Hosting Static Websites on AWS

Prescriptive Guidance

January 2016

Contents

Abstract

In this whitepaper, we cover comprehensive architectural guidance for developing, deploying, and managing static websites on Amazon Web Services (AWS) while keeping both operational simplicity and business requirements in mind. We also recommend an approach that provides these business benefits: 1) the cost of operation is insignificant, 2) there is little or no management required, and 3) the resulting website is highly scalable, resilient, and reliable.

This whitepaper first reviews how static websites are hosted in traditional hosting environments. Then it explores a simpler and more cost-efficient approach using Amazon Simple Storage Service (Amazon S3). Then it shows how you can enhance the AWS architecture to layer on functionality and improve quality of service by using Amazon CloudFront.

Introduction

As enterprises become more digital operations, we see a proliferation of many kinds of websites. They span a wide spectrum, from mission-critical e-commerce sites to departmental apps, and from business-to-business (B2B) portals to marketing sites. Many factors such as business value, mission criticality, service level agreements (SLAs), quality of service, and information security will drive the choice of architecture and technology stack.

The simplest form of website architecture is the *static website*, where users are served static content (HTML, images, video, JavaScript, style sheets, etc.). Some examples include brand micro sites, marketing websites, and intranet information pages. Static websites are straightforward in one sense, but they still can have demanding requirements in terms of scalability, availability, and service level guarantees. For example, a marketing site for a consumer brand might need to be prepared for an unpredictable onslaught of visitors when a new product is launched.

Static Versus Dynamic Website

What is the difference between a “static” website and a “dynamic” website? The word *static* refers to the fact that a static website delivers content in the same format in which it is stored. No server-side code execution is required. For example, if a static website consists of HTML documents displaying images, it will deliver the HTML and images as-is to the browser, without altering the contents of the files.

Static websites can be delivered to web browsers on desktops, tablets, or mobile devices. They usually consist of a mix of HTML documents, images, videos, CSS style sheets, and JavaScript files. Static doesn’t have to mean boring—static sites can provide client-side interactivity as well. Using HTML5 and client-side JavaScript technologies such as JQuery, AngularJS, React, and Backbone, you can deliver rich user experiences that are engaging and interactive.

Some examples of static sites include:

- Marketing websites
- Product landing pages

- Micro-sites that display the same content to all users
- Team homepages
- Small, self-contained websites for a department, project, or team
- A site listing of available assets (e.g., image files, video files, press releases), which allow the user to download the files as-is
- Proofs-of-concept used in the early stages of web development, to test user experience flows and gather feedback

By contrast, *dynamic* websites can display dynamic or personalized content. They usually interact with data sources and web services, and require code development expertise to create and maintain. As an example, a sports news site may display information based on the visitor's preferences, and use server-side code to display the sport scores. Other examples of dynamic sites are e-commerce shopping sites, news portals, social networking sites, finance sites, and most other websites that display ever-changing information.

Static websites load faster than dynamic ones, since content is delivered as-is and can be cached by a content delivery network (CDN), and the web server doesn't need to perform any application logic or database queries. They're also relatively inexpensive to develop and host. However, maintaining large static websites can be cumbersome without the aid of automated tools, and static websites can't deliver personalized information.

Static websites are most suitable when the content is infrequently updated. After the content evolves in complexity or needs to be frequently updated, personalized, or dynamically generated, it's time to consider a dynamic website architecture.

Core Architecture

In a traditional (non-AWS) architecture, web servers serve up static content. Typically, content is managed using a content management system (CMS), and multiple static sites will be hosted on the same infrastructure. The content is stored on local disks, or on a file share on network-accessible storage. A sample file system structure might look like the structure that follows.

```
├─ css/
│   ├─ css/
│   ├─ main.css
│   └─ navigation.css
├─ images/
│   ├─ banner.jpg
│   └─ logo.jpg
├─ index.html
├─ scripts/
│   ├─ script1.js
│   └─ script2.js
├─ section1.html
└─ section2.html
```

A network firewall protects against unauthorized access. It is common to deploy multiple web servers behind a load balancer for high availability and scalability. Since pages are static, the web servers do not need to maintain any state or session information and the load balancer doesn't need to implement session affinity (aka "sticky sessions"). In a traditional (non-AWS) hosting environment, a basic architecture looks like this:

