Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region

April 29, 2011

Now that we have fully restored functionality to all affected services, we would like to share more details with our customers about the events that occurred with the Amazon Elastic Compute Cloud ("EC2") last week, our efforts to restore the services, and what we are doing to prevent this sort of issue from happening again. We are very aware that many of our customers were significantly impacted by this event, and as with any significant service issue, our intention is to share the details of what happened and how we will improve the service for our customers.

The issues affecting EC2 customers last week primarily involved a subset of the Amazon Elastic Block Store ("EBS") volumes in a single Availability Zone within the US East Region that became unable to service read and write operations. In this document, we will refer to these as "stuck" volumes. This caused instances trying to use these affected volumes to also get "stuck" when they attempted to read or write to them. In order to restore these volumes and stabilize the EBS cluster in that Availability Zone, we disabled all control APIs (e.g. Create Volume, Attach Volume, Detach Volume, and Create Snapshot) for EBS in the affected Availability Zone for much of the duration of the event. For two periods during the first day of the issue, the degraded EBS cluster affected the EBS APIs and caused high error rates and latencies for EBS calls to these APIs across the entire US East Region. As with any complicated operational issue, this one was caused by several root causes interacting with one another and therefore gives us many opportunities to protect the service against any similar event reoccurring.

Overview of EBS System

It is helpful to understand the EBS architecture so that we can better explain the event. EBS is a distributed, replicated block data store that is optimized for consistency and low latency read and write access from EC2 instances. There are two main components of the EBS service: (i) a set of EBS clusters (each of which runs entirely inside of an Availability Zone) that store user data and serve requests to EC2 instances; and (ii) a set of control plane services that are used to coordinate user requests and propagate them to the EBS clusters running in each of the Availability Zones in the Region.

An EBS cluster is comprised of a set of EBS nodes. These nodes store replicas of EBS volume data and serve read and write requests to EC2 instances. EBS volume data is replicated to multiple EBS nodes for durability and availability. Each EBS node employs a peer-to-peer based, fast failover strategy that aggressively provisions new replicas if one of the copies ever gets out of sync or becomes unavailable. The nodes in an EBS cluster are connected to each other via two networks. The primary network is a high bandwidth network used in normal operation for all necessary communication with other EBS nodes, with EC2 instances, and with the EBS control plane services. The secondary network, the replication network, is a lower capacity network used as a back-up network to allow EBS nodes to reliably communicate with other nodes in the EBS cluster and provide overflow capacity for data replication. This network is not designed to handle all traffic from the primary network but rather provide highly-reliable connectivity between EBS nodes inside of an EBS cluster.

When a node loses connectivity to a node to which it is replicating data to, it assumes the other node failed. To preserve durability, it must find a new node to which it can replicate its data (this is called re-mirroring). As part of the re-mirroring process, the EBS node searches its EBS cluster for another node with enough available server space, establishes connectivity with the server, and propagates the volume data. In a normally functioning cluster, finding a location for the new replica occurs in milliseconds. While data is being re-mirrored, all nodes that have copies of the data hold onto the data until they can confirm that another node has taken ownership of their portion. This provides an additional level of protection against customer data loss. Also, when data on a customer's volume is being re-mirrored, access to that data is blocked until the system has identified a new primary (or writable) replica. This is required for consistency of EBS volume data under all potential failure modes. From the perspective of an EC2 instance trying to do I/O on a volume while this is happening, the volume will appear "stuck".

In addition to the EBS clusters, there is a set of control plane services that accepts user requests and propagates them to the appropriate EBS cluster. There is one set of EBS control plane services per EC2 Region, but the control plane itself is highly distributed across the Availability Zones to provide availability and fault tolerance. These control plane services also act as the authority to the EBS clusters when they elect primary replicas for each volume in the cluster (for consistency, there must only be a single primary replica for each volume at any time). While there are a few different services that comprise the control plane, we will refer to them collectively as the "EBS control plane" in this document.

Primary Outage

At 12:47 AM PDT on April 21st, a network change was performed as part of our normal AWS scaling activities in a single Availability Zone in the US East Region. The configuration change was to upgrade the capacity of the primary network. During the change, one of the standard steps is to shift traffic off of one of the redundant routers in the primary EBS network to allow the upgrade to happen. The traffic shift was executed incorrectly and rather than routing the traffic to the other router on the primary network, the traffic was routed onto the lower capacity redundant EBS network. For a portion of the EBS cluster in the affected Availability Zone, this meant that they did not have a functioning primary or secondary network because traffic was purposely shifted away from the primary network and the secondary network couldn't handle the traffic level it was receiving. As a result, many EBS nodes in the affected Availability Zone were completely isolated from other EBS nodes in its cluster. Unlike a normal network interruption, this change disconnected both the primary and secondary network simultaneously, leaving the affected nodes completely isolated from one another.

