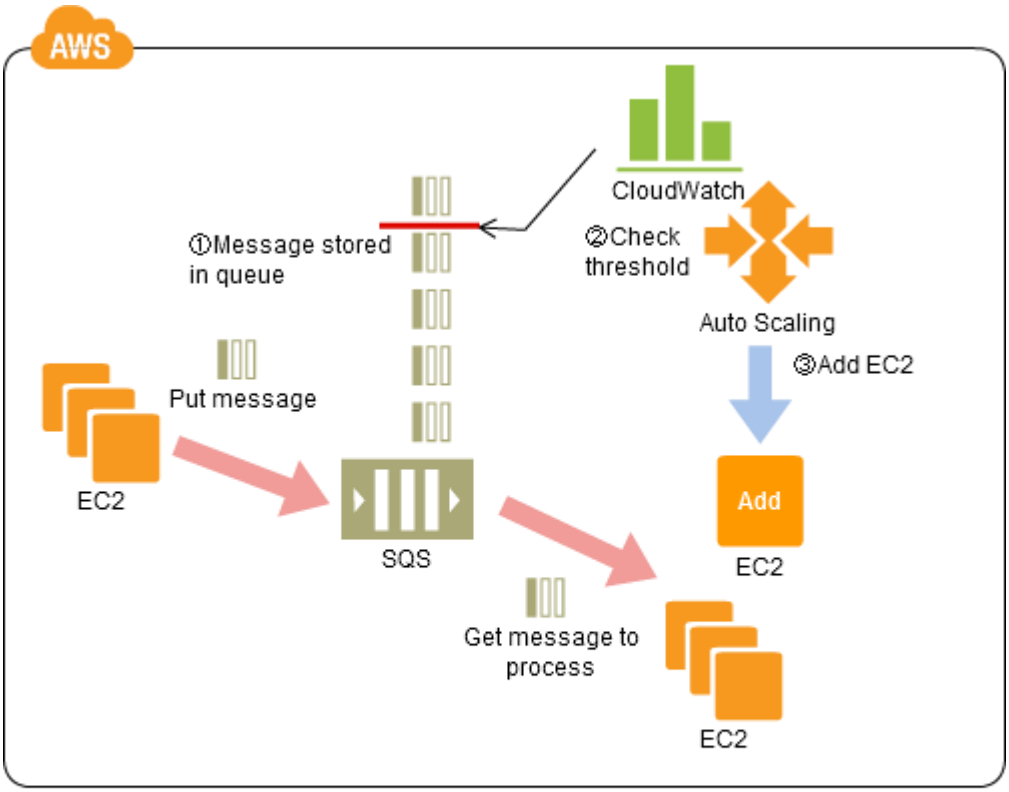
Because, in the past, the server resources could not the increased or decreased dynamically, the batch servers were provided considering the peak demand and the allowable cost. This resulted in problems with poor cost effectiveness and the inability to handle any unanticipated load. In the AWS Cloud, however, a system is provided where you can increase or decrease the number of virtual servers automatically by monitoring the load. You can use this system to increase or decrease the number of batch servers depending on the load, resulting in excellent cost effectiveness and enabling handling of unanticipated loads. Specifically, the volume of job requests (queued messages) is monitored, and batch servers are added or deleted automatically as required.

Implementation

AWS has a system known as "Auto Scaling" that is able to increase or decrease the number of EC2 instances automatically, and which works in cooperation with a resource monitoring tool known as "CloudWatch" to increase or decrease the EC2 instances in accordance with the value that is monitored. One thing that can be monitored by CloudWatch is the numbers of messages in the queues of the Simple Queue System (SQS) provided by AWS. You can use SQS to manage job requests and use Auto Scaling and CloudWatch to structure a system able to increase or decrease the number of batch servers automatically depending on the number of messages (job requests) within a queue.

- Enqueue job requests as SQS messages.
- Have the batch server dequeue and process messages from SQS.
- Set up Auto Scaling to automatically increase or decrease the number of batch servers, using the number of SQS messages (CloudWatch) as the trigger to do so.

Configuration



Benefits

- This lets you coordinate the number of EC2 instances for job servers with the number of jobs, improving cost effectiveness.
- This lets you reduce the overall time for executing jobs through parallel processing.
- Even if a job server EC2 instance were to fail, the SQS messages (job requests) would remain, enabling processing to be continued immediately upon recovery of the EC2 instance, producing a system that is robust to failure.

Cautions

- EC2 instances are charged by unit time, with a one-hour minimum, so one hour of time will be billed if an EC2 instance is launched and then shut down in a short period of time. Because of this, you need to be aware of the timing of launching and termination of instances.

Scheduled Autoscaling Pattern

Problem to Be Solved

In many systems, batch processes are performed at specific times, and often a scheduler (such as cron in UNIX) is used on a server that is always running. However, the time for performing the batch processes is actually short, wasting the server resources at other times, causing poor cost effectiveness. The question is how to use the resources of the batch servers efficiently in such applications.

Explanation of the Cloud Solution/Pattern

In the past, with batch servers that run at specific times, it has been necessary to assign one server to run at all times. If you were clever, you could increase the efficiency by having that server perform other functions (processes) as well. AWS Cloud allows you to limit the use of the virtual servers to only when they are required, making it practical to run the virtual servers only when actually performing batch processes.

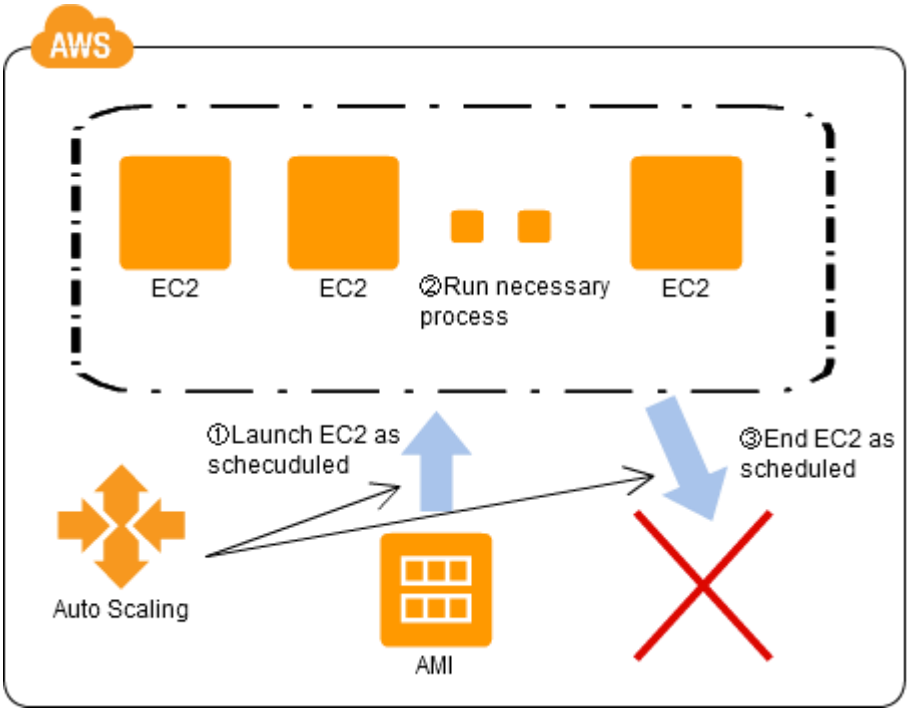
Batch processes that are executed at set times require a system for launching the virtual server at the set times. You can achieve this through using the scheduling system in AWS Cloud.

Implementation

AWS has a system known as "Auto Scaling" that is able to increase or decrease the number of EC2 instances automatically. Auto Scaling has a function that allows you to increase or decrease EC2 instances at specific times, which you can use for batch processes that are performed at specific times.

- Prepare an Amazon Machine Image (AMI) (a machine image) for executing the batch process when launched.
- In Auto Scaling, set up the EC2 instance to be launched at the specified time from the AMI.
- Set up the EC2 instance itself and Auto Scaling to terminate the EC2 instance after processing has been completed.

Configuration



Benefits

- The EC2 instance for processing batches at specific times does not have to run constantly. The EC2 instance is started only when a process is to be executed, reducing costs substantially.

Cautions

- When it is difficult to determine the timing with which the batch processing will be completed, you can use a method where the EC2 instance itself shuts itself down after completion of batch processing.
- EC2 instances are charged by unit time, with a one-hour minimum, so one hour of time will be billed if an EC2 instance is launched and then shut down in a short period of time. Because of this, you need to be aware of the timing of launching and termination of instances.

Pattern for Operation and Maintenance

Bootstrap Pattern

Problem to Be Solved

There has often been debate over how frequently machine images should be taken, in terms of operating efficiency, when you use a technique where you start up a server from a machine image, that is, when you use the Stamp Pattern.

In the Stamp Pattern, you can take the machine image after all set up (including middleware through applications) has been completed, with the middleware and applications started up and are running.While this lets you start up the virtual server extremely quickly, if you need to upgrade one of the middleware products, or you need to change a setting in an application, you will have to rebuild the machine image.

