Migrating AWS Resources to a New AWS Region

July 2017

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

This document is intended for experienced customers of Amazon Web Services who want to migrate existing resources to a new AWS Region. You might want to migrate for a variety of reasons. In particular, if a new region becomes available that is closer to your user base, you might want to locate various services geographically closer to those users.

This document is not intended to be a “step-by-step” or “definitive” guide. Rather, it provides a variety of options and methods for migrating various services that you might require in a new region.

Introduction

Amazon Web Services (AWS) now operates in a number of geographic AWS Regions worldwide (United States, Europe, Asia Pacific, South America, etc.), serving customers in over 190 countries. For many AWS services, you can choose the region from which you want to deliver those services. Each region has multiple Availability Zones. By using separate Availability Zones, you can additionally protect your applications from the failure of a single location. By using separate AWS Regions, you can design your application to be closer to your customers and achieve lower latency and higher throughput. AWS has designed the regions to be isolated from each other so that you can achieve greater fault tolerance and improved stability in your applications.

Scope of AWS Resources

While most AWS services operate within a region, the following services operate across all regions and require no migration:

- AWS Identity and Access Management (AWS IAM)
- AWS Management Console
- Amazon CloudWatch

Further, as all services are accessible using API endpoints, you do not necessarily need to migrate all components of your architecture to the new region depending on your application. For example, you can migrate Amazon Elastic Compute Cloud (Amazon EC2) instances but retain existing Amazon Simple Storage Service (Amazon S3) and Amazon CloudFront configurations.

When planning a migration to a new region, we recommend that you check what AWS products and services are available in that region. An updated list of AWS product and service offerings by region is available here.[1]

AWS IAM and Security Considerations

AWS IAM enables you to securely control access to AWS services and resources for your users.

IAM users are created and managed within the scope of an AWS account, rather than a particular region. No migration of users or groups is required.

When migrating to a new region, it is important to note any defined policy restrictions on IAM users. For example, Amazon Resource Names (ARNs) might restrict you to a specific region. For more information, see IAM Identifiers in the *AWS Identity and Access User Guide*. [2]

IAM is a core security service that enables you to add specific policies to control user access to AWS resources. Some policies can affect:

- Time-of-day access (which can require consideration due to time zone differences)
- Use of new originating IP addresses
- Whether you need to use SSL connections
- How users are authenticated
- Whether you can use multi-factor authentication (MFA) devices

Because IAM underpins security, we recommend that you carefully review your security configuration policies, procedures, and practices before a region migration.

Migrating Compute Resources

This section covers the migration of compute services such as Amazon EC2 and other closely associated services for security, storage, load balancing, and Auto Scaling.

Migrating Amazon EC2 Instances

Amazon EC2 is a web service that provides resizable compute capacity in the cloud. It is designed to make web-scale computing easier for developers. Migrating an instance involves copying the data and images, ensuring that the security groups and SSH keys are present, and then restarting fresh instances.

SSH Keys

AWS does not keep any of your user SSH private keys after they are generated. These public keys are made available to Amazon EC2 instances when they are running. (Under Linux operating systems, these are normally copied into the relevant user's `~/.ssh/authorized_keys` file.)

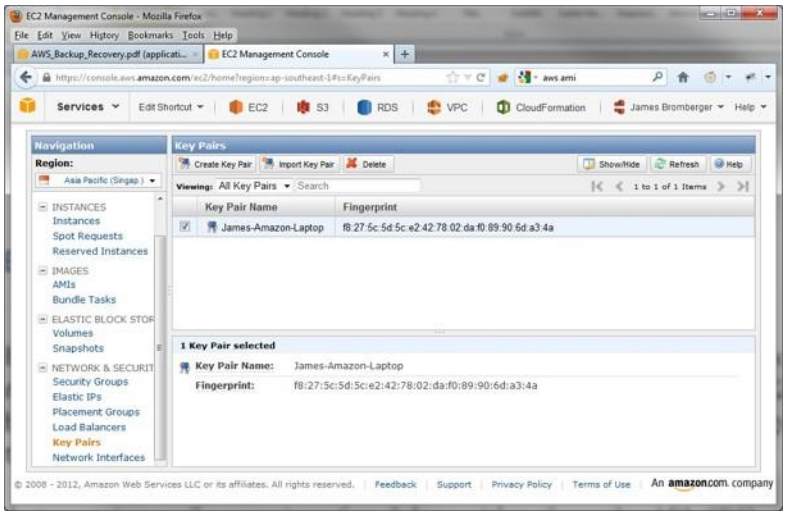


Figure 1. Key pairs in the AWS Management Console

You can retrieve a fingerprint of each key from the application programming interface (API), software development kit (SDK), command line interface (CLI), or the AWS Management Console.

SSH public keys are only stored per region. AWS does not copy or synchronize your configured SSH keys between regions. It is up to you to determine if you will use separate SSH keys per region or the same SSH keys in several regions.

> **Note:** You can log in to an existing Linux instance in the source region, obtain a copy of the public key (from `~/.ssh/authorized_keys`), and import this public key into the target region.

It is important to know that Auto Scaling launch configurations and AWS CloudFormation templates might refer to SSH keys using the key pair name. In these cases, you must take care to either update any Auto Scaling launch configuration or AWS CloudFormation template to use keys that are available in a new region, or deploy the public key with the same key pair name to the new region.

For more information, see AWS Security Credentials in the *AWS General Reference*.[3]

Security Groups

Security groups in Amazon EC2 restrict ingress traffic (or in the case of virtual private cloud or VPC, ingress and egress traffic) to a group of EC2 instances. Each rule in a security group can refer to the source (or in VPC, the destination) by either a CIDR notation IPv4 address range (*a.b.c.d/x*), or by using the security group identifier (*sg-XXXXXXXX*).

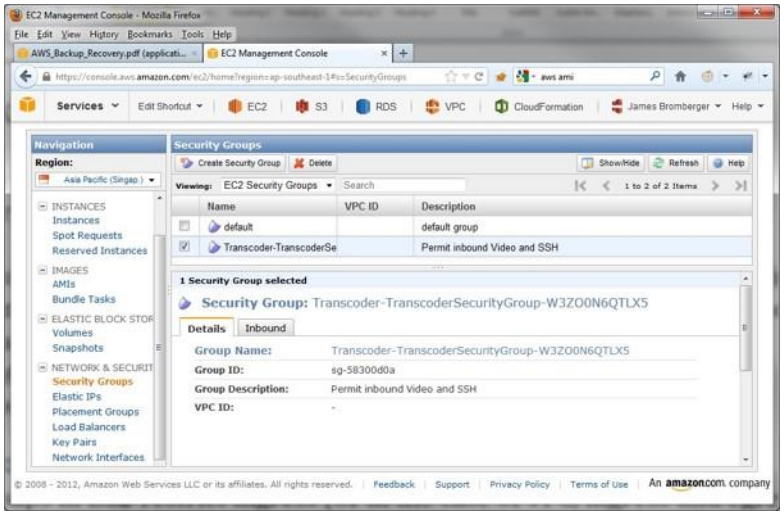


Figure 2. Security group configuration in the AWS Management Console

Each security group can exist within the scope of only one region. The same name can exist in multiple regions but have different definitions of what traffic is permitted to pass.