When this network connectivity issue occurred, a large number of EBS nodes in a single EBS cluster lost connection to their replicas. When the incorrect traffic shift was rolled back and network connectivity was restored, these nodes rapidly began searching the EBS cluster for available server space where they could re-mirror data. Once again, in a normally functioning cluster, this occurs in milliseconds. In this case, because the issue affected such a large number of volumes concurrently, the free capacity of the EBS cluster was quickly exhausted, leaving many of the nodes "stuck" in a loop, continuously searching the cluster for free space. This quickly led to a "re-mirroring storm," where a large number of volumes were effectively "stuck" while the nodes searched the cluster for the storage space it needed for its new replica. At this point, about 13% of the volumes in the affected Availability Zone were in this "stuck" state.

After the initial sequence of events described above, the degraded EBS cluster had an immediate impact on the EBS control plane. When the EBS cluster in the affected Availability Zone entered the re-mirroring storm and exhausted its available capacity, the cluster became unable to service "create volume" API requests. Because the EBS control plane (and the create volume API in particular) was configured with a long time-out period, these slow API calls began to back up and resulted in thread starvation in the EBS control plane. The EBS control plane has a regional pool of available threads it can use to service requests. When these threads were completely filled up by the large number of queued requests, the EBS control plane had no ability to service API requests and began to fail API requests for other Availability Zones in that Region as well. At 2:40 AM PDT on April 21st, the team deployed a change that disabled all new Create Volume requests in the affected Availability Zone, and by 2:50 AM PDT, latencies and error rates for all other EBS related APIs recovered.

Two factors caused the situation in this EBS cluster to degrade further during the early part of the event. First, the nodes failing to find new nodes did not back off aggressively enough when they could not find space, but instead, continued to search repeatedly. There was also a race condition in the code on the EBS nodes that, with a very low probability, caused them to fail when they were concurrently closing a large number of requests for replication. In a normally operating EBS cluster, this issue would result in very few, if any, node crashes; however, during this re-mirroring storm, the volume of connection attempts was extremely high, so it began triggering this issue more frequently. Nodes began to fail as a result of the bug, resulting in more volumes left needing to re-mirror. This created more "stuck" volumes and added more requests to the re-mirroring storm.

By 5:30 AM PDT, error rates and latencies again increased for EBS API calls across the Region. When data for a volume needs to be re-mirrored, a negotiation must take place between the EC2 instance, the EBS nodes with the volume data, and the EBS control plane (which acts as an authority in this process) so that only one copy of the data is designated as the primary replica and recognized by the EC2 instance as the place where all accesses should be sent. This provides strong consistency of EBS volumes. As more EBS nodes continued to fail because of the race condition described above, the volume of such negotiations with the EBS control plane increased. Because data was not being successfully re-mirrored, the number of these calls increased as the system retried and new requests came in. The load caused a brown out of the EBS control plane and again affected EBS APIs across the Region. At 8:20 AM PDT, the team began disabling all communication between the degraded EBS cluster in the affected Availability Zone and the EBS control plane. While this prevented all EBS API access in the affected Availability Zone (we will discuss recovery of this in the next section), other latencies and error rates returned to normal for EBS APIs for the rest of the Region.

A large majority of the volumes in the degraded EBS cluster were still functioning properly and the focus was to recover the cluster without affecting more volumes. At 11:30AM PDT, the team developed a way to prevent EBS servers in the degraded EBS cluster from futilely contacting other servers (who didn't have free space at this point anyway) without affecting the other essential communication between nodes in the cluster. After this change was made, the cluster stopped degrading further and additional volumes were no longer at risk of becoming "stuck". Before this change was deployed, the failed servers resulting from the race condition resulted in an additional 5% of the volumes in the affected Availability Zone becoming "stuck". However, volumes were also slowly re-mirroring as some capacity was made available which allowed existing "stuck" volumes to become "unstuck". The net result was that when this change was deployed, the total "stuck" volumes in the affected Availability Zone was 13%.