Explanation of the Cloud Solution/Pattern

AWS Cloud not only lets you create the machine image easily, but lets you set up the parameters at the time of start up. The use of this function lets you pass required parameters to the server configuration to have the server itself acquire the required settings when the server starts up, enabling you to create a machine image for executing installation, startup, and setup.This frees you from having to re-create the machine image each time there is a version update of an individual package.

Implementation

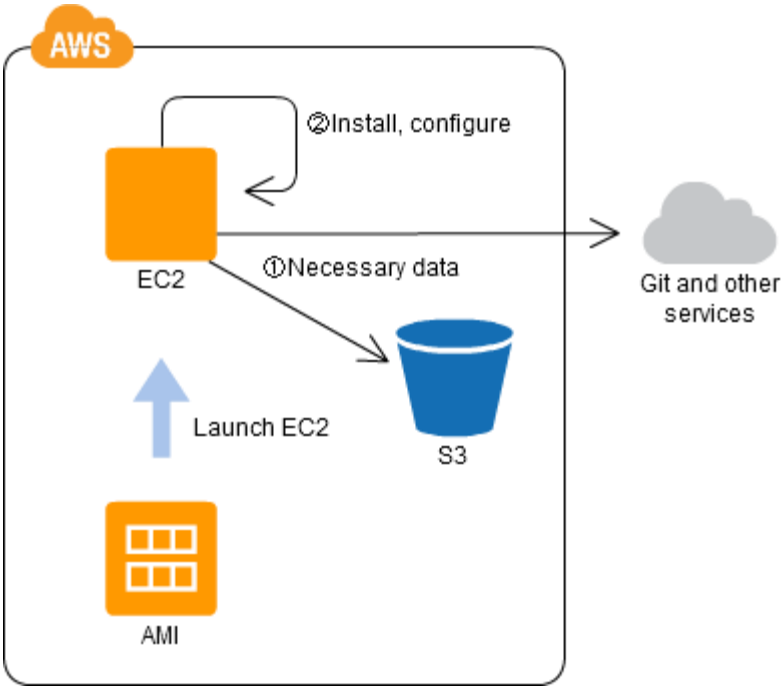
Put some thought into the creation of the Amazon Machine Image (AMI). Specifically, place the various parameter files required for initializing the EC2 instance (the virtual server) in Amazon Simple Storage Service (S3) (Internet storage), and then, when starting up an EC2 instance, have the EC2 instance read out the parameter files to build itself. You can place the parameter files in a repository such as Git.

Note that Amazon Linux has an initialization function known as cloud-init, enabling automatic execution through an initialization script that is written to the user data region.

- Prepare the data required for bootstrapping in S3 or another repository such as Git. Create an AMI that includes a bootstrap.
- Startup an EC2 instance from the specific AMI that includes the bootstrap.
- At the time of start up, the EC2 instance itself will acquire, install, startup, and setup the required packages.

- When necessary, dynamically replicate the servers based on the AMI.

Configuration



Benefits

- This eliminates the need for you to rebuild the AMI when there is a revision to a package that must be installed.
- Passing parameters, and the like, at the time of startup makes it easy for you to change dynamically the sequencing and details of set up at the time of startup.

Cautions

- You have a broad scope of choice when it comes to the layer for using a static AMI, and the layer above which to set up dynamically at the time of startup.
- When considering which pattern to apply, consider the trade-offs with the Stamp Pattern.

Cloud DI Pattern

Problem to Be Solved

In large-scale systems, large numbers of servers are added when there is growth in, for example, the number of accesses. Manually performing all of the installation and setup operations required for configuring the individual servers would be extremely time-consuming in such a case, and often would be difficult to achieve in a limited time. While you may use a technique where system management tools are used to automate the server configuration, this approach has issues in terms of cost.

Explanation of the Cloud Solution/Pattern

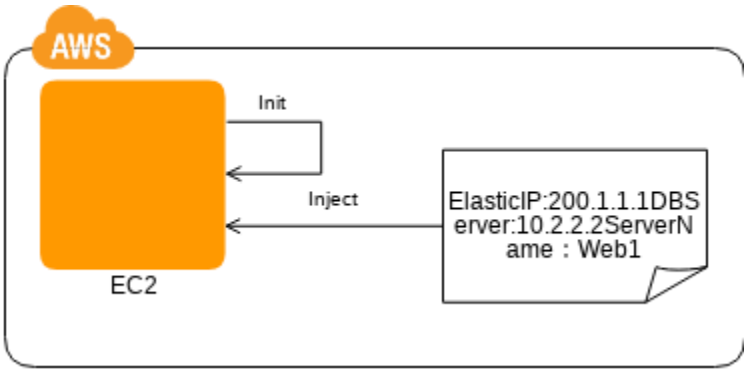
The internal structure of a server may be configured automatically, depending on the purpose for the server, when launching a virtual server. In particular, you may use the Scale Out Pattern and Scheduled Autoscaling Pattern if you want to automate these operations. In these cases, you may use the Bootstrap Pattern, but if there is lots of information you want to locate externally (for example, an IP address of a database connection destination, a server name, a security number, or the like), you can initialize the server more flexibly through the use of the **Cloud DI Pattern**.

Implementation

There is a function that lets you add arbitrary tags to an EC2 instance when starting up the EC2 instance. You can use this function to readout tag information when starting up an EC2, to perform the setup accordingly.

- Set tags as information that is unique to the EC2 instance. (For example, set up an Elastic IP (EIP) as a tag.) * Set up so as to launch an application for acquiring the tags when an EC2 instance is launched.
- In the application, initialize the EC2 using the tag information (where the EIP that has been set will be assigned automatically to the EC2 instance).

Configuration



Benefits

This provides unique settings to the general-use base image that uses the Stamp Pattern or the Bootstrap Pattern.

- Because the parameters are set using the tag information, you can set and confirmed them easily using the Amazon Management Console.
- You can perform the setup automatically to reduce errors during operation.
- You can use this not only when configuring an EC2 instance, but also when creating a system for producing Amazon Machine Images (AMIs) and snapshots automatically.

Cautions

- Sometimes there is a limit to the number of characters that can be contained in a tag. In such a case, set the tag to information pointing to the information you wish to transfer, such as an S3 URL or a network file path.

Other

- In addition to tags, you can also use metadata, known as "user data," to pass information.

Stack Deployment Pattern

Problem to Be Solved

Commonly a test environment and a staging environment are provided for system development and operations/maintenance. These environments are not used at other times, so the use of the same number of servers as in the actual environment would have poor cost effectiveness. The use of virtual servers improves the cost effectiveness.

When the system is complex and there is a large number of virtual servers, however, the operations to construct the environment and to launch and terminate the related virtual servers, for example, may become complex and laborious. The more complex and laborious, the more time it will take you, and the more likely it will be that you will make a mistake.

Explanation of the Cloud Solution/Pattern

You can prepare a template for starting up a group of servers and start up the servers automatically all at once following the template. You can list, in that template, the Cloud components that are required in the environment that is being configured, and then configure the environment following that template, to establish complex systems easily and without errors.