Every instance being launched must be a member of a security group. If a host is being started as part of an Auto Scaling launch configuration or an AWS CloudFormation template, the required security group must exist. (AWS CloudFormation templates might often define the security group to be created as part of the template.)

It is vital that you review your configured security groups to ensure that the required level of network access restrictions is in place.

To export a copy of the definitions of existing security groups (using the command line tools), run the following command:

```
ec2-describe-group -H --region <sourceregionname> >
security_groups.txt
```

For more information, see Security Groups in the *Amazon EC2 User Guide*.[4]

Amazon Machine Images

An Amazon Machine Image (AMI) is a special type of preconfigured operating system image used to create a virtual machine (an EC2 instance) within the Amazon EC2 environment. Each AMI is assigned an identifier, of the form "ami- XXXXXXXX", where "X" is a hexadecimal value (0-9, A-F).

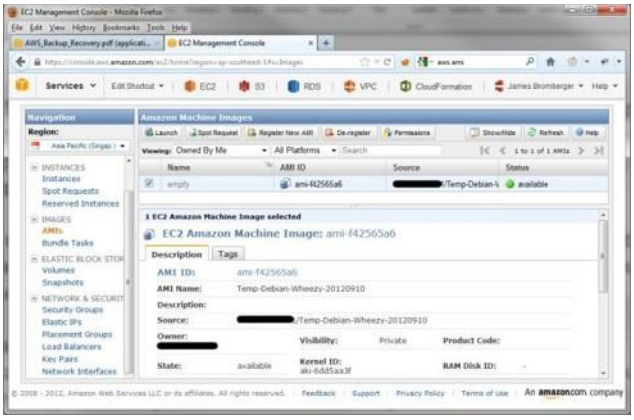


Figure 3. AMIs in the AWS Management Console

Each AMI is unique per region. AMIs do not span multiple regions. However, the same content of an AMI can be available in other regions (for example, Amazon Linux 2016.09 or Windows Server 2012 R2). Each region has its own unique AMI ID for its copy of this data.

You can create your own AMIs from running instances and use these as a starting point for launching additional instances. These user-created AMIs are assigned a unique AMI ID within the region.

AMI IDs are used within Auto Scaling launch configuration and AWS CloudFormation templates. If you plan to use Auto Scaling or AWS CloudFormation, you need to update the AMI ID references to match the ones that exist in the target region.

Migration of AMIs across regions is supported using the EC2 AMI Copy function.[5] AMI Copy enables you to copy an AMI to as many regions as you want from the AWS Management Console, the Amazon EC2 CLI, or the Amazon EC2 API. AMI Copy is available for AMIs backed by Amazon Elastic Block Store (EBS) and instance store-backed AMIs, and is operating system-agnostic.

Each copy of an AMI results in a new AMI with its own unique AMI ID. Any changes made to the source AMI during or after a copy are not propagated to the new AMI as part of the AMI copy process. You must recopy the AMI to the target regions to copy the changes made to the source AMI.

> **Note**: Permissions and user-defined tags applied to the source AMI are not copied to the new AMIs as part of the AMI copy process. After the copy is complete, you can apply any permissions and user-defined tags to the new AMIs.

Amazon EBS Volumes

Amazon EBS is a block storage volume that can be presented to an EC2 instance. You can format an EBS volume with a specific file system type such as NTFS, Ext4, XFS, etc. EBS volumes can contain the operating system boot volume or be used as an additional data drive (Windows) or mount point (Linux).

You can migrate EBS volumes using the cross-region EBS snapshot copy capability.[6] This enables you to copy snapshots of EBS volumes between regions using either the AWS Management Console, API call, or command line.

EBS Snapshot Copy offers the following key capabilities:

- The AWS Management Console shows you the progress of a snapshot copy in progress, where you can check the percentage completed.
- You can initiate multiple EBS Snapshot Copy commands simultaneously either by selecting and copying multiple snapshots to the same region or by copying a snapshot to multiple regions in parallel. The in-progress copies do not affect the performance of the associated EBS volumes.
- The console-based interface is push-based. You log in to the source region and tell the console where you'd like the snapshot to end up. The

API and the command line are, by contrast, pull-based. You must run them within the target region.

The entire process takes place without the need to use external tools or perform any configuration. Here is a high-level overview of the migration process:

1. Identify relevant EBS volumes to migrate (you can choose to use tagging to assist in identification).
2. Identify which volumes can be copied with the application running and which require you to pause or shut down the application. EBS Snapshot Copy accesses a snapshot of the primary volume rather than the volume itself. Therefore, you might need to shut down the application during the copy process to ensure the latest data is copied across.
3. Create the necessary EBS snapshots and wait for their status to be "Complete".
4. Initiate the EBS Snapshot Copy feature using either the AWS Management Console, API, or CLI.
5. Create EBS volumes at the target region by selecting the relevant snapshots and using the "create volume from snapshot" functionality.

Volumes and Snapshots

Amazon EBS volumes can currently be from 1 GB to 16 TB in size (in 1 GB increments). They can be used with disk management tools, such as Logical Volume Manager (LVM) or Windows Disk Manager, to span or stripe across multiple block devices. You can stripe multiple EBS volumes together to deliver higher performance storage volumes to applications.

Volumes that are in constant use might benefit from having a snapshot taken, especially if there are multiple volumes being used in RAID 1 stripe or part of an LVM volume group.

Provisioned IOPS volumes are another way to increase EBS performance. These volumes are designed to deliver predictable, high performance for I/O-intensive workloads such as databases. To enable EC2 instances to fully use the IOPS provisioned on an EBS volume, you can launch selected EC2 instance types as "EBS-Optimized" instances. Before a region migration, we recommend that you

check that these instances are supported in the Availability Zones in the target region.

For more information about for getting optimal performance from your EBS volumes, see Amazon EBS Performance Tips in the *Amazon EC2 User Guide*.[7]

Elastic IP Addresses

Elastic IP addresses are assigned to an account from the pool of addresses for a given region. As such, an Elastic IP address cannot be migrated between regions.

We recommend you update the time-to-live (TTL) value on your Domain Name System (DNS) server that points to this Elastic IP address, and reduce it to an amount that is a tolerable delay in DNS cache expire, such as 300 seconds (five minutes) or less. Any decrease in DNS TTL could result in an increase in DNS requests, increase load on your current DNS service, and affect charges from your DNS service provider.

You can make DNS changes more optimally by taking a staged approach to TTL modifications. For example:

- The current TTL for www.example.com (which points to an Elastic IP address) is 86,400 seconds (one day).
- Modify the TTL for www.example.com to 300 seconds (five minutes), and schedule work for two days' time.
- Monitor the increase of DNS traffic in this period.
- At the start of the day of the scheduled work, reduce the TTL for www.example.com further. Later, optionally reduce the TTL more, depending upon load on your DNS infrastructure (possibly 10 seconds).
- 10 minutes after the last change, update the A record to point to a new Elastic IP address in the new region.
- After a short period, confirm that traffic is being adequately serviced, and then increase the TTL back to five minutes (300 seconds).
- After another period of operation, return the TTL to normal level.

Elastic Load Balancing

Elastic Load Balancing (ELB) automatically distributes incoming application traffic across multiple EC2 instances.

You cannot migrate ELB to a new region. Instead, you must launch a new ELB service in the target region that contains a new set of EC2 instances spanning the Availability Zones you want within the service group.