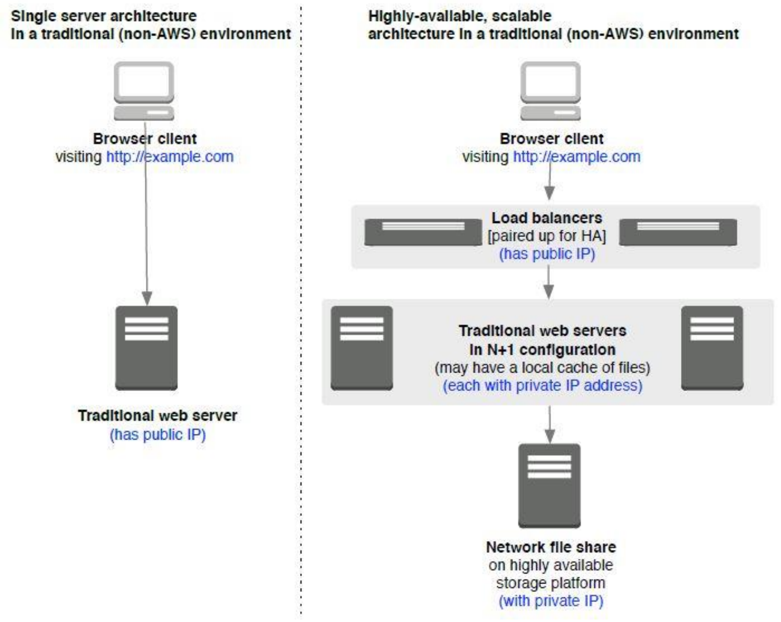

Single server architecture in a traditional (non-AWS) environment
Browser client visiting http://example.com
Traditional web server (has public IP)
Highly-available, scalable architecture in a traditional (non-AWS) environment
Browser client visiting http://example.com
Load balancers [paired up for HA] (has public IP)
Traditional web servers in N+1 configuration (may have a local cache of files) (each with private IP address)
Network file share on highly available storage platform (with private IP)

Figure 1: Basic architecture of a traditional hosting environment

Moving to an AWS Architecture

To translate a traditional hosting environment to an AWS architecture you could use a so-called "lift-and-shift" approach. A lift and shift is a like-for-like substitution of AWS services instead of the traditional environment.

You can run Linux or Windows web servers on Amazon Elastic Compute Cloud (Amazon EC2). You can use Elastic Load Balancing (ELB) to load-balance and distribute the web traffic. The static content can be stored on Elastic Block Store (Amazon EBS), or Amazon Elastic File System (Amazon EFS). You can deploy your Amazon EC2 instances in an Amazon Virtual Private Cloud (Amazon VPC), which is your isolated and private virtual network in the AWS cloud. This gives you full control over the network topology, firewall configuration, and routing rules. The web servers can be spread across multiple Availability Zones (AZs) for high availability, even if an entire data center were to be down. You can use Auto Scaling to automatically add servers during high traffic periods, and then scale back when traffic quiets down.