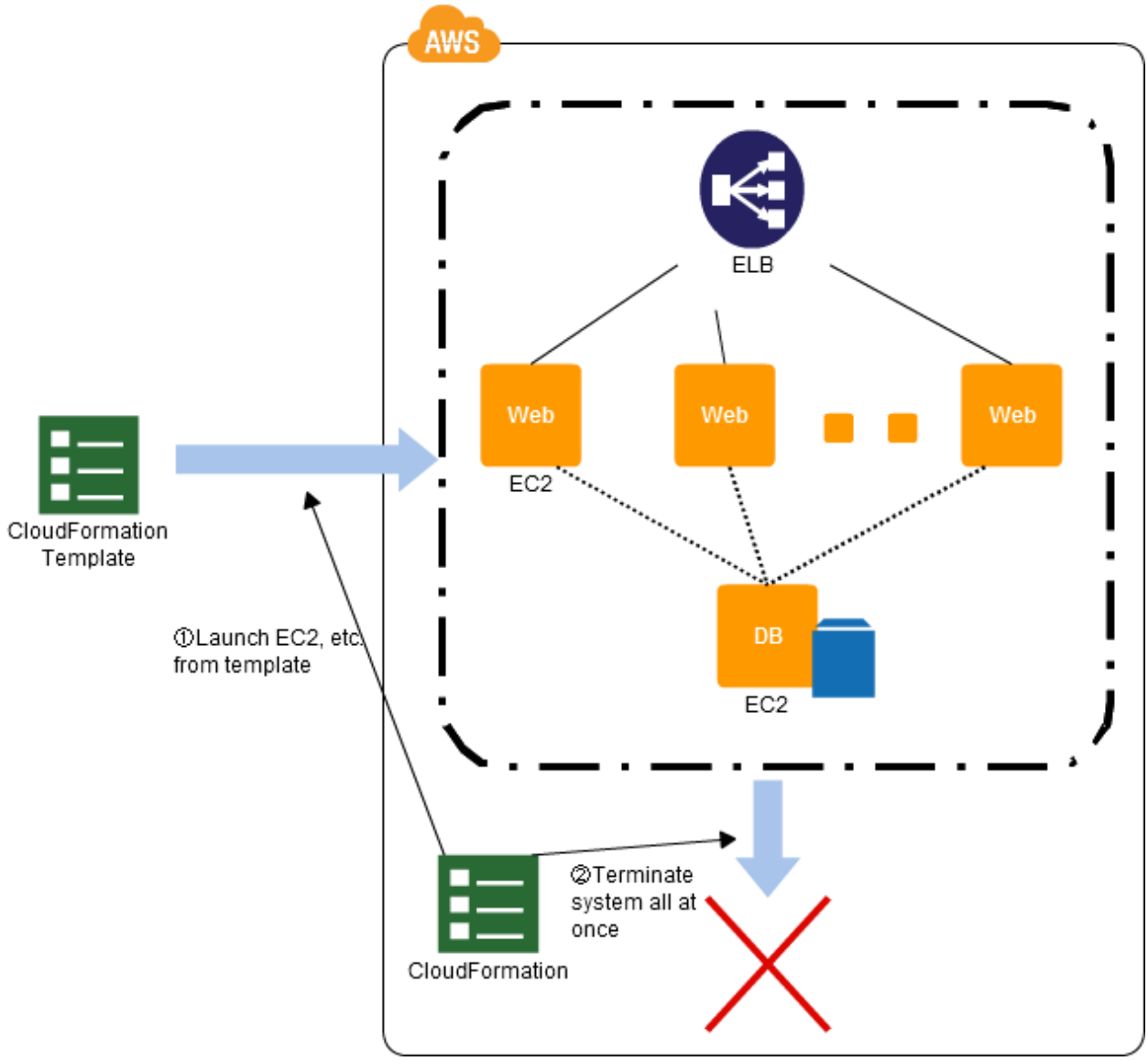
Implementation

You can use a service known as "CloudFormation" in AWS to start up the system (EC2 instances, and the like) from a "CloudFormation template" that lists server resources, and the like.

- Create the CloudFormation template from a preexisting or newly created environment.
- When using the system, startup the system (the EC2 instances, or the like), all at once from the CloudFormation template.
- When terminating the system, use the CloudFormation template to delete the system all at once.

When creating the template, you can also use the CloudFormer tool tocreate a template from an existing system.

Configuration



Benefits

- You can create a stack template (CloudFormation template) for starting up, in the right sequence, even virtual servers (EC2 instances) that require a specific startup sequence, thereby decreasing operating errors.
- You can also manage the system configuration history through versioning the stack template (the CloudFormation template).
- This makes easy for you to not just construct the environments, but deconstruct the environments as well.

Cautions

- If you update the system (the virtual servers, or the like), you will have to update the Amazon Machine Image (AMI) ID in the stack template (the CloudFormation template) as well.

Other

This pattern not only allows you to simply copy a system, but also to move from the classic EC2 environment to a Virtual Private Cloud (VPC) environment that is connected to the user enterprise by a dedicated network.

- Use a tool (CloudFormer) to create a CloudFormation template of an existing system.
- Construct a transfer-destination VPC environment (network system and subnets).
- Register, in the CloudFormation template, the ID of the VPC and the IDs of the subnets that have been built.
- Startup the system using the CloudFormation template.

Server Swapping Pattern

Problem to Be Solved

When there is a server failure, it is followed by attempts at recovery. While you can consider many causes for failures, often there is no problem with the disk. In such a case, you can swap the problem-free disk to another server to recover quickly. Unfortunately, in a datacenter performing such a swap, including the operations to do so, procurement of a substitute server, installation of the disk, and so forth, takes a lot of work.

Explanation of the Cloud Solution/Pattern

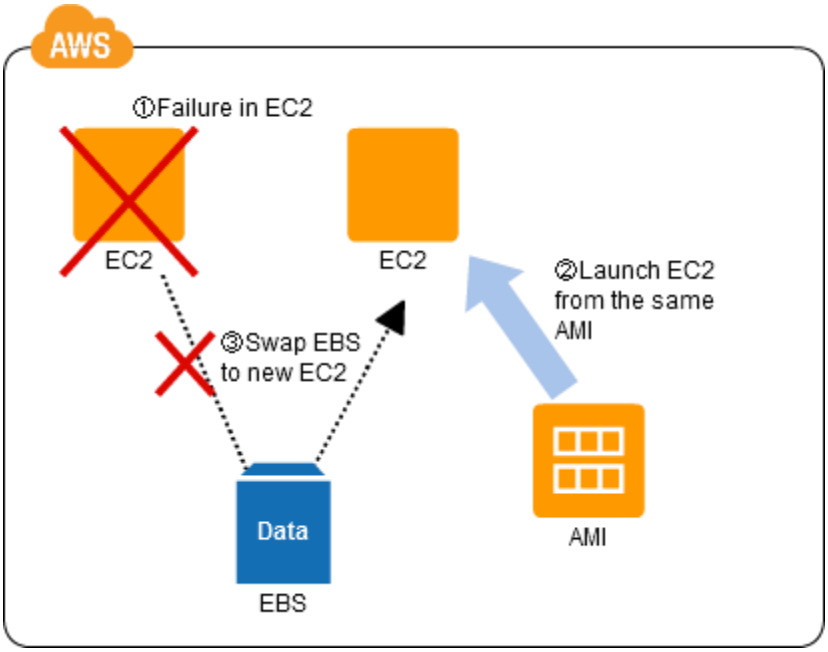
With the Amazon Cloud, the servers are virtual servers, enabling use of virtual disks that can be swapped. If there is a failure in a virtual server, you can swap the disk from the virtual server that has experienced the failure to another virtual server, to restore the system. When compared to physical servers, this eliminates procurement of a machine and also movement by a technician, so the recovery will be fast.

Implementation

The Elastic Block Store (EBS) exists independently of the EC2 instance, and can be swapped between EC2 instances, allowing you to start up a substitute EC2 instance, and to swap the EBS there, when a failure has occurred.

- Attach the EBS to the EC2 instance and store data there as always.
- When there is a failure in an EC2 instance, start up a virtual server from the same Amazon Machine Image (AMI), and swap, to the new EC2 instance, the EBS used for the data for the failed EC2 instance.
- Set up the middleware and file system again (by, for example, making symbolic links to the newest data), to restore to the state immediately prior to the failure.
- You can use monitoring software (Nagios, Zabbix, Heartbeat, or the like) to automate these processes.

Configuration



Benefits

- This lets you to recover to the state immediately prior to the failure of the EC2 instance.
- You can swap the root disk to rapidly recover the operating system (OS) as well.

Cautions

- Considering the possibility of a failure of the EBS as well, make sure to make backups by taking snapshots, for example.
- You can use this together with Floating IP Pattern to swap not only the the EBS, but the static IP address as well.

Monitoring Integration Pattern

Problem to Be Solved

Monitoring (of services and resources, and the like) is necessary in system operation. A monitoring service is provided by the Amazon Cloud. However, because the monitoring service in the Amazon Cloud is unable to monitor the internal workings of a virtual server (the operating system, middleware, applications, and so forth), you need to have an independent monitoring system. This is a problem in that it results in multiple monitoring systems, and complicates the operations (especially the monitoring operations).

Explanation of the Cloud Solution/Pattern

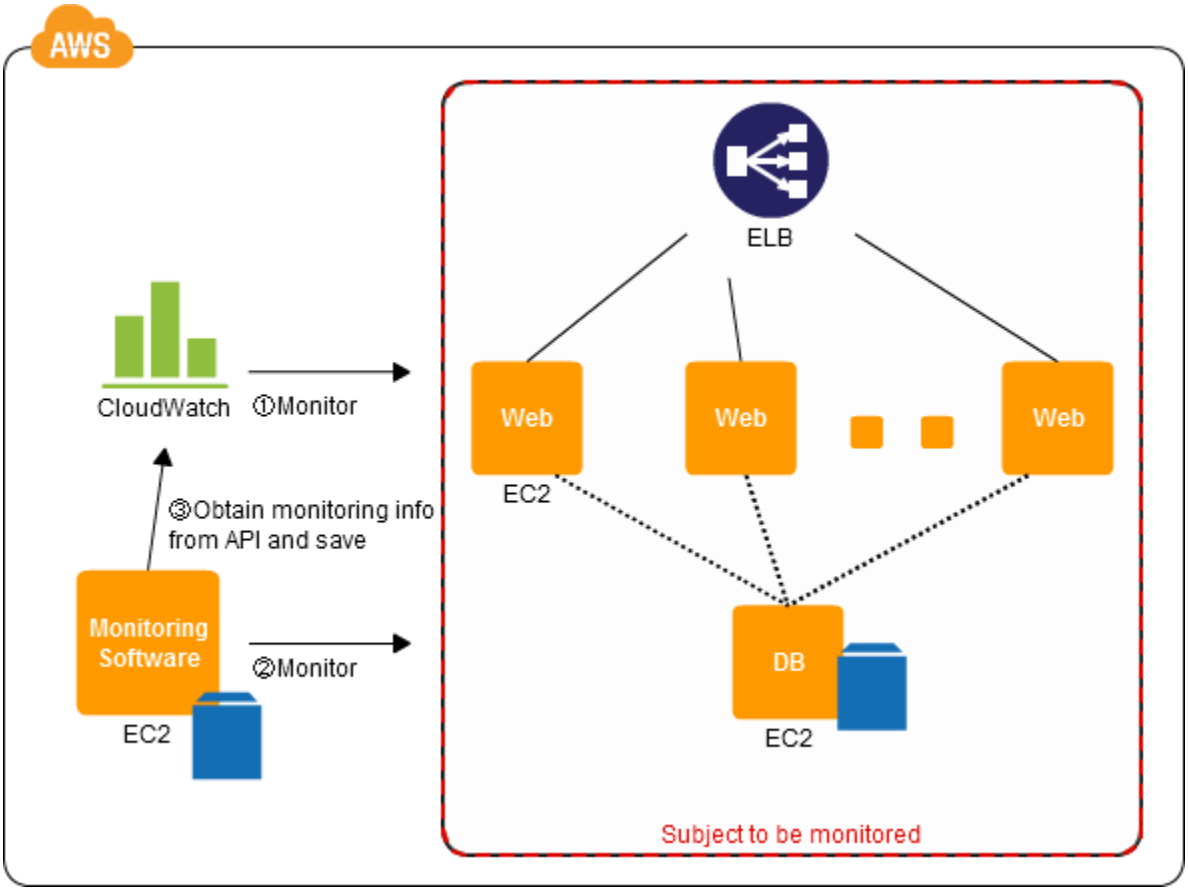
The virtual server and the like are monitored by the AWS Cloud monitoring service, but you need to use your own system to monitor the operating system, middleware, applications, and so forth. The Cloud monitoring service provides an API, enabling you to use your monitoring system to perform centralized control, including of the Cloud side, through this API, to obtain information from the Cloud monitoring system.