Before a region migration, we recommend that you review the source and target Availability Zones to confirm that matching levels of zones exist. In scenarios where you discover extra Availability Zones, you might need to revise application load balancing and scalability. This could lead to further assessments of CloudWatch alarms and thresholds that are used for Auto Scaling group configuration.

Furthermore, you must add associated SSL certificates on the old ELB service to the new ELB service. You must also add health check conditions to verify EC2 instance health tests.

Launch Configurations and Auto Scaling Groups

Auto Scaling allows you to scale your Amazon EC2 capacity up or down automatically according to conditions you define.[8]

You can view the current Auto Scaling and launch configuration definitions from the AWS Management Console. Alternatively, you can use the following commands to capture this information:

```
as-describe-auto-scaling-groups -H --region <sourceregionname> >
autoscale_groups.txt
as-describe-launch-configs -H --region <sourceregionname> >
launch_configs.txt
```

These extracted Auto Scaling groups and launch configuration settings reference AMIs, security groups, and SSH key pairs as they exist in the source region. See the earlier sections on migrating these resources to the target region. Then create new Auto Scaling groups and launch configurations in the target region using new AMI IDs and security groups.

For more information on Auto Scaling groups and launch configurations, see Getting Started with Auto Scaling in the *Auto Scaling User Guide*.[9]

Considerations for Reserved Instances

Many customers take advantage of greatly reduced pricing of Reserved Instances for Amazon EC2, Amazon Redshift, Amazon Relational Database Service (Amazon RDS), and Amazon EMR. Amazon EC2 Standard Reserved Instances (or reserved cache nodes) are assigned to a specific instance type in a specific region for a period of one or three years, while Amazon EC2 Convertible Reserved Instances give you the flexibility to change the instance type.

Reserved Instances are available in three payment options: All Upfront, Partial Upfront, and No Upfront. The upfront cost and per-hour charges vary between these utilization levels, as well as between different geographic regions.

When you buy Reserved Instances, the larger the upfront payment, the greater the discount. To maximize your savings, you can pay all upfront and receive the largest discount. Partial upfront Reserved Instances offer lower discounts but give you the option to spend less upfront. You can also choose to spend nothing upfront and receive a smaller discount, allowing you to free up capital to spend in other projects.

If you purchased Reserved Instances for EC2 and want to migrate them to a different region, we recommend that you first sell them in the Reserved Instances Marketplace. As soon as they are sold, the billing switches to the new buyer and you are no longer billed for the Reserved Instances. The buyer then continues to pay the remainder of the term. To get savings over On-Demand Instances, you can either buy Reserved Instances for a shorter term in the region where you are migrating from the Reserved Instance Marketplace or purchase directly from AWS. Reserved Instance Marketplace makes it easy to "migrate" your billing to a new region. For more detailed information about how to buy and sell Reserved Instances, see Buying in the Reserved Instance Marketplace[10] and Amazon EC2 Reserved Instance Marketplace[11].

We recommend that you carefully assess cost implications of the purchase and sale of Reserved Instances before undertaking a migration to a new region.

Migrating Networking and Content Delivery Network Resources

This section covers the migration of network resources such as subnets, route tables, virtual private networks, access control lists, and Domain Name Systems.

Migrating Amazon Virtual Private Cloud

Amazon Virtual Private Cloud (VPC) lets you provision a private, isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define.

When you create a VPC, it exists within a region and spans all the Availability Zones in that region. You cannot move or migrate it to a new region. However, you can create a new VPC in a target region and potentially use the same IP address ranges that the existing VPC uses. You can list all VPCs using the following command:

```
aws ec2 describe-vpcs
```

A VPC consists of multiple components, which you need to recreate in the target region.

- **Subnets.** You must recreate the same subnets in the target VPC. You can list all subnets in a VPC using the following command:

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-abcd1234"
```

- **DHCP option set.** If you have a customized DHCP option set, you must recreate it in the target region. You can get details of the DHCP option set using the following command:

```
aws ec2 describe-dhcp-options
```

- **Internet gateways.** You must recreate internet gateways in the target region. You can do this using the following commands:

```
aws ec2 create-internet-gateway
aws ec2 attach-internet-gateway --vpc-id vpc-abcd1234
```

The internet gateway resource IDs returned by the previous command are used in the route tables.

- **NAT gateways.** You must recreate a NAT gateway in the VPC of the target region. You can list all the NAT gateways using the following command:

```
aws ec2 describe-nat-gateways
```

- **Route tables.** You must recreate the route tables in the target region. You can list all route tables for a VPC using the following command:

```
aws ec2 describe-route-tables --filters "Name=vpc-id,Values=vpc-abcd1234"
```

Note: As the resource IDs for gateways (internet gateway, NAT gateway, etc.) change in the target range, be sure you use the new resource IDs.

- **Security groups.** You must recreate the security groups in the target region. You can list all security groups using the following command:

```
aws ec2 describe-security-groups
```

- **Network access control lists (ACLs).** You must recreate the network ACL if you have made changes to it. You can list the network ACLs for a VPC using the following command:

```
aws ec2 describe-network-acls --filters "Name=vpc-id,Values=vpc-0f7ec66a"
```

- **Customer gateways:** You must recreate the customer gateway in the target region. You can list all the customer gateways using the following command:

```
aws ec2 describe-customer-gateways
```

- **Virtual private gateways.** You must recreate the virtual private gateway in the target region. You can list all the virtual private gateways using the following command:

```
aws ec2 describe-vpn-gateways
```

- **VPN.** Details about creating a VPN with the target region can be found here.[12]

Migrating AWS Direct Connect Links

AWS Direct Connect is a service that links physical infrastructure to AWS services. One or more fiber connections are provisioned in a Direct Connect location facility.

If you want to provision new links in a new region, you must request a new Direct Connect service and provision any tail circuits to their infrastructure. Charges for Direct Connect vary per geographic location.

You can terminate existing connections at any time when they're no longer required.

AWS has relationships with several different peering partners in each geographic region. You can find an updated list of AWS Direct Connect Amazon Partners that can assist with service provisioning at http://aws.amazon.com/directconnect/partners/.

Using Amazon Route 53 to Aid the Migration Process

Amazon Route 53 is a highly available DNS service that is available from all AWS Regions and edge locations worldwide. DNS can be very effective when

managing a migration scenario, as it can assist in gracefully migrating traffic from one location to another by routing traffic by single cutover or gradually. By adding new DNS records for copying an application in the destination region, you can test access to the application and choose when to cut over to the new site or region.

One approach is to use weighted resource record sets. This functionality enables you to determine what percentage of traffic to route to each particular address when using the same DNS name. For example, use the following configuration to route all traffic to the existing region and none to the new region.

```
www.mysite.com  CNAME elbname.sourceregion.com      100
www.mysite.com   CNAME elbname.destinationregion.com 0
```

When it is time to perform the migration, the weighting on these records is flipped as follows.