Customers also experienced elevated error rates until Noon PDT on April 21st when attempting to launch new EBS-backed EC2 instances in Availability Zones other than the affected zone. This occurred for approximately 11 hours, from the onset of the outage until Noon PM PDT on April 21st. Except for the periods of broader API issues describe above, customers were able to create EBS-backed EC2 instances but were experiencing significantly-elevated error rates and latencies. New EBS-backed EC2 launches were being affected by a specific API in the EBS control plane that is only needed for attaching new instances to volumes. Initially, our alarming was not fine-grained enough for this EBS control plane API and the launch errors were overshadowed by the general error from the degraded EBS cluster. At 11:30 AM PDT, a change to the EBS control plane fixed this issue and latencies and error rates for new EBS-backed EC2 instances declined rapidly and returned to near-normal at Noon PDT.

Recovering EBS in the Affected Availability Zone

By 12:04 PM PDT on April 21st, the outage was contained to the one affected Availability Zone and the degraded EBS cluster was stabilized. APIs were working well for all other Availability Zones and additional volumes were no longer becoming "stuck". Our focus shifted to completing the recovery. Approximately 13% of the volumes in the Availability Zone remained "stuck" and the EBS APIs were disabled in that one affected Availability Zone. The key priority became bringing additional storage capacity online to allow the "stuck" volumes to find enough space to create new replicas.

The team faced two challenges which delayed getting capacity online. First, when a node fails, the EBS cluster does not reuse the failed node until every data replica is successfully re-mirrored. This is a conscious decision so that we can recover data if a cluster fails to behave as designed. Because we did not want to re-purpose this failed capacity until we were sure we could recover affected user volumes on the failed nodes, the team had to install a large amount of additional new capacity to replace that capacity in the cluster. This required the time-consuming process of physically relocating excess server capacity from across the US East Region and installing that capacity into the degraded EBS cluster. Second, because of the changes made to reduce the node-to-node communication used by peers to find new capacity (which is what stabilized the cluster in the step described above), the team had difficulty incorporating the new capacity into the cluster. The team had to carefully make changes to their negotiation throttles to allow negotiation to occur with the newly-built servers without again inundating the old servers with requests that they could not service. This process took longer than we expected as the team had to navigate a number of issues as they worked around the disabled communication. At about 02:00AM PDT on April 22nd, the team successfully started adding significant amounts of new capacity and working through the replication backlog. Volumes were restored consistently over the next nine hours and all but about 2.2% of the volumes in the affected Availability Zone were restored by 12:30PM PDT on April 22nd. While the restored volumes were fully replicated, not all of them immediately became "unstuck" from the perspective of the attached EC2 instances because some were blocked waiting for the EBS control plane to be contactable, so they could safely re-establish a connection with the EC2 instance and elect a new writable copy.

Once there was sufficient capacity added to the cluster, the team worked on re-establishing EBS control plane API access to the affected Availability Zone and restoring access to the remaining "stuck" volumes. There was a large backlog of state changes that had to be propagated both from the degraded EBS nodes to the EBS control plane and vice versa. This effort was done gradually to avoid impact to the restored volumes and the EBS control plane. Our initial attempts to bring API access online to the impacted Availability Zone centered on throttling the state propagation to avoid overwhelming the EBS control plane. We also began building out a separate instance of the EBS control plane, one we could keep partitioned to the affected Availability Zone to avoid impacting other Availability Zones in the Region, while we processed the backlog. We rapidly developed throttles that turned out to be too coarse-grained to permit the right requests to pass through and stabilize the system. Through the evening of April 22nd into the morning of April 23rd, we worked on developing finer-grain throttles. By Saturday morning, we had finished work on the dedicated EBS control plane and the finer-grain throttles. Initial tests of traffic against the EBS control plane demonstrated progress and shortly after 11:30 AM PDT on April 23rd we began steadily processing the backlog. By 3:35PM PDT, we finished enabling access to the EBS control plane to the degraded Availability Zone. This allowed most of the remaining volumes, which were waiting on the EBS control plane to help negotiate which replica would be writable, to once again be usable from their attached instances. At 6:15 PM PDT on April 23rd, API access to EBS resources was restored in the affected Availability Zone.