Implementation

Install monitoring software on the EC2 instance, to obtain monitoring information from the "CloudWatch" monitoring service of AWS.

- Install monitoring software (Nagios, Zabbix, Munin, or the like).
- Use a plug-in to obtain monitoring information using the CloudWatch API and to write that information to the monitoring software.
- Use the plug-in to perform monitoring, including the information from AWS.

Configuration



Benefits

- This lets you monitor the Cloud resources in the same way as monitoring of the operating system, middleware, and applications through your own system.
- While data is stored for two weeks in CloudWatch, you can extend the time period for storing data through acquiring the data using monitoring software.
- The use of monitoring software in relation to AWS resource status notifications let you customize the emailed reports, for example.

Cautions

- An API usage fee is required for the monitoring software (plug-in) to obtain data periodically from CloudWatch through the API.

Web Storage Archive Pattern

Problem to Be Solved

Logs and backup files that are produced in large volumes in the individual servers must be stored for some period of time. Provision of a high-volume disk for that purpose is inefficient in terms of costs. In particular, in a system that is growing, it is difficult to estimate the sizes of the files that will be stored (in capacity planning). You can archive the logs from the individual files and rotate them over short intervals to eliminate the maintenance work required in enlarging the disks of the individual servers. However, there will still be the same issue of capacity planning for the shared storage.

Explanation of the Cloud Solution/Pattern

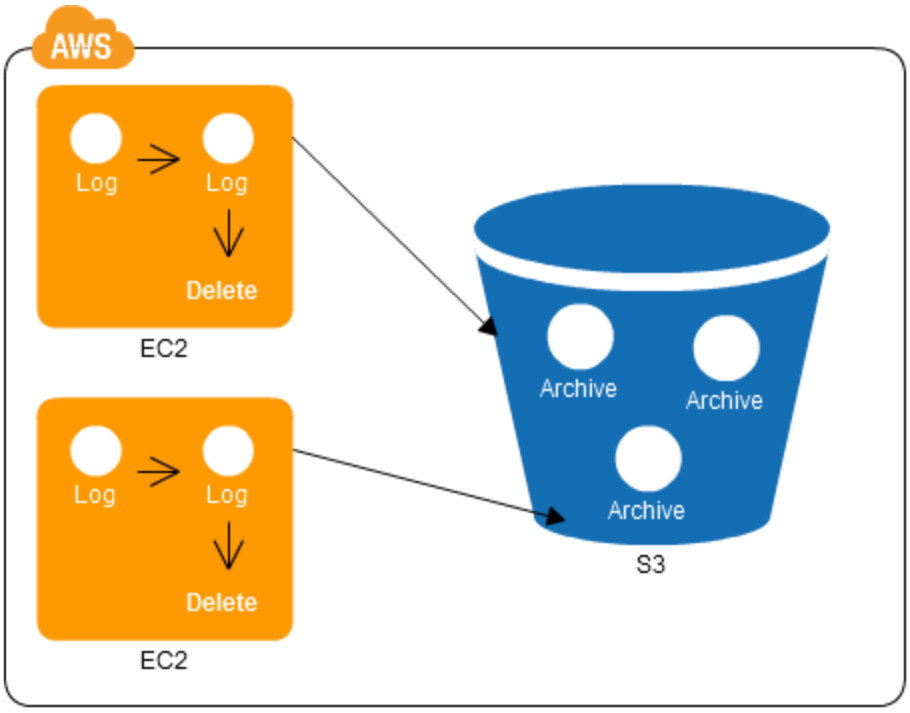
AWS provides Internet storage, which has essentially limitless capacity, and you can use this Internet storage as a place to archive logs. This eliminates worries about disk enlargement maintenance or capacity planning in advance, making it easy to archive logs in shared storage. (You still need to perform capacity planning in relation to costs, however.)

Implementation

The Amazon Simple Storage Service (S3) provides high reliability and durability, and is well-suited as a location for storing logs. You use a tool (such as s3cmd or s3sync) to upload to S3 easily.

- Use rotation software (such as logrotate) to rotate the various types of logs outputted by the EC2 instances. Store the logs to S3 with a specific rotation timing. (Write a procedure for uploading to S3 in the rotation script.)

Configuration



Benefits

- You can store the logs to S3 to eliminate the need to worry about disk space, enabling the logs to be archived without the risk of loss due to a failure.
- You can use the same system for saving backup files as well.
- While in Elastic Block Store (EBS) charges are based on size, in S3 charges are based on the amount of use.This enables more reasonable operations.

Cautions

- If there is a failure in the EBS prior to log rotation, the log from the previous rotation will be lost.
- When Auto Scaling is used, you will need to store the log in S3 prior to shutting down the EC2 instance.

Weighted Transition Pattern

Problem to Be Solved

You may wish to transfer your system as a whole from one region to another region. For example, you may want to move from an on-premises data system to the AWS system, or move from one region of the AWS Cloud to another region. You want the transfer to go as smoothly as possible, of course, without changing the domain name of the system and without having to take the system down.

Explanation of the Cloud Solution/Pattern

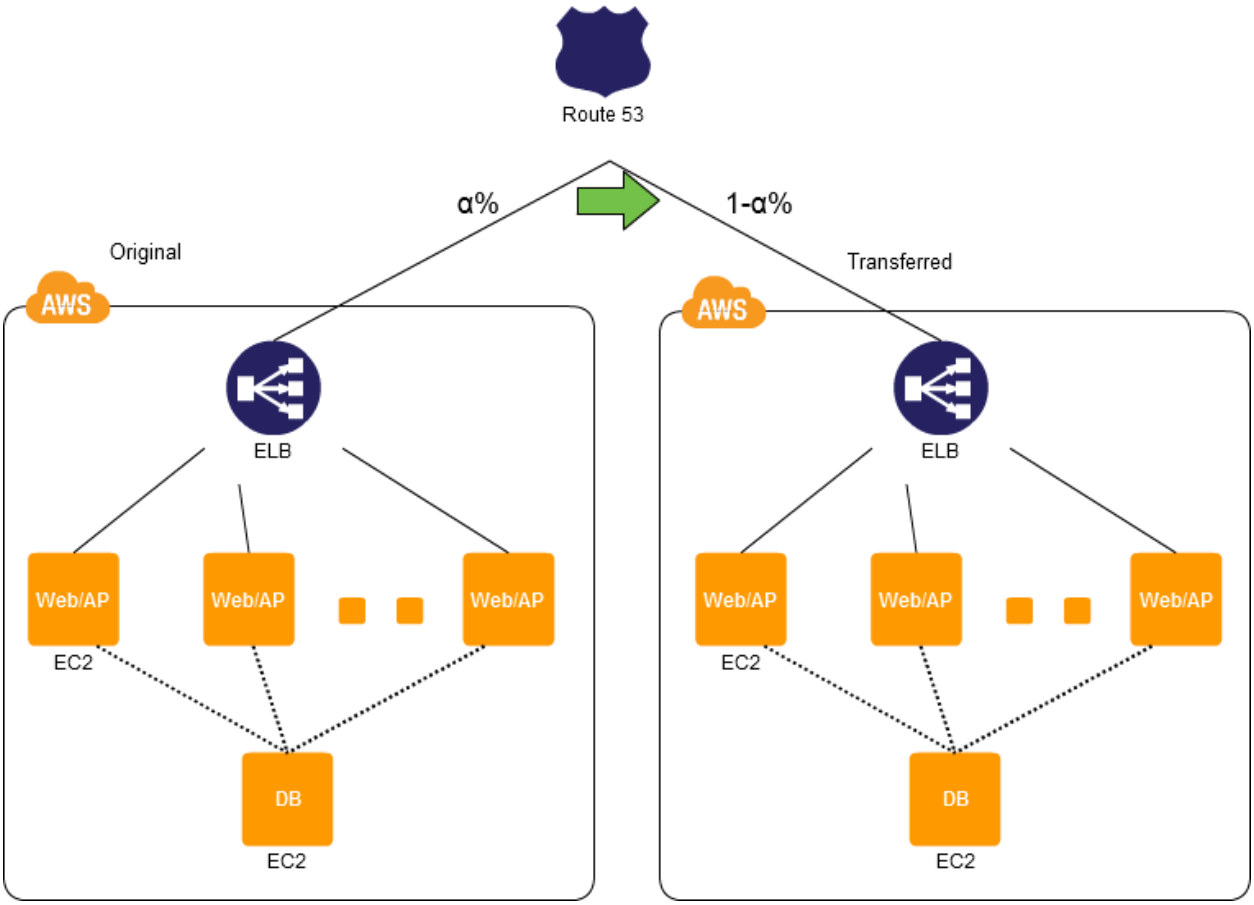
In transferring the system as a whole, without changing the domain name, there is a technique where the "Weighted Round Robin" function within the Domain Name System (DNS) server is used to switch from an existing system to a new system when performing name resolution. At first, you allocate only a small portion to the new system, and then, if there is no problem, you increase the distribution to the new system gradually. You can use the DNS service provided by AWS to set up the Weighted Round Robin easily, thus letting you perform the transfer operation without having to think much about the operation of the DNS server.