```
www.mysite.com  CNAME elbname.sourceregion.com      0
www.mysite.com   CNAME elbname.destinationregion.com 100
```

This causes all new DNS requests to resolve to the new region.

> **Note:** Some clients might continue to use the old address if they have cached their DNS resolution, if a long TTL exists, or if a TTL update has not been honored.

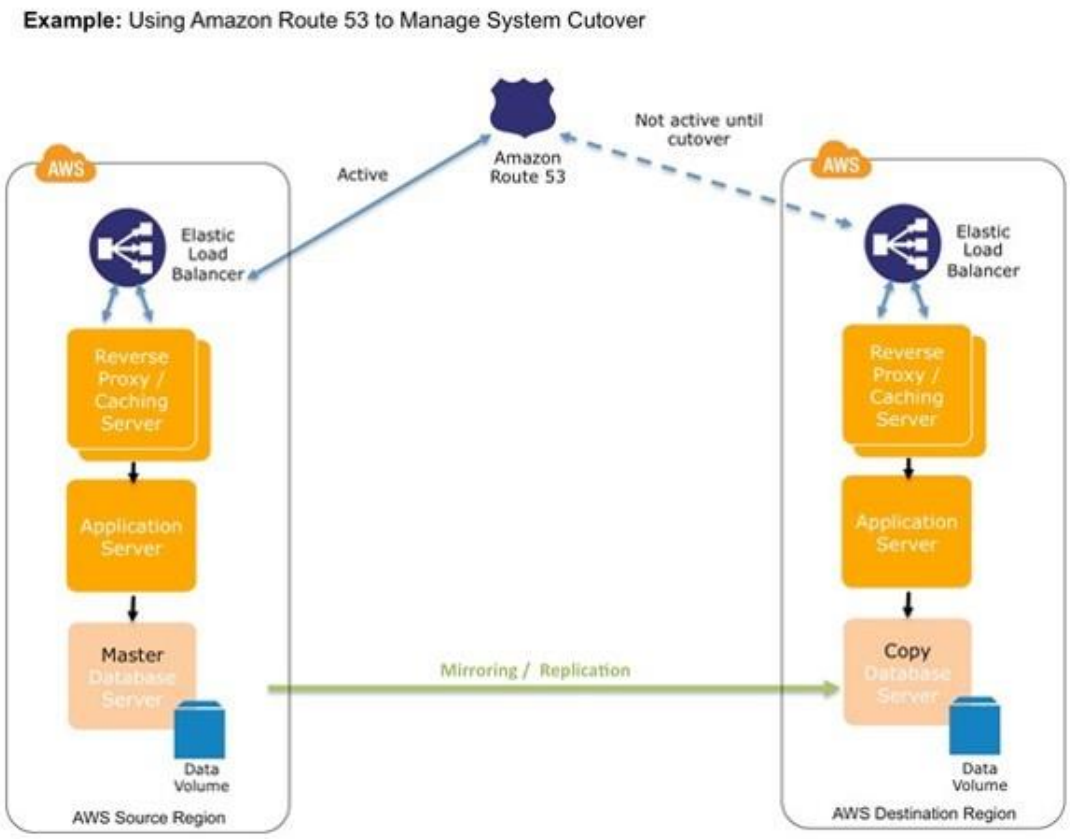


Figure 4. Using Amazon Route 53 to facilitate region migration

It is also possible to perform gradual cutover using varied weightings as long as the application supports a dual region operational model. For more information, see Working with Resource Record Sets in the *Amazon Route 53 User Guide*.[13]

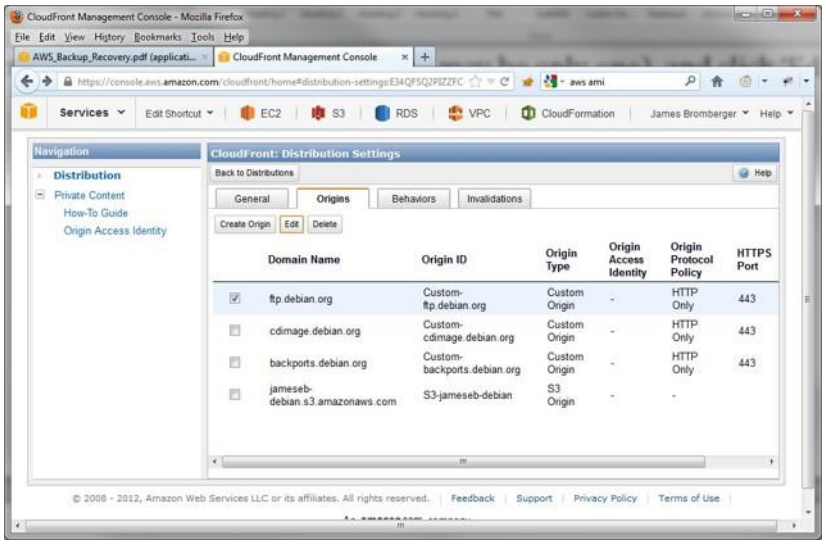
Migrating Amazon CloudFront Distributions

Amazon CloudFront is a content delivery service that operates from the numerous AWS edge locations worldwide. CloudFront delivers customer data in configuration sets known as *distributions*. Each distribution has one configured origin but can have more, as in the case of cache behaviors.

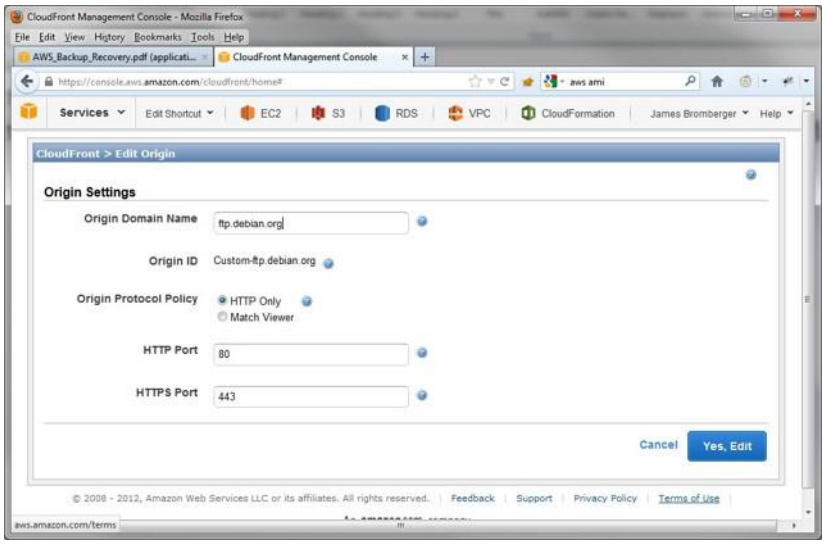
Each origin can be an S3 bucket or a web server, including web servers running from within EC2 (in any AWS Region worldwide).

To update an origin in the CloudFront console

1. Move your origin server or S3 bucket to the new region by referring to the relevant section of this document for EC2 instances or S3 buckets.
2. In the CloudFront console, select the distribution, and then choose **Distribution Settings**.
3. On the **Origins** tab, choose the origin to edit (there can be only one), and then choose **Edit**.



4. Update **Origin Domain Name** with the new server or bucket name.



5. Choose **Yes, Edit**.

For more information, see Listing, Viewing, and Updating CloudFront Distributions in the *Amazon CloudFront Developer Guide*.[14]

Migrating Storage Resources

This section covers the migration of services used for object storage, file storage, and archiving.

Migrating Amazon S3 Buckets

Amazon S3 provides a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web.

When you create an S3 bucket, it resides physically within a single AWS Region. Network latency can affect access when the bucket is accessed from another remote region. You should pay careful attention to any references to the S3 buckets and their geographic distribution, as this can introduce latency.

To migrate an S3 bucket, you need to create a new S3 bucket in the region and copy the data to it. The new bucket requires a universally unique name and cannot have the same name as the bucket in the source region. If your goal is to preserve the bucket name when copying the bucket between accounts, you need to perform an intermediate step. In this case, copy the data to an intermediary bucket first, and then delete the source bucket. After the source bucket is deleted, you must wait about 24 hours until the name becomes available again. Then, create the bucket with the old name in the new account and transfer the data using the same method as before.

For more information about Amazon S3 bucket naming rules, see Bucket Restrictions and Limitations in the *Amazon S3 User Guide*.[15]

Virtual Hosting with Amazon S3 Buckets

You might be hosting websites through the static website hosting feature of Amazon S3. For more information, see Hosting a Static Website on Amazon S3 in the *Amazon S3 User Guide*.[16]

For simplicity and user friendliness, customers often use a DNS CNAME alias for their hosted web content using Amazon S3, from a URL such as *http://bucketname.s3.amazonaws.com* to *http://my.bucketname.com*. Through a CNAME alias, the specific Amazon S3 URL endpoint is abstracted from the web browser. For more information, see Virtual Hosting of Buckets in the *Amazon S3 User Guide*.[17]

When you migrate an S3 bucket that was previously used as a static website to a new AWS Region (under a new bucket name), you can still access it using the same user-friendly name. You do this by changing the CNAME alias within the DNS record (for example, Amazon Route 53) from the old bucket name to the new.

Moving Objects Using the AWS Management Console

The AWS Management Console gives you the ability to copy or move multiple objects between S3 buckets. By manually selecting one or more objects and selecting **Cut** or **Copy** from the pop-up menu, you can paste or move these items into a target S3 bucket in another geographic region.

Figure 5. Copy an Amazon S3 object using the AWS Management Console

Copying or Moving Objects Using Third-Party Tools

To copy or move Amazon S3 objects between buckets, you can use a variety of third-party tools. You can look for AWS Partner products by searching for "Storage ISV" using the AWS Partner Solutions Finder.[18]

Copying Using the Amazon API and SDK

You can copy or move Amazon S3 objects between buckets programmatically through the Amazon SDKs and APIs. For more information about Amazon S3 object-level operations, see Operations on Objects in the *Amazon S3 API Reference*.[19]

To speed up the object copying process, you can use the PUT Object - Copy operation. A PUT Object - Copy operation performs a GET and then a PUT API call in a single operation, which can copy to a destination bucket. For more information, see PUT Object – Copy in the *Amazon S3 API Reference*.[20]

You can also use the S3DistCp tool with the Amazon EMR tool to efficiently copy large amounts of data from an S3 bucket in the source region to an S3 bucket in the target region. We recommend this for large buckets and to significantly decrease the overall migration time. S3DistCp is an extension of the open source tool DistCp that is optimized to work with AWS, particularly Amazon S3. S3DistCp uses an EMR cluster to transfer the data. You will incur additional charges for the EMR cluster. You can reduce the cost of running the EMR cluster for copying the data by using Spot Instances, as outlined here.[21] Find more details on S3DistCp here.[22]

You can also use the Amazon S3 cross-region replication feature to replicate data across AWS Regions. With cross-region replication, every object uploaded to an S3 bucket is automatically replicated from the S3 bucket in the source region to a bucket in the target region. Data that exists in the S3 bucket before you enable cross-region replication is not replicated. For migrating existing data, you can write a script to update the underlying metadata or ACLs on the object in the source bucket, which triggers a replication to the destination bucket. You can find more details on using cross-region replication here.[23]

Migrating Amazon Glacier Storage

Amazon Glacier is the AWS deep archive storage service. It is designed to handle large volumes of data that are infrequently accessed.

With Amazon Glacier you have multiple options for retrieving data, depending on the urgency of the requirement. Options include Expedited, Standard, and Bulk retrieval. You can find details on pricing options here.[24] For Standard retrieval, Amazon Glacier offers free retrieval up to 10 GB per month. Retrieval of more than this amount of data incurs additional charges.

The process used to retrieve data from Amazon Glacier depends on the way the data was archived, as follows.

Amazon S3 lifecycle policy is used to transition data from Amazon S3 to Amazon Glacier

Even though the storage class of these objects is GLACIER, you can access them only via the Amazon S3 console or APIs.

1. Use the Amazon S3 console or APIs to restore a temporary copy of an archived object to Amazon Glacier. Specify the number of days that you want the temporary copy to be available. During this period, you will incur storage charges for both Amazon Glacier and for the temporary copy.
2. Copy the S3 data from source region to target region using the steps in Migrating Amazon S3 Buckets in this whitepaper.
3. Configure an Amazon S3 lifecycle policy in the target region to transition the data from Amazon S3 to Amazon Glacier.
4. Delete the data stored in Amazon Glacier in the source region by updating the Amazon S3 lifecycle policy.

Amazon Glacier APIs are used to store the data in archives in vaults

1. Initiate an archival retrieval job to request Amazon Glacier to prepare an entire archive or a portion of the archive for subsequent download.
2. After the retrieval job completes, download the bytes to a staging area. If you are using Amazon S3 as your staging area and your archive is greater than 5 TB, you need to use byte ranges to limit the output size to less than 5 TB. Although Amazon Glacier supports individual archives of up to 40 TB, Amazon S3 has an object size limit of 5 TB.
3. You can transfer the data from the staging area to Amazon Glacier using Amazon Glacier APIs. Alternatively, if you use Amazon S3 as your staging area in the source region, you can use tools such as S3DistCp to copy the data to Amazon S3 in the target region and then use Amazon Glacier APIs to recreate the archive in Amazon Glacier in the target region.