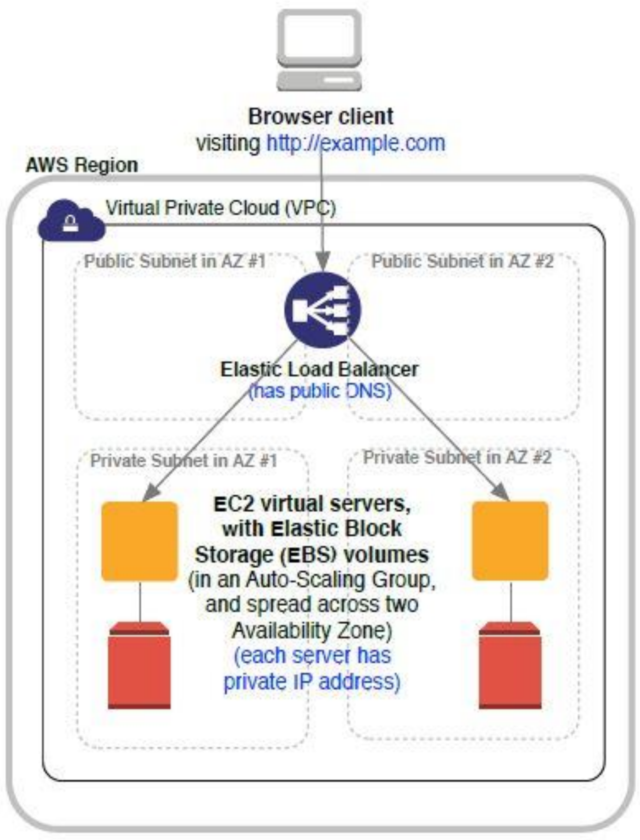

Browser client
visiting http://example.com
AWS Region
Virtual Private Cloud (VPC)
Public Subnet in AZ #1
Public Subnet in AZ #2
Elastic Load Balancer
(has public DNS)
Private Subnet in AZ #1
Private Subnet in AZ #2
EC2 virtual servers, with Elastic Block Storage (EBS) volumes
(in an Auto-Scaling Group, and spread across two Availability Zone)
(each server has private IP address)

Figure 2: AWS architecture for a "Lift and Shift"

This basic AWS architecture would be a highly effective solution. You gain the security, scalability, cost, and agility benefits of running in AWS. This architecture benefits from AWS world-class infrastructure and security operations. By using Auto Scaling, the website is ready for traffic spikes, which you might want for product launches and viral websites. With AWS, you only pay for what you use, and there's no need to over-provision for peak capacity. In addition, you gain from increased agility because AWS services are available on demand. (Compare this to the traditional process in which provisioning servers, storage, or networking can take weeks.) You don't have to manage infrastructure, so this frees up time and resources to create business-differentiating value. In many ways, this initial AWS architecture is a major win over the non-AWS one.

However, you can do better.

AWS challenges traditional IT assumptions and enables new "cloud-native" architectures. You can architect a modern static website without needing a single web server.

Use Amazon S3 Website Hosting to Host Without a Single Web Server

Amazon Simple Storage Service (Amazon S3) can be used to host static websites without a need for a web server. The website will be highly performant and scalable at a fraction of the cost of a traditional web server.[1] Amazon S3 is storage for the cloud, providing you with secure, durable, highly scalable object storage. It's easy to use: a simple web services interface allows you to store and retrieve any amount of data from anywhere on the web.[2]

You can start by creating an Amazon S3 bucket, enabling the Amazon S3 website hosting feature, and configuring access permissions for the bucket. After you upload files, Amazon S3 takes care of serving your content to your visitors.

Amazon S3 provides HTTP web-serving capabilities, and the content can be viewed by any browser. You also need to configure Amazon Route 53, a managed Domain Name System (DNS) service, to point *http://example.com* to your Amazon S3 bucket.

This core architecture is as simple as it sounds. Figure 3 illustrates what it looks like.

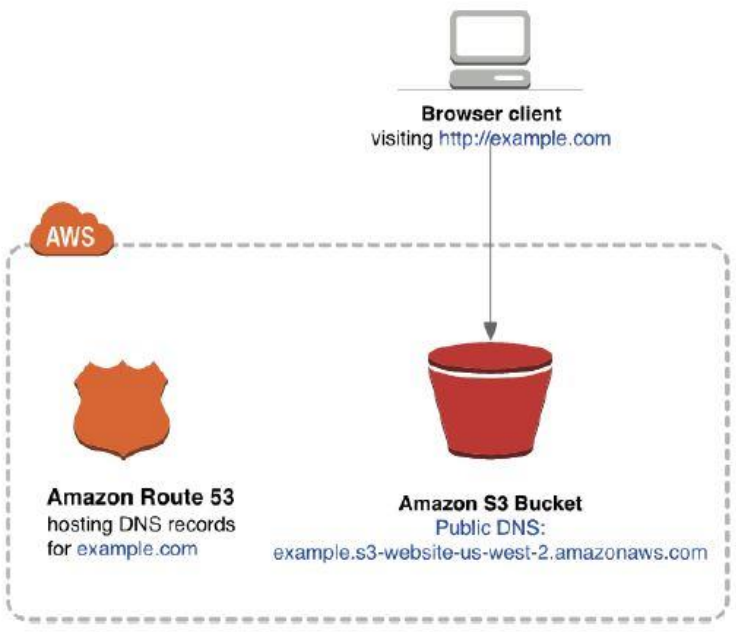


Figure 3: Amazon S3 website hosting

There are no Windows or Linux servers to manage! No need to provision machines, install operating systems, or fine-tune web server configurations. There's also no need to manage storage infrastructure (e.g., SAN, NAS) because Amazon S3 provides practically limitless cloud-based storage. Fewer moving parts means fewer troubleshooting headaches.