Implementation

You can transfer the system using the Weighted Round Robin function in AWS Route 53 (a name resolution service).

- Create a record set for the existing system in a Route 53 hosted zone.
- Provide a new system with a structure that is identical to that of the existing system.
- Create a record set in the new system within a Route 53 hosted zone, and set up the weighting.
- Initially make an extremely small allocation to the new system (for example, 1%), and if the system operates without problems, gradually increase the weighting until ultimately the weighting switches over entirely to the new system.

Configuration



Benefits

- This lets you transfer to a new system without modifying the existing system.
- Because this lets you switch to the existing system while controlling traffic, it also lets you switch back immediately if there is some sort of problem.

Cautions

- If data synchronization of a database is necessary at the time of the transfer, you have to think about that separately.

Hybrid Backup Pattern

Problem to Be Solved

Since the Great Eastern Japan Earthquake, there have been an increasingly large number of disaster recovery (DR) systems structured, and backup data has been stored in datacenters that are geographically separated from a firm's datacenter for the purposes of protecting data and ensuring continuity of operations in the event of a disaster. Unfortunately, building a new datacenter is expensive and time-consuming, and thus is undesirable in terms of cost effectiveness.

Explanation of the Cloud Solution/Pattern

Back up by linking on-premises and the AWS Cloud.

Implementation

Transmit data using the Internet, a VPN, or dedicated lines.

- Use the Amazon Simple Storage Service (S3).

Configuration

Benefits

- You can construct a DR system without contracting with a new datacenter.
- Enables use of high-durability Internet storage.
- Enables a reduction in cost through stopping the system until a disaster occurs.

Patterns for Network

OnDemand NAT Pattern

Problem to Be Solved

In secure systems, outbound access to the Internet by individual servers is usually prohibited. This makes maintenance operations that require accessing the Internet, such as updating the operating system package, impossible. One method to solve this problem is to provide Network Address Translation (NAT) for the Internet connection, to access the Internet through the NAT. The conditions for individual servers to access the Internet are controlled by the NAT. However, it is only during maintenance (such as updating the operating system package) that NAT is needed. The NAT it is not required at other times, so the NAT resources will sit idle the vast majority of the time, which is wasteful.

Explanation of the Cloud Solution/Pattern

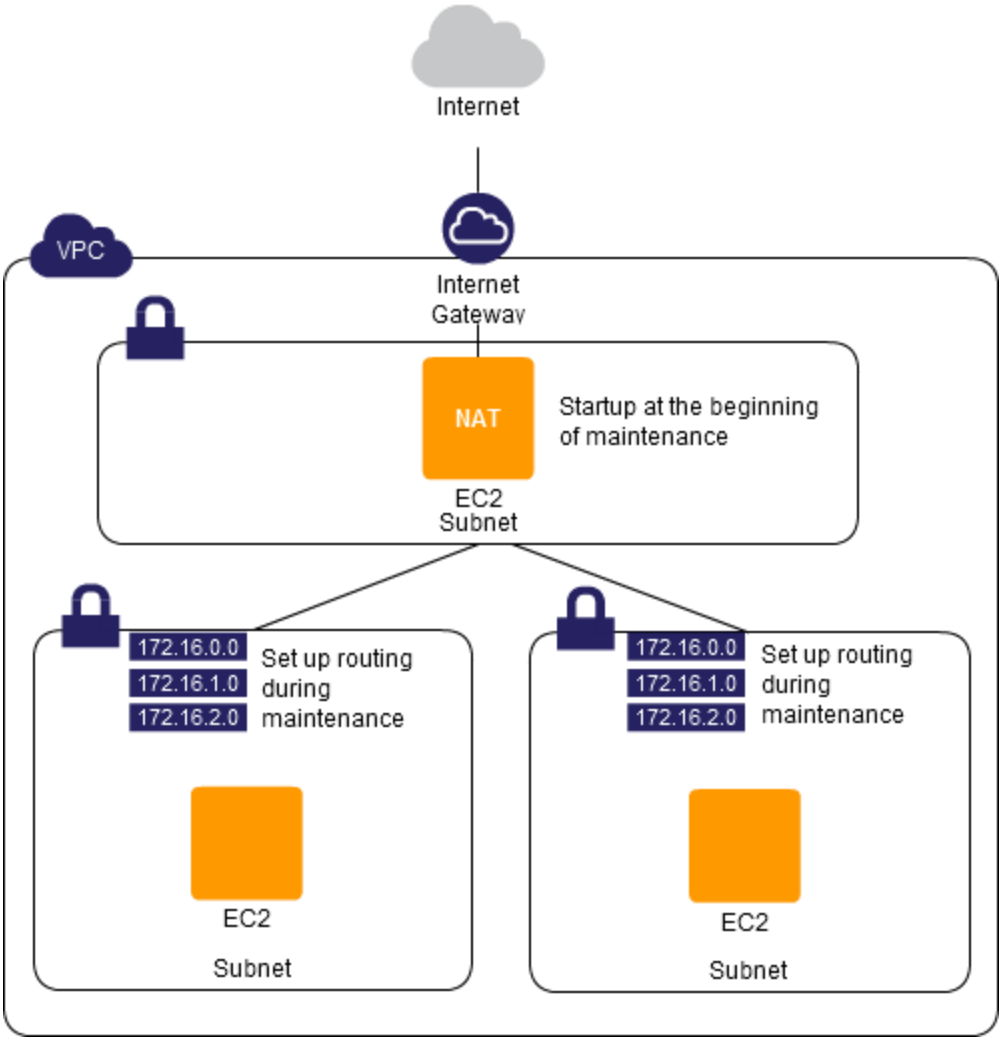
In the past, the assumption has been that after purchasing, server hardware would continue to be used on a permanent basis. That is, there would still be the same expenses for purchasing and operation, even if the servers were no longer used. It would be extremely difficult to operate, under temporary usage fees, a server that would be used only temporarily. However, most of the virtual servers on the AWS Cloud are billed by time of use. Given this, producing the NAT in a virtual server that is launched only at the time of maintenance (such as when updating the operating system package), and shutting it down otherwise, improves cost effectiveness. The AWS Cloud provides many APIs for launching and terminating virtual servers. You can use these APIs to automate launching and termination of NATs (virtual servers).

Implementation

The Virtual Private Cloud (VPC) that performs virtual networking on the AWS Cloud has a function for creating an NAT instance. There is also a function for setting up routing for each subnet, allowing you to route EC2 instances within a subnet through the NAT instance.

- Prepare an NAT instance on a VPC.
- Startup the NAT instance at the beginning of maintenance (when it will be necessary to access the Internet), and set up the NAT instance with subnet routing.
- When maintenance has been completed, delete the settings for the NAT instance from the routing, and then stop/delete the NAT instance.

Configuration



Benefits

- This lets you maintain system security because there is no routing of access from within to the Internet except for during maintenance.
- The NAT instance operates only during use, reducing costs.

Cautions

- Because, at the time of maintenance, operations are performed ranging from starting up the NAT instance through adjusting the routing of the subnets, it is safest if you automate these operations through a script, so that there will be no operator errors.

Backnet Pattern

Problem to Be Solved

The servers that are open to the Internet and that are accessed by a large number of non-specific users (such as web servers) are often accessed through that same network interface for the purposes of management. However, when a high security level is required, this is problematic because use of the same network interface as access that cannot be trusted should be avoided, meaning that you should separate these interfaces.