4. Delete the temporary files created in the staging area and from Amazon Glacier in the source region.

Migrating Amazon Elastic File System

Amazon Elastic File System (Amazon EFS) provides simple, scalable file storage for use with EC2 instances in the AWS Cloud. With Amazon EFS, storage capacity is elastic. It grows and shrinks automatically as you add and remove

files, so your applications have the storage they need, when they need it. Amazon EFS file systems can automatically scale from gigabytes to petabytes of data without needing to provision storage. Amazon EFS uses the NFSv4.1 protocol and is accessible from Linux-based AMIs.

You have two options for migrating data stored in Amazon EFS from one region to another:

- Copy the files from Amazon EFS to Amazon EBS. If the data in Amazon EFS is more than what a single EBS volume (maximum size of 16 TB) can accommodate, you might need to use third-party software to distribute the data across multiple EBS volumes. Then you can migrate the EBS volumes using the cross-region EBS snapshot copy capability, and copy the files from EBS to EFS.
- Copy the files from EFS to S3. Then use the process described earlier for copying data in S3 buckets from source region to target region. In the target region copy the files from S3 to EFS.

After confirming the successful migration, make sure to delete EFS files in the source region and the temp resources (S3 and EBS) used in the transfer to avoid incurring charges for services you are not using.

Migrating AWS Storage Gateway

The AWS Storage Gateway service helps you seamlessly integrate your existing on-premises storage infrastructure and data with the AWS Cloud. It uses industry-standard storage protocols to connect existing storage applications and workflows to AWS Cloud storage services for minimal process disruption. It maintains frequently accessed data on-premises to provide low-latency performance while securely and durably storing data in Amazon S3, Amazon EBS, or Amazon Glacier.

After creating a gateway in the new region, you can migrate your data stored in Amazon S3 or Amazon EBS using the native migration capabilities of these services detailed elsewhere in this paper. The approach you take to migrate Storage Gateway depends on the interface you used in the source region:

- **File Interface.** Create a File Storage Gateway pointing to the target region.[25] Create an S3 bucket in the target region, and copy the data from the S3 bucket in the source region to target region using the

process defined earlier. You can also enable S3 cross-region replication to ensure that any updates to the S3 bucket in the source region are automatically replicated to the target region. Create a Storage Gateway file share on the S3 bucket in the target region to access your S3 files from your gateway.[26] You can update the inventory of objects maintained and stored on the gateway by initiating the refresh[27] from the AWS Storage Gateway console or by using the RefreshCache operation in the API Reference.[28]

- **Volume Interface.** Create a volume gateway pointing to the target region.[29] Create an EBS snapshot of the volume in the source region. Copy the EBS volume to the target region using the cross-region EBS snapshot copy capability. Create a Storage Gateway volume in the target region from the EBS snapshot.[30]
- **Tape Interface.** Archived tapes are stored in archive, which provides offline storage. You must first retrieve the tape from the archive back to your gateway and then from the gateway to your client machine. More details on the steps can be found here.[31] Once you retrieve the tape data to your client machine, you can store the same data in the target region by creating a tape gateway,[32] which is pointing to the target region.

You should clean up your gateway resources that are associated with your source region to avoid incurring charges for resources you don't plan to continue using.[33]

Migrating Database Resources

This section covers migration of database services for relational databases, NoSQL, caching, and data warehouse.

Migrating Amazon RDS Services

Amazon RDS is a web service that makes it easy to set up, operate, and scale a relational database in the cloud. It provides cost-efficient and resizable capacity while managing time-consuming database administration tasks, freeing you to focus on your applications and business.

Database Security Groups

Amazon RDS has its own set of security groups that restrict access to the database service, using either a CIDR notation IPv4 network address or an

Amazon EC2 security group. Each Amazon RDS security group has a name and exists in only one AWS Region (just as an Amazon EC2 security group does).

Database Instances and Data

The steps required for migrating Amazon Aurora are different from the other RDS database engines such as Oracle or MySQL. Amazon Aurora is a MySQL-compatible relational database engine that combines the speed and availability of high-end commercial databases with the simplicity and cost-effectiveness of open source databases.

You can create an Amazon Aurora database (DB) cluster as a Read Replica in the target region. Read Replicas can be created for encrypted and unencrypted DB clusters. You must encrypt the Read Replica if the source DB cluster is encrypted. When you create the Read Replica, Amazon RDS takes a snapshot of the source cluster and transfers the snapshot to the Read Replica in the target region. For each data modification made in the source databases, Amazon RDS transfers data from the source region to the Read Replica in the target region. You can find more details on the steps required for replicating data across regions here.[34]

For database engines other than Aurora, you can use the AWS Database Migration Service to migrate databases from the source region to the target region.[35] Alternatively, you can follow the steps given below. You may need to schedule downtime in an application to quiesce the data, move the database, and resume operation. Here is a high-level overview of the migration process:

1. Stop all transactions or take a snapshot (however, changes after this point in time are lost and might need to be reapplied to the target Amazon RDS DB instance).
2. Using a temporary EC2 instance, dump all data from Amazon RDS to a file:
    - For MySQL, make use of the mysqldump tool. You might want to compress this dump (see bzip or gzip).
    - For MS SQL, use the bcp utility to export data from the Amazon RDS SQL DB instance into files. You can use the SQL Server Generate and Publish Scripts Wizard to create scripts for an entire database or for just selected objects.[36]

Note: Amazon RDS does not support Microsoft SQL Server backup file restores.

   - For Oracle, use the Oracle Export/Import utility or the Data Pump feature (see http://aws.amazon.com/articles/Amazon-RDS/4173109646282306).
   - For PostgreSQL, you can use the pg_dump command to export data.

3. Copy this data to an instance in the target region using standard tools such as CP, FTP, or Rsync.
4. Start a new Amazon RDS DB instance in the target region, using the new Amazon RDS security group.
5. Import the saved data.
6. Verify that the database is active and your data is present.
7. Delete the old Amazon RDS DB instance in the source region.

For more information about importing data into Amazon RDS, see Importing Data into a DB instance in the *Amazon RDS User Guide*.[37]

Migrating Amazon DynamoDB

Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. It is a fully managed cloud database and supports both document and key-value store models.

Here is a high-level overview of the process for migrating DynamoDB from one Region to another:

- (Optional) If your source table is not receiving live traffic, you can skip this step. Otherwise, if your source table is being continuously updated, you must enable DynamoDB Streams to record these live writes while the table copy is ongoing. After the one-time table copy (given below) is complete, create a replication process that continuously consumes DynamoDB Stream records (generated from the source table) and applies them to the destination table. This will continue until the DynamoDB table in the target region catches up to the DynamoDB table

in the source region. At this point, all new writes should go to the DynamoDB table in the target region. For more information on how to do this, see Capturing Table Activity with DynamoDB Streams in the *Amazon DynamoDB Developer Guide*.[38]

- Start the table copy process. You can do this in a few ways:
    - Use the Import/Export option available via the Amazon DynamoDB console, which exports data to Amazon S3 and then imports it to a different DynamoDB table. For more information, see Exporting and Importing DynamoDB Data Using AWS Data Pipeline in the *Amazon DynamoDB Developer Guide*.[39]
    - Use the custom Java DynamoDB Import Export Tool available in the Amazon Web Services Labs repository on GitHub that performs a parallel table scan and then writes scanned items to the destination table.[40]
    - Write your own tool to perform the table copy, essentially scanning items in the source table and using parallel PutItem calls to write items into the destination table.

Whichever method you choose to migrate the data, you should consider how much read and write throughput will be required for the migration activity and make sure you provision sufficient capacity, especially if the table is serving production traffic.

Migrating Amazon SimpleDB

Amazon SimpleDB is a highly available and flexible non-relational data store that offloads the work of database administration. Developers simply store and query data items via web service requests and Amazon SimpleDB does the rest.

To copy Amazon SimpleDB data between AWS Regions, you need to create a specific job or script that extracts the data from the Amazon SimpleDB domain in one region and copies it to the relevant destination in another region. This job should be hosted on an EC2 instance. You should use the relevant SDK that suits your purposes and expertise.

Migration approaches include:

- Establishing simultaneous connections to the new and old domains, querying the existing domain for data, and then putting data into the new domain.
- Extracting data from the existing domain and storing it in a file (or set of files) and then putting that data into the new domain. We recommend that you use the API call BatchPutAttribute to increase performance and decrease the number of API calls performed.

A third-party solution that may suit your needs is also available from http://backupsdb.com/.

When you use any third-party solution, we recommend that you share only specifically secured IAM user credentials that are deleted after the migration takes place.

Migrating Amazon ElastiCache

Amazon ElastiCache is a web service that makes it easy to deploy, operate, and scale an in-memory cache in the cloud. The service improves the performance of web applications by allowing you to retrieve information from fast, managed, in-memory data stores, instead of relying entirely on slower disk-based databases. ElastiCache supports two open source in-memory engines: Redis and Memcached.

Here is an overview of the steps required to migrate an Amazon ElastiCache cluster running Redis:

1. Take a manual backup of the ElastiCache cluster. More details for carrying out a manual backup can be found here.[41] The backup consists of the cluster's metadata and all of the data in the cluster.
2. Export the backup to Amazon S3 using the ElastiCache console, the AWS CLI, or the ElastiCache API. More details for exporting the backup to Amazon S3 can be found here.[42]
3. Copy the backup data from the S3 bucket in the source region to the target region using the process defined earlier.
4. Restore the ElastiCache cluster from the backup in the target region. The restore operation creates a new Redis cluster and populates it.[43]

For an ElastiCache cluster using Memcached, the recommended approach is to start a new ElasticCache cluster, and let it start to populate itself through application usage.

Migrating Amazon Redshift

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse that makes it simple and cost-effective to analyze all your data using your existing business intelligence tools

We recommend that you pause updates to the Amazon Redshift cluster during the migration process.

Here is a high-level overview of the steps for moving the entire cluster:

- Use cross-region snapshot functionality to create a snapshot in the target region. Find more details for creating a cross-region snapshot here.[44]
- Restore the cluster from the snapshot. When you do, Amazon Redshift creates a new cluster with all the snapshot data on the new cluster. Find more details for restoring a cluster from a snapshot here.[45]

Here is a high-level overview of the steps for moving specific tables:

1. Connect to the Amazon Redshift cluster in the source region and use the Unload command to export data from Amazon Redshift to Amazon S3.
2. Copy your S3 data from the source region to the target region using the steps given earlier.
3. Create an Amazon Redshift cluster and the required tables in the target region.
4. Use the COPY command to load data from Amazon S3 to the required tables.

Migrating Analytics Resources

This section covers migration of analytics services for interactive query, Hadoop, and Elasticsearch.

Migrating Amazon Athena

Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL.

We recommend that you run Athena in the same region where the S3 bucket resides. Running Athena and Amazon S3 in different regions will result in an increase in latency and inter-region data transfer costs. Therefore, you should first migrate your S3 data from the source region to the target region and then run Athena against your S3 data in the target region.

Migrating Amazon EMR

Amazon EMR provides a managed Hadoop framework that makes it easy, fast, and cost-effective to process vast amounts of data across dynamically scalable EC2 instances.

The EMR cluster must be recreated in the target region. Migration of data from the source region to the target region will depend on whether the data is stored in Amazon S3 or the Hadoop Distributed File System (HDFS). If the data is stored in Amazon S3, you can follow the steps given earlier to migrate S3 data from the source region to the target region. Here is a high-level overview of the migration process if your data is stored in HDFS:

- Use the S3DistCp command to copy data residing in HDFS in the source region to Amazon S3 in the target region.
- Use S3DistCp to copy data from Amazon S3 to HDFS in the target region.

Migrating Amazon Elasticsearch Service

Amazon Elasticsearch Service (Amazon ES) is a fully managed service that makes it easy to deploy, operate, and scale Elasticsearch for log analytics, full text search, application monitoring, and more.

You will need to recreate the Amazon ES domain in the target region. Here is a high-level overview of the process of migrating the data from the source region to the target region:

- Create a manual snapshot of your Amazon ES domain. The snapshot is stored in an S3 bucket.[46]
- Copy your S3 data from the source region to the target region.
- Restore the snapshot into your Elasticsearch domain in the target region.

Migrating Application Services and Messaging Resources

This section covers migration of application services for queues, notifications, and Amazon API Gateway.

Migrating Amazon SQS

Amazon Simple Queue Service (Amazon SQS) offers a reliable, highly scalable, hosted queue for storing messages as they travel between computers.

Amazon SQS queues exist per region. To migrate the data in a queue, you need to drain the queue from the source region and insert it into a new queue in the target region.

When migrating a queue, it is important to note if you need to continue to process the messages in order or not.

When order is not important:

1. Create a new queue in the target region.
2. Configure applications to write messages to the new queue in the target region.
3. Reconfigure applications that read messages from the Amazon SQS queue in the source region to read from the new queue in the target region.
4. Have a script that repeatedly reads from the old queue and submits to the new queue.
5. Delete the old queue in the source region when it's empty.

When order is important:

1. Create a new first-in, first-out (FIFO) queue in the target region.
2. Create an additional new temporary FIFO queue in the target region.
3. Configure applications to write messages to the new FIFO queue in the target region.
4. Reconfigure applications that read messages from the SQS queue in the source region to read from the new temporary FIFO queue in the target region.
5. Have a script that repeatedly reads from the old queue and submits to the new temporary FIFO queue.
6. Delete the old queue in the source region when it's empty.
7. When the temporary FIFO queue is empty, reconfigure applications to read from the new FIFO queue. Then delete the temporary FIFO queue.

Migrating Amazon SNS Topics

Amazon Simple Notification Service (Amazon SNS) is a web service that makes it easy to set up, operate, and send notifications from the cloud.

Amazon SNS topics exist per region. You can recreate these in a target region manually through the AWS Management Console, command line tools, or direct API calls.

To list the current Amazon SNS topic in a designated region, use the following command:

```
aws sns list-topics –region <sourceregionname>
```

For more information about the Amazon SNS CLI tools, see Using the AWS Command Line Interface with Amazon SNS.[47]

Migrating Amazon API Gateway

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. Amazon API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic

management, authorization and access control, monitoring, and API version management.

Here is a high-level overview of the steps required for migrating Amazon API Gateway from the source region to the target region:

1. Export the API from the API Gateway into a Swagger file using the API Gateway Export API.[48]
2. Copy the Swagger file to the target region using standard tools like CP, FTP, or rsynch.
3. Import the Swagger file to create the API in the API Gateway in the target region.[49]

Migrating Deployment and Management Resources

Migrating with AWS CloudFormation

AWS CloudFormation gives developers and systems administrators an easy way to create and manage a collection of related AWS resources. It also enables provisioning and updating those resources in an orderly and predictable fashion.

You can use AWS CloudFormation sample templates or create your own templates to describe the AWS resources and any associated dependencies or runtime parameters required to run applications. For more information, see What is AWS CloudFormation?[50]

While many customers use AWS CloudFormation to create development, test, and multiple production environments within a single AWS Region, these same templates can be reused in other regions. You can address disaster recovery and region migration scenarios by running such a template with minor modifications in another region.

Commonly, AWS CloudFormation templates can be readily used by changing the mapping declarations to substitute region-specific information, such as the unique IDs for AMIs. These can vary across regions, as shown below.