Scalability and Availability

Amazon S3 is inherently scalable. For popular websites, the Amazon S3 architecture will scale seamlessly to serve thousands of HTTP requests per second without any changes to the architecture.

In addition, by hosting with Amazon S3, the website is inherently highly available. Amazon S3 is designed for 99.99% availability, and carries a service level agreement (SLA) of 99.9% availability.[3] Amazon S3 gives you access to the same highly scalable, reliable, fast, and inexpensive infrastructure that Amazon uses to run its own global network of websites. As soon as you upload files to Amazon S3, Amazon S3 automatically replicates your content across multiple data centers. Even if an entire AWS data center were to be impaired, your static website would still be running and available to your end users.

Compare this with traditional non-AWS costs for implementing "active-active" hosting for important projects. Active-active, or deploying web servers in two distinct data centers, is prohibitive in terms of server costs and engineering time. As a result, traditional websites are usually hosted in a single data center, because most projects can't justify the cost of "active-active" hosting.

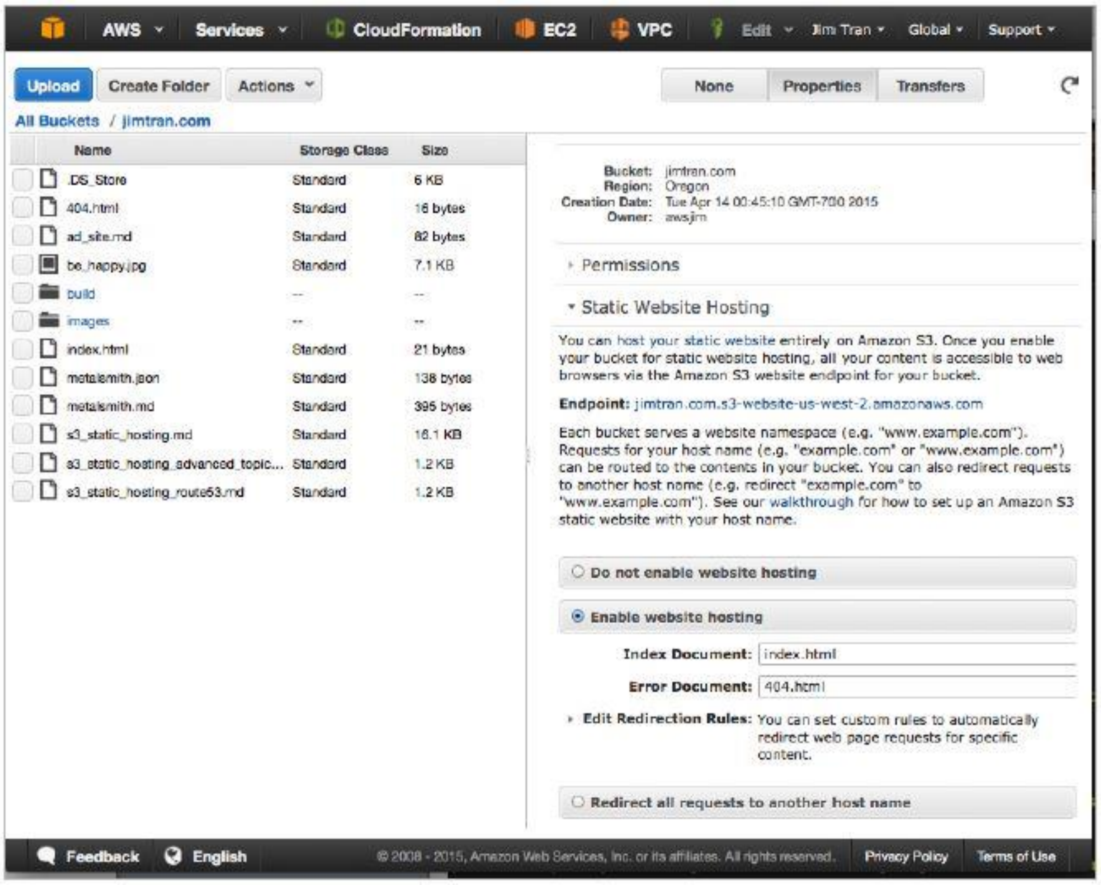
Configuration Basics

Getting started is easy. First, you need an Amazon S3 bucket. When you first create a bucket, you select the AWS Region in which the files will be geographically stored.[4] Select a region based on proximity to your visitors, proximity to your corporate datacenters, and/or regulatory or compliance requirements (e.g., some countries have restrictive data residency regulations). Since Amazon S3 supports virtual-host style access to your S3 buckets, the bucket name must comply with DNS naming conventions.[5] If you plan to use your own custom domain/subdomain, such as *example.com* or *www.example.com*, your bucket name must be the same as your domain/subdomain. For example, a website available at *http://www.example.com* needs to be in a bucket named www.*example.com*. Each AWS account can have a maximum of 1000 buckets.

After your Amazon S3 bucket is created, toggle on the static website hosting feature for the bucket.[6] This will generate an Amazon S3 website endpoint. You will be able to access your Amazon S3-hosted website at the following URL:

```
http://<bucket-name>.s3-website-<AWS-region>.amazonaws.com
```