Explanation of the Cloud Solution/Pattern

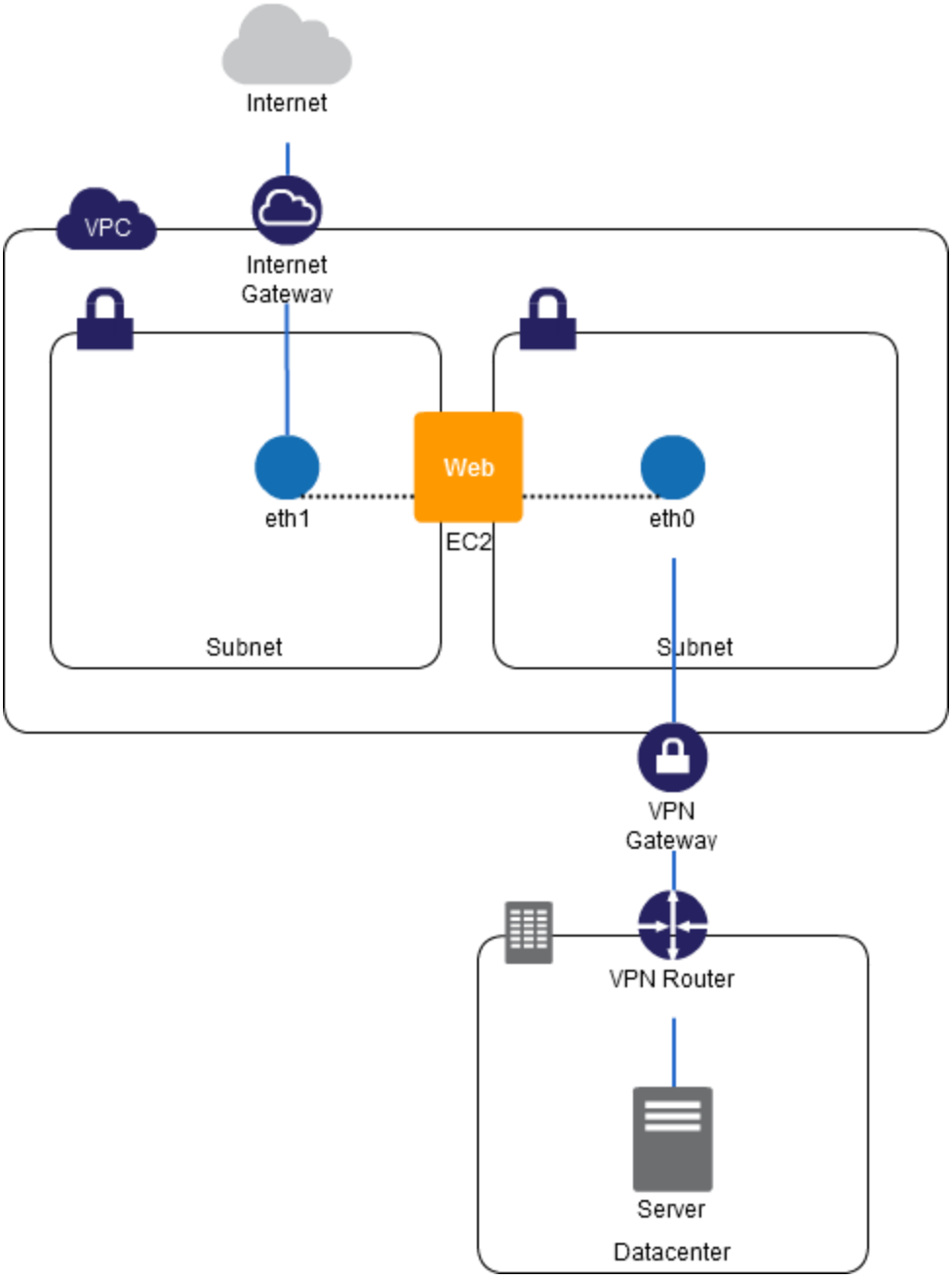
A technique that is often used in structuring and managing ordinary systems is to provide multiple network interfaces on a web server that is open to the public, to separate the Management Network Interface from the public network interface. The provision of the Management Network Interface is known as a "Backnet," and the provision of this Backnet can reduce the network risk in management.

Implementation

In a Virtual Private Cloud (VPC), you can use two virtual network interfaces or "Elastic Network Interfaces" (ENI) for an EC2 instance. One is set up as a public network interface, and the other is set up as an Management Network Interface.

- Install a web server or database server on an EC2 instance and prepare the two ENIs (outward-looking and inward-looking).
- Have one of the ENIs belong to the public network of the VPC and route 0.0.0.0/0 (all traffic) to the outward-looking Internet gateway.
- Have the second ENI belong to the private subnet of the VPC, and route 0.0.0.0/0 (all traffic) to the VPN gateway that is connected to, for example, the company intranet. Use this also for SSH access and management/logs.
- Because you can provide a separate security group for the individual virtual network interfaces, set up the one virtual network interface so as to allow traffic on Port 80, and the other network interface so as to allow traffic on Port 22.

Configuration



Benefits

- Because there is no port for the outside Internet that can be accessed as SSH, security is high.
- There is a clear distinction between the outward-looking and inward-looking virtual network interfaces, preventing errors in procedures.

Cautions

- Because you also have to set up a virtual private network (VPN), the operating cost will be somewhat higher.

Functional Firewall Pattern

Problem to Be Solved

Multi-tier access constraints using firewalls have been used as security measures that have been routinely employed even in systems of the past. However, when the number of access control rules becomes large, the setup of the firewall becomes complex, with many settings. The operating cost rises commensurately. If the rules for the firewall cannot be grouped, maintenance becomes complex as well, increasing the probability that there will be an error.

Explanation of the Cloud Solution/Pattern

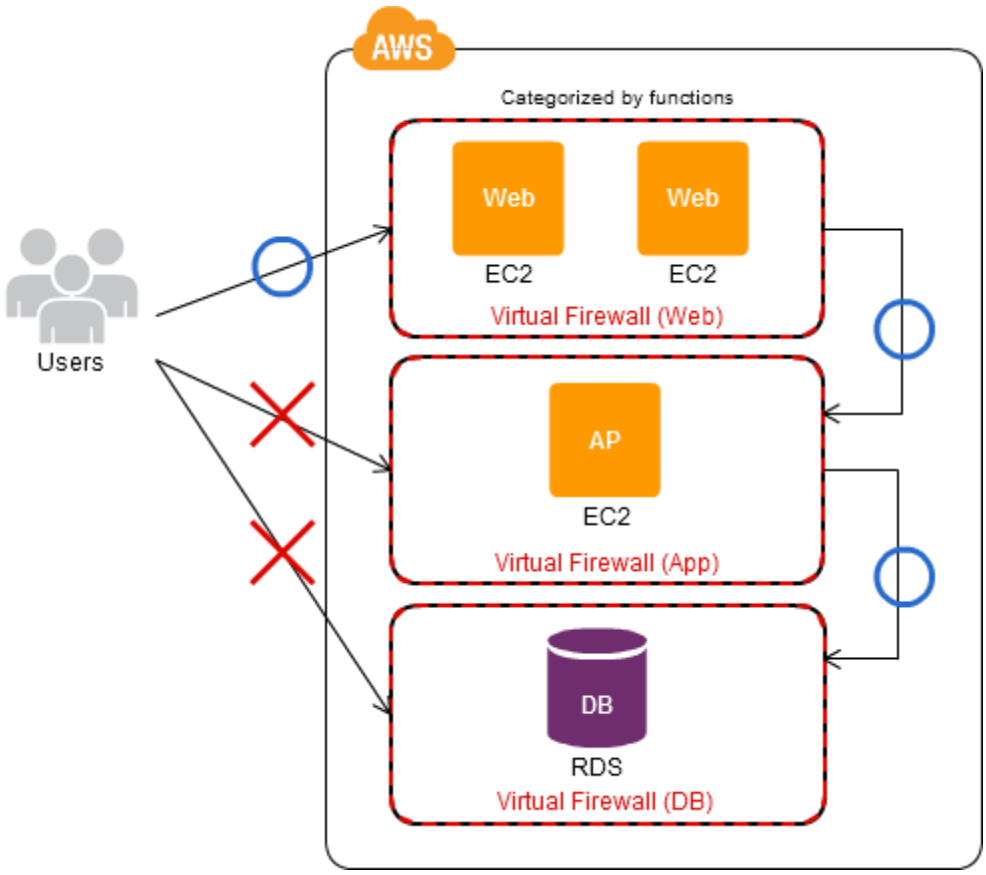
In the past, firewalls have used dedicated machines, and usually the rules have been controlled without grouping. Even if grouping has been possible, it has been difficult to apply by the server unit. In the AWS Cloud, however, the firewall has also been virtualized, enabling a more flexible setup. You can group the rules, and the setup can be by the group unit, or applied to individual servers. By having the group units be by the individual functions (web servers versus database servers, and so forth), the setup of the functions can be controlled centrally, within the groups. Application to virtual servers can also be through functional group units, simplifying the maintenance of access control and reducing the possibility of error.