With the opening up of API access in the affected Availability Zone, APIs were now operating across all Availability Zones in the Region. The recovery of the remaining 2.2% of affected volumes required a more manual process to restore. The team had snapshotted these volumes to S3 backups early in the event as an extra precaution against data loss while the event was unfolding. At this point, the team finished developing and testing code to restore volumes from these snapshots and began processing batches through the night. At 12:30 PM PDT on April 24, we had finished the volumes that we could recover in this way and had recovered all but 1.04% of the affected volumes. At this point, the team began forensics on the remaining volumes which had suffered machine failure and for which we had not been able to take a snapshot. At 3:00 PM PDT, the team began restoring these. Ultimately, 0.07% of the volumes in the affected Availability Zone could not be restored for customers in a consistent state.

Impact on Amazon Relational Database Service (RDS)

In addition to the direct effect this EBS issue had on EC2 instances, it also impacted the Relational Database Service ("RDS"). RDS depends upon EBS for database and log storage, and as a result a portion of the RDS databases hosted in the primary affected Availability Zone became inaccessible.

Customers can choose to operate RDS instances either in a single Availability Zone ("single-AZ") or replicated across multiple Availability Zones ("multi-AZ"). Single-AZ database instances are exposed to disruptions in an Availability Zone. In this case, a single-AZ database instance would have been affected if one of the EBS volumes it was relying on got "stuck". In the primary affected Availability Zone, a peak of 45% of single-AZ instances were impacted with "stuck" I/O. This was a relatively-bigger portion of the RDS population than the corresponding EBS volume population because RDS database instances make use of multiple EBS volumes. This increases aggregate I/O capacity for database workloads under normal conditions, but means that a "stuck" I/O on any volume for a single-AZ database instance can make it inoperable until the volume is restored. The percentage of "stuck" single-AZ database instances in the affected Availability Zone decreased steadily during the event as the EBS recovery proceeded. The percentage of "stuck" single-AZ database instances in the affected Availability Zone decreased to 41.0% at the end of 24 hours, 23.5% at 36 hours and 14.6% at the end of 48 hours, and the rest recovered throughout the weekend. Though we recovered nearly all of the affected database instances, 0.4% of single-AZ database instances in the affected Availability Zone had an underlying EBS storage volume that was not recoverable. For these database instances, customers with automatic backups turned on (the default setting) had the option to initiate point-in-time database restore operations.

RDS multi-AZ deployments provide redundancy by synchronously replicating data between two database replicas in different Availability Zones. In the event of a failure on the primary replica, RDS is designed to automatically detect the disruption and fail over to the secondary replica. Of multi-AZ database instances in the US East Region, 2.5% did not automatically failover after experiencing "stuck" I/O. The primary cause was that the rapid succession of network interruption (which partitioned the primary from the secondary) and "stuck" I/O on the primary replica triggered a previously un-encountered bug. This bug left the primary replica in an isolated state where it was not safe for our monitoring agent to automatically fail over to the secondary replica without risking data loss, and manual intervention was required. We are actively working on a fix to resolve this issue.

Preventing the Event

The trigger for this event was a network configuration change. We will audit our change process and increase the automation to prevent this mistake from happening in the future. However, we focus on building software and services to survive failures. Much of the work that will come out of this event will be to further protect the EBS service in the face of a similar failure in the future.

We will be making a number of changes to prevent a cluster from getting into a re-mirroring storm in the future. With additional excess capacity, the degraded EBS cluster would have more quickly absorbed the large number of re-mirroring requests and avoided the re-mirroring storm. We now understand the amount of capacity needed for large recovery events and will be modifying our capacity planning and alarming so that we carry the additional safety capacity that is needed for large scale failures. We have already increased our capacity buffer significantly, and expect to have the requisite new capacity in place in a few weeks. We will also modify our retry logic in the EBS server nodes to prevent a cluster from getting into a re-mirroring storm. When a large interruption occurs, our retry logic will back off more aggressively and focus on re-establishing connectivity with previous replicas rather than futilely searching for new nodes with which to re-mirror. We have begun working through these changes and are confident we can address the root cause of the re-mirroring storm by modifying this logic. Finally, we have identified the source of the race condition that led to EBS node failure. We have a fix and will be testing it and deploying it to our clusters in the next couple of weeks. These changes provide us with three separate protections against having a repeat of this event.