For small, non-public websites, the Amazon S3 website endpoint is probably adequate. You can also use internal DNS to point to this endpoint. For a public-facing website, we recommend using a custom domain name instead of the provided Amazon S3 website endpoint. This way users can see user-friendly URLs in their browsers. As mentioned earlier, if you plan to use a custom domain name, your bucket name must match the domain name. For custom root domains (such as *example.com*), only Amazon Route 53 can configure a DNS record to point to the Amazon S3-hosted website. For non-root subdomains (such as *www.example.com*), any DNS service (including Amazon Route 53) will be able to create a CNAME entry to the subdomain. See the Amazon S3 documentation for more details on how to associate domain names with your website.[7]



Name	Storage Class	Size
.DS_Store	Standard	6 KB
404.html	Standard	16 bytes
ad_site.md	Standard	82 bytes
be_happy.jpg	Standard	7.1 KB
build	--	--
images	--	--
index.html	Standard	21 bytes
metalsmith.json	Standard	138 bytes
metalsmith.md	Standard	395 bytes
s3_static_hosting.md	Standard	16.1 KB
s3_static_hosting_advanced_topic...	Standard	1.2 KB
s3_static_hosting_route53.md	Standard	1.2 KB

Bucket: jimtran.com
Region: Oregon
Creation Date: Tue Apr 14 00:45:10 GMT-700 2015
Owner: awsjim

Permissions

Static Website Hosting

You can host your static website entirely on Amazon S3. Once you enable your bucket for static website hosting, all your content is accessible to web browsers via the Amazon S3 website endpoint for your bucket.

Endpoint: jimtran.com.s3-website-us-west-2.amazonaws.com

Each bucket serves a website namespace (e.g. "www.example.com"). Requests for your host name (e.g. "example.com" or "www.example.com") can be routed to the contents in your bucket. You can also redirect requests to another host name (e.g. redirect "example.com" to "www.example.com"). See our walkthrough for how to set up an Amazon S3 static website with your host name.

○ Do not enable website hosting

● Enable website hosting

Index Document: index.html

Error Document: 404.html

Edit Redirection Rules: You can set custom rules to automatically redirect web page requests for specific content.

○ Redirect all requests to another host name

Feedback
English
© 2008 - 2015, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use

Figure 4: Configuring static website hosting using the Amazon S3 console

The Amazon S3 website hosting configuration screen in the AWS Management Console presents additional options to configure. Some of the key ones are as follows:

- You can configure a default page that users will see if they visit the domain name directly (without specifying a specific page).[8] You can also specify a custom "404 - Page Not Found" error page if the user stumbles onto a non- existent page.
- By default, logging is disabled. You'll also want to enable it so you can have access to the raw web access logs.[9]

- Add tags to your Amazon S3 bucket. These tags will be useful later on when you want to analyze your AWS spend by project.[10]