```
"Mappings" : {
"RegionMap" : {
"us-east-1"     : { "AMI" : "ami-97ed27fe" },
"us-west-1"     : { "AMI" : "ami-59c39c1c" },
"us-west-2"     : { "AMI" : "ami-9e901dae" },
"eu-west-1"     : { "AMI" : "ami-87cef2f3" },
"ap-southeast-1" : { "AMI" : "ami-c44e0b96" },
"ap-northeast-1" : { "AMI" : "ami-688a3d69" },
"sa-east-1"     : { "AMI" : "ami-4e37e853" }
}
}
```

For more information on mapping declarations, see Mapping in the *AWS CloudFormation User Guide*.[51]

Capturing Environments by Using CloudFormer

CloudFormer is a template creation tool that enables you to create AWS CloudFormation templates from the pre-existing AWS resources.

You provision and configure application resources using your existing processes and tools. After these resources are provisioned within your environment within an AWS Region, the CloudFormer tool takes a snapshot of the resource configurations. The tool places these resources in an AWS CloudFormation template, enabling you to launch copies of the application environment through the AWS CloudFormation console.

The CloudFormer tool creates a starting point for an AWS CloudFormation template that you can customize further. For example, you can:

- Add parameters to enable stacks to be configured at launch time.
- Add mappings to allow the template to be customized to the specific environments and geographic regions.
- Replace static values with the Ref and Fn::GetAtt functions to flow property data between resources, where the value of one property is dependent on the value of a property from a different resource.
- Fill in your EC2 instance user data to pass parameters to EC2 instances at launch time.

- Customize Amazon RDS DB instance names and master passwords.

For more information on setting up CloudFormer to capture a customer resource stack, see http://www.youtube.com/watch?v=KIpWnVLeP8k.[52]

For more details on steps required to create a CloudFormation template using CloudFormer, see Using CloudFormer to Create AWS CloudFormation Templates from Existing AWS Resources.[53]

API Implications

When programmatic access is required to connect to AWS Regions, publically defined endpoints must be used for API service requests. While some web services allow you to use a general endpoint that does not specify a region, these generic endpoints do resolve to the service's specific regional endpoint.

For the authoritative list of current regions and service endpoint URLs, see AWS Regions and Endpoints.[54]

Updating Customer Scripts and Programs

You may need to update your self-developed scripts and programs that interact with the AWS API (either directly or using one of the SDKs or command line tools) to ensure that they are communicating with the appropriate regional endpoint.

Each SDK has its own format for specifying the region being accessed. The command line tools generally support the `–region` parameter.

Important Considerations

Do not leave your AWS certificate or private key on the disk. Clear out the shell history file, in case you typed secret information in commands or in environment variables.

Do not leave any password active on accounts. Make sure that the image does not include the public SSH key in the `authorized_keys` files. This leaves a back door into other people's servers, even if they do not intend to use it.

It is good practice to use the options [`--region`], [`--kernel`], [`--ramdisk`] explicitly whenever applicable, even though those options are optional.

Verify whether any IP address associations are associated with the AMI. If so, remove them or modify them with the correct details post migration.

Conclusions

When you undertake any type of system migration, we recommend comprehensive planning and testing. You should be sure to plan all elements of the migration, with fail-back processes for unanticipated outcomes. AWS makes this process easier by enabling cost-effective testing and the ability to retain the existing system infrastructure until the migration is successfully completed.

Contributors

The following individuals and organizations contributed to this document:

- Dhruv Singhal, Head Solutions Architect, AISPL
- Vijay Menon, Solutions Architect, AISPL
- Raghuram Balachandran, Solutions Architect, AISPL
- Lee Kear, Solutions Architect, AWS
- Paul Reed, Sr. Product Manager, AWS

Notes

[1] http://aws.amazon.com/about-aws/globalinfrastructure/regional-product-services

[2] http://docs.aws.amazon.com/IAM/latest/UserGuide/reference_identifiers.html#Identifiers_ARNs.

[3] http://docs.aws.amazon.com/general/latest/gr/aws-security-credentials.html

[4] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-network-security.html

[5] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/CopyingAMIs.html

[6] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-copy-snapshot.html

[7] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSPerformance.html

[8] http://aws.amazon.com/ec2

[9] http://docs.aws.amazon.com/autoscaling/latest/userguide/GettingStartedTutorial.html

[10] https://aws.amazon.com/ec2/pricing/reserved-instances/buyer/

[11] http://aws.amazon.com/ec2/reserved-instances/marketplace/

[12] http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_VPN.html

[13] http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/rrsets-working-with.html

[14] http://docs.amazonwebservices.com/AmazonCloudFront/latest/DeveloperGuide/HowToUpdateDistribution.html

[15] http://docs.amazonwebservices.com/AmazonS3/latest/dev/BucketRestrictions.html

[16] http://docs.amazonwebservices.com/AmazonS3/latest/dev/WebsiteHosting.html

[17] http://docs.amazonwebservices.com/AmazonS3/latest/dev/VirtualHosting.html

[18] https://aws.amazon.com/partners/find/results/?keyword=Storage+ISV

[19] http://docs.amazonwebservices.com/AmazonS3/latest/API/RESTObjectOps.html

[20] http://docs.amazonwebservices.com/AmazonS3/latest/API/RESTObjectCOPY.html

[21] http://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-instance-purchasing-options.html#emr-spot-instances

[22] http://docs.aws.amazon.com/emr/latest/ReleaseGuide/UsingEMR_s3distcp.html

[23] http://docs.aws.amazon.com/AmazonS3/latest/dev/crr.html

[24] https://aws.amazon.com/glacier/pricing/

[25] http://docs.aws.amazon.com/storagegateway/latest/userguide/create-gateway-file.html

[26] http://docs.aws.amazon.com/storagegateway/latest/userguide/GettingStartedCreateFileShare.html

[27] http://docs.aws.amazon.com/storagegateway/latest/userguide/managing-gateway-file.html#refresh-cache

[28] http://docs.aws.amazon.com/storagegateway/latest/APIReference/API_RefreshCache.html

[29] http://docs.aws.amazon.com/storagegateway/latest/userguide/create-volume-gateway.html

[30] http://docs.aws.amazon.com/storagegateway/latest/userguide/GettingStartedCreateVolumes.html

[31] http://docs.aws.amazon.com/storagegateway/latest/userguide/backup_netbackup-vtl.html#GettingStarted-retrieving-tapes

[32] http://docs.aws.amazon.com/storagegateway/latest/userguide/create-tape-gateway.html

[33] http://docs.aws.amazon.com/storagegateway/latest/userguide/deleting-gateway-common.html

[34] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Aurora.Replication.CrossRegion.html

[35] https://aws.amazon.com/documentation/dms/

[36] http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/SQLServer.Procedural.Importing.Snapshots.html#SQLServer.Procedural.Exporting.SSGPSW

[37] http://docs.amazonwebservices.com/AmazonRDS/latest/UserGuide/ImportData.html

[38] http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html

[39] http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBPipeline.html

[40] https://github.com/awslabs/dynamodb-import-export-tool

[41] http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/backups-manual.html

[42] http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/backups-exporting.html

[43] http://docs.aws.amazon.com/AmazonElastiCache/latest/UserGuide/backups-restoring.html#backups-restoring-CON

[44] http://docs.aws.amazon.com/redshift/latest/mgmt/managing-snapshots-console.html#snapshot-crossregioncopy-configure

[45] http://docs.aws.amazon.com/redshift/latest/mgmt/managing-snapshots-console.html#snapshot-restore

[46] http://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/es-managedomains.html#es-managedomains-snapshot-create

[47] http://docs.aws.amazon.com/cli/latest/userguide/cli-sqs-queue-sns-topic.html

[48] http://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-export-api.html

[49] http://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-import-api.html

[50] http://docs.amazonwebservices.com/AWSCloudFormation/latest/UserGuide/Welcome.html

[51] http://docs.amazonwebservices.com/AWSCloudFormation/latest/UserGuide/mappings-section-structure.html

[52] http://www.youtube.com/watch?v=KIpWnVLeP8k

[53] http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-using-cloudformer.html

[54] http://docs.aws.amazon.com/general/latest/gr/rande.html