Impact to Multiple Availability Zones

EC2 provides two very important availability building blocks: Regions and Availability Zones. By design, Regions are completely separate deployments of our infrastructure. Regions are completely isolated from each other and provide the highest degree of independence. Many users utilize multiple EC2 Regions to achieve extremely-high levels of fault tolerance. However, if you want to move data between Regions, you need to do it via your applications as we don't replicate any data between Regions on our users' behalf. You also need to use a separate set of APIs to manage each Region. Regions provide users with a powerful availability building block, but it requires effort on the part of application builders to take advantage of this isolation. Within Regions, we provide Availability Zones to help users build fault-tolerant applications easily. Availability Zones are physically and logically separate infrastructure that are built to be highly independent while still providing users with high speed, low latency network connectivity, easy ways to replicate data, and a consistent set of management APIs. For example, when running inside a Region, users have the ability to take EBS snapshots which can be restored in any Availability Zone and can programmatically manipulate EC2 and EBS resources with the same APIs. We provide this loose coupling because it allows users to easily build highly-fault-tolerant applications.

This event had two distinct impacts. First, there was an impact to running applications in the affected Availability Zone because affected EBS volumes became "stuck". Because of the architecture of the EBS service, the impact to running instances was limited to the affected Availability Zone. As a result, many users who wrote their applications to take advantage of multiple Availability Zones did not have significant availability impact as a result of this event. Some customers reported that they had "stuck" EBS volumes in Availability Zones other than the impacted Availability Zone on Thursday. While our monitoring clearly shows the effect of the re-mirroring storm on the EBS control plane and on volumes within the affected Availability Zone, it does not reflect significant impact to existing EBS volumes within other Availability Zones in the Region. We do see that there were slightly more "stuck" volumes than we would have expected in the healthy Availability Zones, though still an extremely small number. To put this in perspective, the peak "stuck" volume percentage we saw in the Region outside of the affected Availability Zone was less than 0.07%. We investigated a number of these "stuck" volumes. The slightly-elevated number of "stuck" volumes in these non-impacted zones was caused by the delays in recovering from normal re-mirrors because of the increased latencies and error rates of the EBS control plane described above; there is always a background rate of volume re-mirroring going on. We also believe that the work described below to further insulate the EBS control plane will prevent even this slightly-elevated rate if something similar happened.

While users' applications taking advantage of multiple Availability Zone ("multi-AZ") architectures were able to avoid impact from this event, there was definitely an impact on the EBS control plane that affected the ability to create and manipulate EBS volumes across the Region. One of the advantages of EC2 is the ability to rapidly replace failed resources. When the EBS control plane was degraded or unavailable, it made it difficult for customers with affected volumes to replace their volumes or EBS-booted EC2 instances in other healthy Availability Zones. Preventing this from reoccurring is a top priority.

Even though we provide a degree of loose coupling for our customers, our design goal is to make Availability Zones indistinguishable from completely independent. Our EBS control plane is designed to allow users to access resources in multiple Availability Zones while still being tolerant to failures in individual zones. This event has taught us that we must make further investments to realize this design goal. There are three things we will do to prevent a single Availability Zone from impacting the EBS control plane across multiple Availability Zones. The first is that we will immediately improve our timeout logic to prevent thread exhaustion when a single Availability Zone cluster is taking too long to process requests. This would have prevented the API impact from 12:50 AM PDT to 2:40 AM PDT on April 21st. To address the cause of the second API impact, we will also add the ability for our EBS control plane to be more Availability Zone aware and shed load intelligently when it is over capacity. This is similar to other throttles that we already have in our systems. Additionally, we also see an opportunity to push more of our EBS control plane into per-EBS cluster services. By moving more functionality out of the EBS control plane and creating per-EBS cluster deployments of these services (which run in the same Availability Zone as the EBS cluster they are supporting), we can provide even better Availability Zone isolation for the EBS control plane.

Making it Easier to Take Advantage of Multiple Availability Zones

We also intend to make it easier for customers to take advantage of multiple Availability Zones. First, we will offer multiple Availability Zones for all of our services, including Amazon Virtual Private Cloud ("VPC"). Today, VPC customers only have access to a single Availably Zone. We will be adjusting our roadmap to give VPC customers access to multiple Availability Zones as soon as possible. This will allow VPC customers to build highly-available applications using multiple Availability Zones just as EC2 customers not using a VPC do today.

A related finding from this event is we need to do a better job of making highly-reliable multi-AZ deployments easy to design and operate. Some customers' applications (or critical components of the application like the database) are deployed in only a single Availability Zone, while others have instances spread across Availability Zones but still have critical, single points of failure in a single Availability Zone. In cases like these, operational issues can negatively impact application availability when a robust multi-Availability Zone deployment would allow the application to continue without impact. We will look to provide customers with better tools to create multi-AZ applications that can support the loss of an entire Availability Zone without impacting application availability. We know we need to help customers design their application logic using common design patterns. In this event, some customers were seriously impacted, and yet others had resources that were impacted but saw nearly no impact on their applications.