A quick note about Amazon S3 object naming*:* In Amazon S3, a bucket is a flat container of objects; it doesn't provide a hierarchical organization as the file system on your computer does. That said, there is a straightforward mapping between a file system's folders/files to Amazon S3 objects. The example that follows shows how folders/files are mapped to Amazon S3 objects. Most third- party tools, as well as the AWS Management Console and AWS Command Line Tools (AWS CLI) will handle this mapping transparently for you; however, it is good to be aware of this fact.[11] It's also a good habit to use lower-case characters for file and folder names. Case is important and simply easier to manage when all characters are lowercase.

```
Hierarchical file system        S3 object names in the S3 bucket

├ css/                          css/
|                               ├ main.css css/main.css
|                               └ navigation.css  css/navigation.com
├ images/                       images/
|                               ├ banner.jpg      images/banner.jpg
|                               └ logo.jpg images/logo.jpg
├ index.html                    index.html
├ scripts/                      scripts/
|                               ├ script1.js      scripts/script1.js
|                               └ script2.js      scripts/script2.js
├ section1.html                 section1.html
└ section2.html                 section2.html
```

Uploading Content

On AWS, uploading content is straightforward. Continue to design your static website using your website authoring tool of choice. Most web design and authoring tools can save the static content on your local hard drive. Then simply upload the HTML, images, JavaScript files, CSS files, and other static assets into your Amazon S3 bucket. The deployment process consists of a single step: copy any new or modified files to the Amazon S3 bucket. You can use the AWS API, SDKs, or CLI to script this step for a fully automated deployment.

You can upload files using the AWS Management Console. You can also use AWS partner offerings such as CloudBerry, S3 Bucket Explorer, S3 Fox, and other visual management tools. The easiest way, however, is to use the AWS CLI.[12] The `s3 sync` command will recursively upload files and synchronize your Amazon S3 bucket with your local folder.[13]

```
aws s3 sync $LOCAL_FOLDER s3://$S3_BUCKET_NAME/
```

In the "Deploying variations" section of this whitepaper, you'll learn how to implement Blue/Green deployments to reduce deployment risks when updating your site.

Making Your Content Publicly Accessible

For your visitors to access content at the Amazon S3 website endpoint, the Amazon S3 objects need to have the appropriate permissions. Amazon S3 enforces a security-by-default policy. New objects in a new bucket are private by default. For example, you'll see an Access Denied error when trying to view a newly uploaded file using your web browser. To fix this, configure the content as publicly accessible. It's possible to set object-level permissions for every individual object, but that quickly becomes tedious. It's far more convenient to define an Amazon S3 bucket-wide policy. The following sample Amazon S3 bucket policy enables everyone to view all objects in a bucket:

```
{
   "Version":"2012-10-17",
   "Statement":[{
         "Sid":"PublicReadGetObject",
         "Effect":"Allow",
         "Principal": "*",
         "Action":["s3:GetObject"],

"Resource":["arn:aws:s3:::S3_BUCKET_NAME_GOES_HERE/*"]
      }
   ]
}
```

This policy defines who can view the contents of your Amazon S3 bucket. In the "Managing Administrative Access to Your AWS Resources" section of this whitepaper, there's a description of the use of AWS Identity and Access Management (IAM) policies to manage permissions for your team members.

Together, Amazon S3 bucket policies and IAM policies give you fine-grained control over who can manage and view your website.

Low Costs Encourage Experimentation

So how much will this website cost to run? Amazon S3 costs are storage plus bandwidth. The actual costs will depend on your asset file sizes, and your site's popularity (the number of visitors making browser requests). There's no minimum charge and no setup costs.

When you use Amazon S3, you pay for what you use. You're only charged for the actual Amazon S3 storage required to store the site assets.[14] These assets include HTML files, images, JavaScript files, CSS files, videos, audio files, and any other downloadable files. Your bandwidth charges will depend on the actual site traffic.[15] Small websites with few visitors will have minimal hosting costs. Popular websites that serve up large videos and images will incur higher bandwidth charges. The "Managing Costs" section of this whitepaper shows you how you can estimate and track your costs.

With Amazon S3, experimenting with new ideas is easy and cheap. If a website idea turns out to be a dud, the costs will be minimal. This should encourage you to experiment more frequently. As they say, "fail fast, fail often!" This is great for micro-sites – publish many independent micro-sites at once, run A/B tests, and keep only the winners. The "Deploying variations" section of this whitepaper shows you more detail on how to deploy A/B tests.

You've learned the basics of hosting a static website in Amazon S3. The architecture is refreshingly simple: No servers, just a single Amazon S3 bucket. It's ready to meet the real-world demands of a high-traffic website. It's highly available, resilient, scalable, and cost-efficient.