Implementation

AWS lets you use virtual firewalls known as "security groups." You can construct security groups for each function, with rules under centralized control. Once they are set up, you can apply the security groups to the EC2 instances that are divided by functional units, enabling grouping by individual functions.

- Group the EC2 instances by individual functions (web layer, application layer, database layer, and so forth).
- Make a security group for each EC2 instance group, and set it up in the EC2 instances.
- Set up security groups for IP addresses, port numbers, and so forth.

Configuration



Benefits

- Multi-tier access control improves security. The EC2 virtual servers are grouped by individual functions, eliminating the need to change the virtual firewall settings even when using the scale-out pattern.

Cautions

- While several different definitions are possible because virtual firewalls are logical entities, creating too many makes them difficult to understand, so you need to think about the granularity of the groups.

Operational Firewall Pattern

Problem to Be Solved

When the scope of a system is large, there will be multiple organizations for performing development and maintenance. For example, you may have a division between the company that performs system development and the company that performs log analysis and operation/monitoring. When defining rules for firewalls for groups of individual functions, if there is a change in the access origin, or if access itself becomes unnecessary, you will need to update the rules that have been grouped by the functions. Not only does such setup work take time, but it interferes with the possibility of performing centralized control of which organization can access which system.

Explanation of the Cloud Solution/Pattern

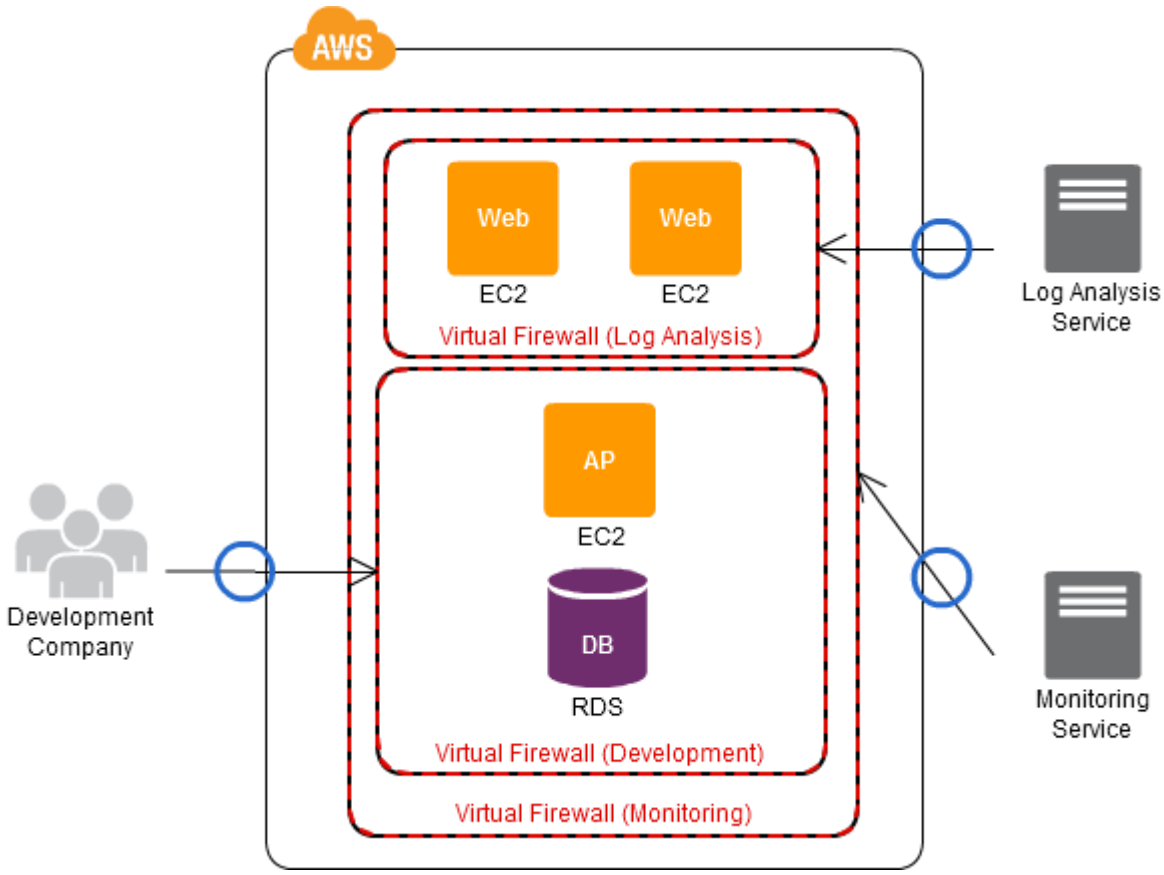
In the past, firewalls have used dedicated machines, and rules were also grouped together by function and written (controlled).In the AWS Cloud, the firewall is virtualized, enabling a more flexible setup. The rules can be grouped and set or applied to servers by group units. Having these units known as groups be organizations enables centralized control of the setup regarding organizations.

Implementation

Because you can create multiple security groups, you can create a security group for each organization, enabling centralized control of rules for individual organizations. For a Virtual Private Cloud (VPC), you can set up the virtual firewall to apply or not apply to the EC2 instances that are running, making it possible to turn them on or off as necessary.

- Create a security group for each organization, such as the development company, the operating company, and so forth.
- For each security group, perform the setup (of the access origin, the access ports, and the like) depending on the group (organization).
- Apply the security groups to the EC2 instances.

Configuration



Benefits

- This lets you have centralized control of access information for each accessing organization.
- This can reduce setup errors when modifying access control.
- You can use this together with the Functional Firewall Pattern.

Cautions

- In virtual firewalls, control mainly uses the connection origin IP address, rather than the organizational functions of ordinary users. Because of this, control by individual users requires implementation on the operating system or application layer.

Multi Load Balancer Pattern

Problem to Be Solved

When a web application is multi-device compatible, there will be access from PCs, mobile phones, and smart phones. At this time, if it is necessary to perform set up such as for SSL or to assign sessions for individual access devices, then if the setup is performed by the EC2 instances themselves, any change to the settings would become extremely laborious as the number of servers increases.

Explanation of the Cloud Solution/Pattern

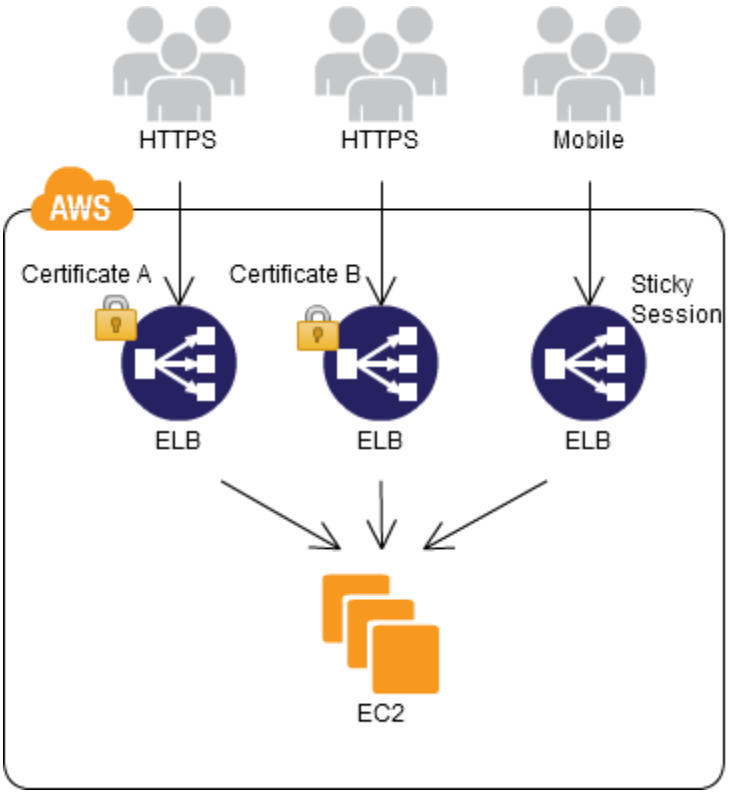
In groups of virtual servers, you can solve this problem by assigning multiple virtual load balancers with different settings. That is, rather than modifying the servers, you can change the behavior relative to access by the different devices by changing the virtual load balancer for routing the access. For example, you can apply this to settings such as for sessions, health checks, HTTPS, and so forth.

Implementation

Assign multiple virtual load balancers, known as "Elastic Load Balancers" (ELBs), to a single EC2 instance. You can use the SSL Termination function of the ELB to perform the HTTPS (SSL) process.

- Place an EC2 instance under the control of the ELBs.
- Prepare ELBs with different settings for sessions, health checks, HTTPS, and the like, and switch between them for the same EC2 instance.

Configuration



Benefits

- The behavior (on the load balancer level) for mobile sites and PC sites can be different, even when using the same EC2 instance.
- Even when multiple SSLs (HTTPS) are used by the same EC2 instance, you can prepare ELBs for each SSL (HTTP).