In order to work more closely with our customers and partners on best practices for architecting in the cloud, we will be hosting a series of free webinars starting Monday, May 2. The first topics we will cover will be Designing Fault-tolerant Applications, Architecting for the Cloud, and Web Hosting Best Practices. We anticipate adding more topics to the series over the next few weeks, and will continue to do these on a frequent ongoing basis. The webinars over the next two weeks will be hosted several times daily to support our customers around the world in multiple time zones. We will set aside a significant portion of the webinars for detailed Q&A. Follow-up discussions for customers or partners will also be arranged. These webinars, as well as a series of whitepapers on best practices for architecting for the AWS cloud, are available in a new Architecture Center on the AWS website. We'll also continue to deliver additional services like S3, SimpleDB and multi-AZ RDS that perform multi-AZ level balancing automatically so customers can benefit from multiple Availability Zones without doing any of the heavy-lifting in their applications.

Speeding Up Recovery

We will also invest in increasing our visibility, control, and automation to recover volumes in an EBS cluster. We have a number of operational tools for managing an EBS cluster, but the fine-grained control and throttling the team used to recover the cluster will be built directly into the EBS nodes. We will also automate the recovery models that we used for the various types of volume recovery that we had to do. This would have saved us significant time in the recovery process. We will also look at what changes we can make to preserve volume functionality during periods of degraded cluster operation, including adding the ability to take a snapshot of a "stuck" volume. If customers had this ability, they would have been able to more easily recover their applications in other Availability Zones in the Region.

Improving Communication and Service Health Tools During Operational Issues

In addition to the technical insights and improvements that will result from this event, we also identified improvements that need to be made in our customer communications. We would like our communications to be more frequent and contain more information. We understand that during an outage, customers want to know as many details as possible about what's going on, how long it will take to fix, and what we are doing so that it doesn't happen again. Most of the AWS team, including the entire senior leadership team, was directly involved in helping to coordinate, troubleshoot and resolve the event. Initially, our primary focus was on thinking through how to solve the operational problems for customers rather than on identifying root causes. We felt that that focusing our efforts on a solution and not the problem was the right thing to do for our customers, and that it helped us to return the services and our customers back to health more quickly. We updated customers when we had new information that we felt confident was accurate and refrained from speculating, knowing that once we had returned the services back to health that we would quickly transition to the data collection and analysis stage that would drive this post mortem.

That said, we think we can improve in this area. We switched to more regular updates part of the way through this event and plan to continue with similar frequency of updates in the future. In addition, we are already working on how we can staff our developer support team more expansively in an event such as this, and organize to provide early and meaningful information, while still avoiding speculation.

We also can do a better job of making it easier for customers to tell if their resources have been impacted, and we are developing tools to allow you to see via the APIs if your instances are impaired.

Service Credit for Affected Customers

For customers with an attached EBS volume or a running RDS database instance in the affected Availability Zone in the US East Region at the time of the disruption, regardless of whether their resources and application were impacted or not, we are going to provide a 10 day credit equal to 100% of their usage of EBS Volumes, EC2 Instances and RDS database instances that were running in the affected Availability Zone. These customers will not have to do anything in order to receive this credit, as it will be automatically applied to their next AWS bill. Customers can see whether they qualify for the service credit by logging into their AWS Account Activity page.

In Conclusion

Last, but certainly not least, we want to apologize. We know how critical our services are to our customers' businesses and we will do everything we can to learn from this event and use it to drive improvement across our services. As with any significant operational issue, we will spend many hours over the coming days and weeks improving our understanding of the details of the various parts of this event and determining how to make changes to improve our services and processes.

Sincerely,
The AWS Team

Sign In to the Console

Learn About AWS

What Is AWS?

What Is Cloud Computing?

What Is DevOps?

What Is a Container?

What Is a Data Lake?

AWS Cloud Security

What's New

Blogs

Press Releases

Resources for AWS

Getting Started

Training and Certification

AWS Solutions Portfolio

Architecture Center

Product and Technical FAQs

Analyst Reports

AWS Partner Network

Developers on AWS

Developer Center

SDKs & Tools

.NET on AWS

Python on AWS

Java on AWS

PHP on AWS