Now you'll learn how to iterate on this core architecture and make it faster, improve its performance for international visitors, and lower its cost even further.

Evolving the Architecture with Amazon CloudFront

When it comes to websites, speed matters. Web visitors enjoy and expect a fast browsing experience. Even slight page-loading delays can hurt business. In today's global economy, your site needs to be responsive and deliver page loads with low-latency. Major search engines penalize slow websites by burying their search results. The Amazon CloudFront content delivery web service integrates with other AWS products to give you an easy way to distribute content to users with low latency, high data transfer speeds, and no minimum usage commitments.

Factors Contributing to Page Load Latency

To explore factors that contribute to latency, we use the example of a user in Singapore visiting a web page hosted from an Amazon S3 bucket in the US West (Oregon) Region, in the United States. From the moment the user visits a web page to the moment it shows up in the browser, several factors contribute to latency.

- **FACTOR (1)** Time it takes for the browser (Singapore) to request the web page from Amazon S3 (US West (Oregon) Region).
- **FACTOR (2)** Then, it takes time for Amazon S3 to retrieve the page contents and serve up the page.
- **FACTOR (3)** Then, it takes time for the page contents (US West (Oregon) Region) to be delivered across the Internet to the browser (Singapore).
- **FACTOR (4)** Finally, it takes some time for the browser to parse and display the web page.

This latency is illustrated in Figure 5.

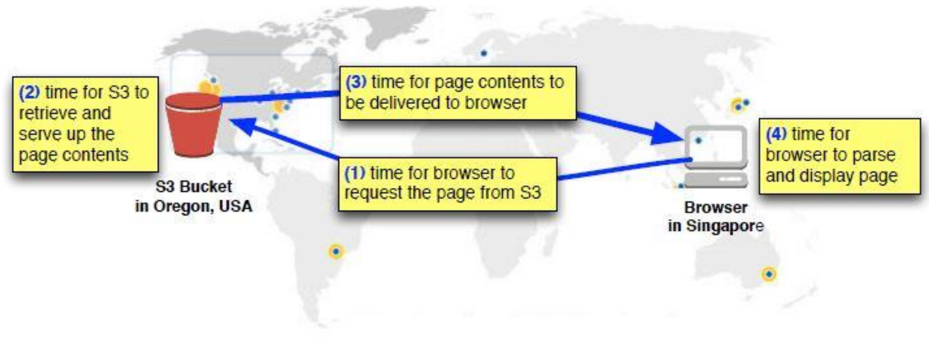

(2) time for S3 to retrieve and serve up the page contents
S3 Bucket in Oregon, USA
(3) time for page contents to be delivered to browser
(1) time for browser to request the page from S3
Browser in Singapore
(4) time for browser to parse and display page

Figure 5: Factors affecting page load latency

The good news is that AWS has already taken care of **Factor (2)** by optimizing Amazon S3 to serve up content as quickly as possible. You can improve **Factor (4)** by optimizing the actual page content (e.g., minifying CSS and JavaScript, using efficient image and video formats). However, page-loading studies consistently show that most latency is due to **Factor (1)** and **Factor (3)**. Most of the delay in accessing pages over the Internet is due to the round-trip delay associated with establishing TCP connections (the infamous three-way TCP handshake) and the time it takes for TCP packets to be delivered across long Internet distances.

To sum up: serve content as close to your users as possible. In our example, American users will experience relatively fast page load times, whereas Singaporean users will experience slower page loads. Ideally, for the Singaporean users, we would want to serve up content as close to Singapore as possible.

Speeding Up Your Amazon S3-Based Website Using Amazon CloudFront

To reduce latency and provide a better user experience, use Amazon CloudFront. Amazon CloudFront is a content delivery network (CDN) that uses a global network of edge locations for content delivery. A side benefit is that CloudFront also provides reports to help you understand how users are using your website.