Cautions

- Note that when you cut off an EC2 instance from an ELB to perform maintenance, for example, you have to cut off the EC2 instance from all of the ELBs.
- When you use the SSL Termination function of an

WAF Proxy Pattern

Problem to Be Solved

Websites that handle sensitive personal information (such as credit card information), such as as e-commerce sites, for example, usually use a web application firewall (WAF) to increase security. However, in AWS Cloud, most of the systems start small, and in most cases no consideration has been given to implementing WAF. There are also many systems that assume that servers will be added or removed through scale-out/in, making the implementation of WAF difficult because it is not possible to define how many licenses will be required.

Explanation of the Cloud Solution/Pattern

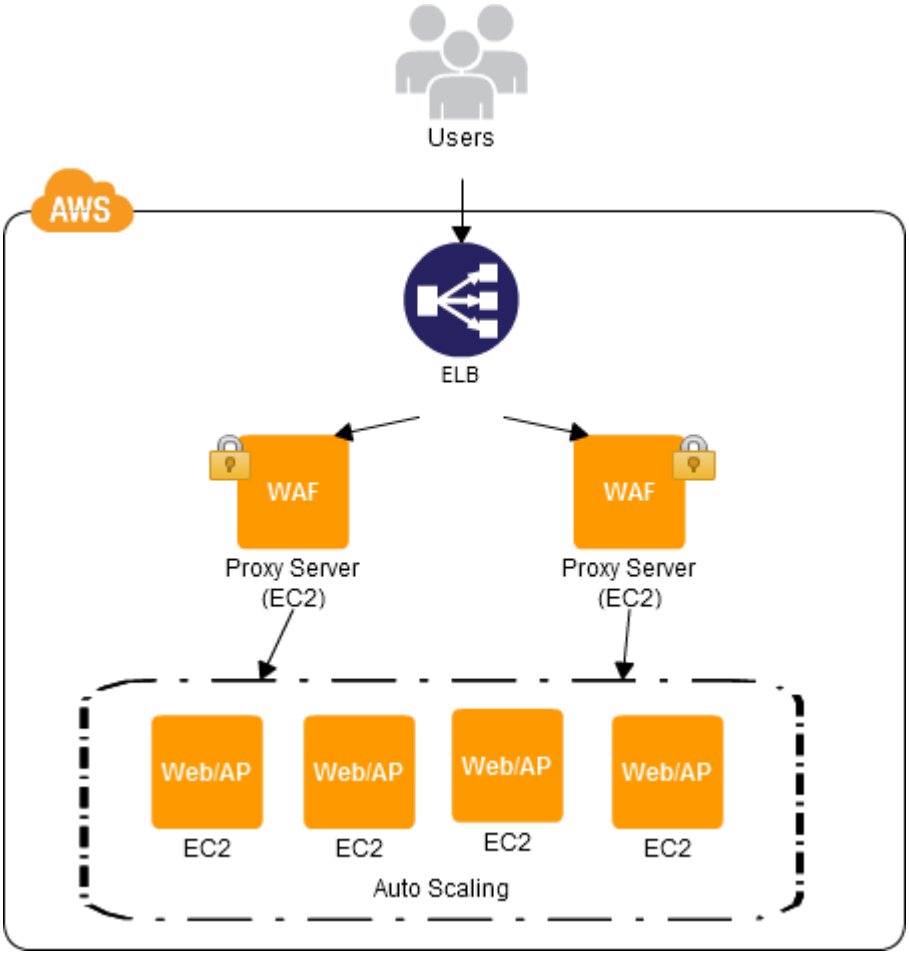
Because in the past the WAF was purchased after determining the number of servers, the number of WAF units installed was also unchanging, and thus this has not been a particular problem. However, in the AWS Cloud environment, which allows the number of servers to be increased or decreased at any time, installation of WAF in these servers is not practical; on the other hand, you can install a proxy server upstream and install WAF there. You can structure a proxy server so that only the WAF functions, making it possible to run with a small number of units, thus making it possible to operate with a minimal number of licenses.

Implementation

Locate a proxy server between the EC2 instances and the ELB, and install WAF there. You may install multiple units for redundancy.

- Prepare a proxy server (an EC2 instance) with WAF installed, between the ELB and the EC2 instances.
- If necessary, implement in the proxy server middleware for distributing the load, such as HAProxy.

Configuration



Benefits

- This lets you implement WAF without touching the web/AP servers.
- The number of WAF licenses required is not the number of web/AP servers, but rather is a smaller number, the number of proxy servers.

Cautions

- Prepare multiple proxy servers as well so as to not make a single point of failure (SPOF).
- Because the web/AP servers are positioned indirectly relative to the ELB, when increasing or decreasing the number of servers, Auto Scaling will be unable to use the function for attaching the EC2 instances to the ELB automatically.

CloudHub Pattern

Problem to Be Solved

When structuring Virtual Private Network (VPN) connections between multiple sites in a full mesh configuration, the greater the number of sites, the more complex the setup of the individual sites in the VPN router. This increases the maintenance costs. When solving this problem through structuring a star-configuration VPN, the VPN router for each site is connected only to a VPN hub. However, a failure in a VPN hub will affect all of the VPN connections, and thus the availability of the VPN hub is an important issue.

Explanation of the Cloud Solution/Pattern

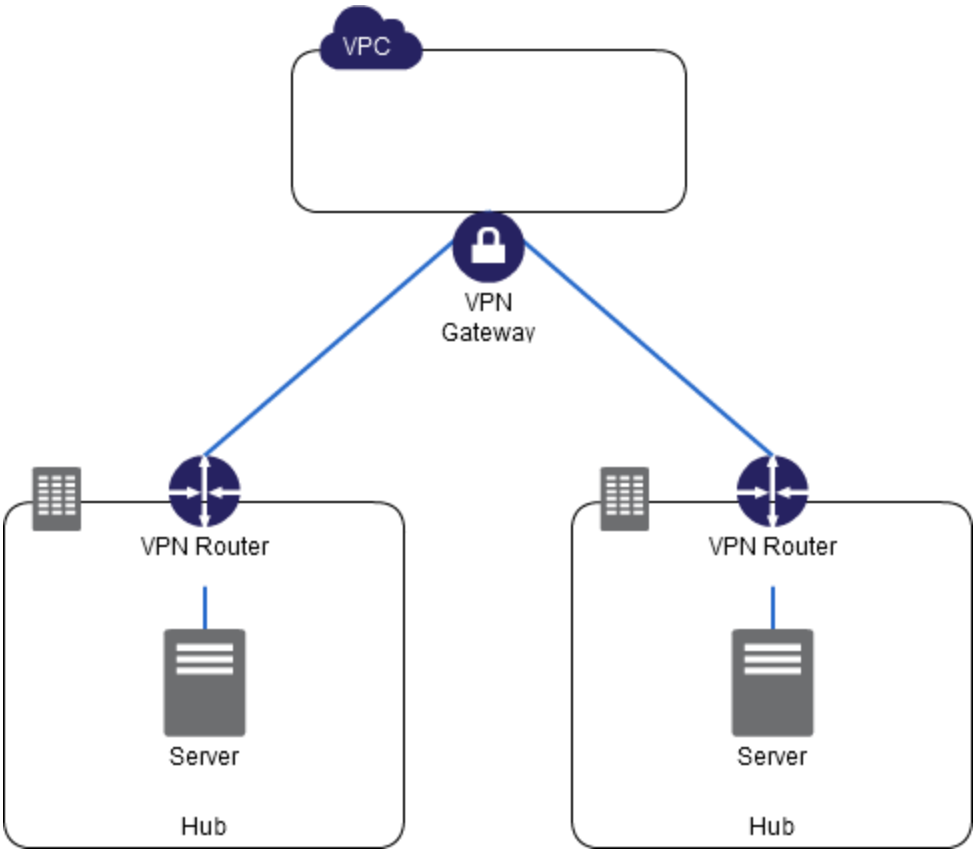
VPN hubs of the past have required high initial costs such as for redundant structures for the communication devices used in the VPN to increase availability. There has also been the fixed cost for maintaining the facilities, regardless of the amount of use of the VPN connections. Because of this, cost effectiveness has been poor. A VPN function is provided in the AWS Cloud, which you can use as a VPN hub. Because you can use the Cloud infrastructure, with its high availability, on a pay-as-you-go basis, you can structure the VPN connections between multiple sites easily, with both excellent availability and excellent cost effectiveness.

Implementation

A VPN connection function is provided in the Virtual Private Cloud (VPC) service. Produce the VPN connections between the multiple sites by connecting from the multiple sites with the VPC as the VPN hubs.

- Structure a VPC and set up Virtual Private Gateways as the VPN hubs.
- Set up a customer gateway for the individual sites, and set up the VPN connections so as to connect to the Virtual Private Gateways.
- Set up VPN routers for the individual sites and connect to the VPN hubs.

Configuration



Benefits

- Making VPN connections to the VPC for each individual site enables communication without having to set up the other sites.
- You can improve the reliability of the VPN as a whole by connecting the VPN hubs to the high availability/high operating efficiency infrastructure that is the AWS Cloud.

Cautions

- Regardless of the site outside of the VPC, the communication will always go through the VPC, which will incur a charge.
- Because the networks (sites) that are connected to the VPN gateways can communicate with each other, if there is need for access control, you must do so through the VPN routers at the individual sites.