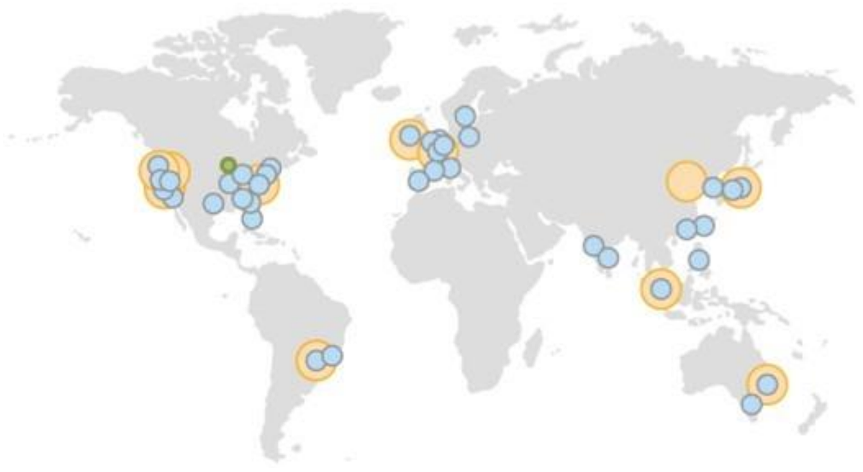


Figure 6: AWS Regions (orange circles) and Amazon CloudFront edge locations (blue circles) as of December 2015

As a CDN, Amazon CloudFront can distribute content with low latency and high data transfer rates.[16] As of this writing, there are [54 CloudFront edge locations](), all around the world—including a couple in Singapore.[17] Therefore, no matter where a visitor lives in the world, there is an Amazon CloudFront edge location that is relatively close (from an Internet latency perspective). The map above shows the Amazon CloudFront edge locations. As of December 2015, there were 11 AWS Regions (orange circles), and 54 Amazon CloudFront edge locations (blue circles). In January 2016, AWS announced its twelfth region, the Asia Pacific (Seoul) Region.

The Amazon CloudFront edge locations will cache content from an *origin server*, and deliver that cached content to the user. When creating an [Amazon CloudFront distribution](), specify your Amazon S3 bucket as the origin server.[18] The Amazon CloudFront distribution itself will have a DNS. You can refer to it using a CNAME if you have a custom domain name. To point the A record of a root domain to an Amazon CloudFront distribution, you can use Amazon Route 53 alias records, as illustrated in Figure 7.

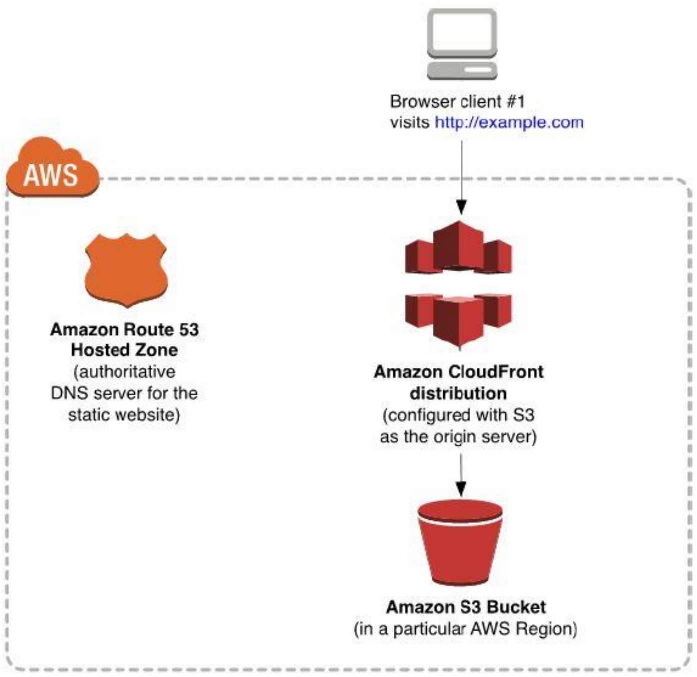


Figure 7: Using Amazon Route 53 alias records with an Amazon CloudFront distribution

To understand how CloudFront accelerates websites, it's helpful to understand what is happening under the covers. When an end user requests a web page using that domain name, CloudFront determines the best edge location to serve the content. If an edge location doesn't yet have a cached copy of the requested content, CloudFront will pull a copy from the Amazon S3 origin server and hold it at the edge location to fulfill future requests. Subsequent users requesting the same content from that edge location will experience faster page loads because that content is already cached. Figure 8 shows the flow in detail. In the "Controlling How Long Amazon S3 Content Is Cached by Amazon CloudFront" section later in this whitepaper, we cover advanced strategies.

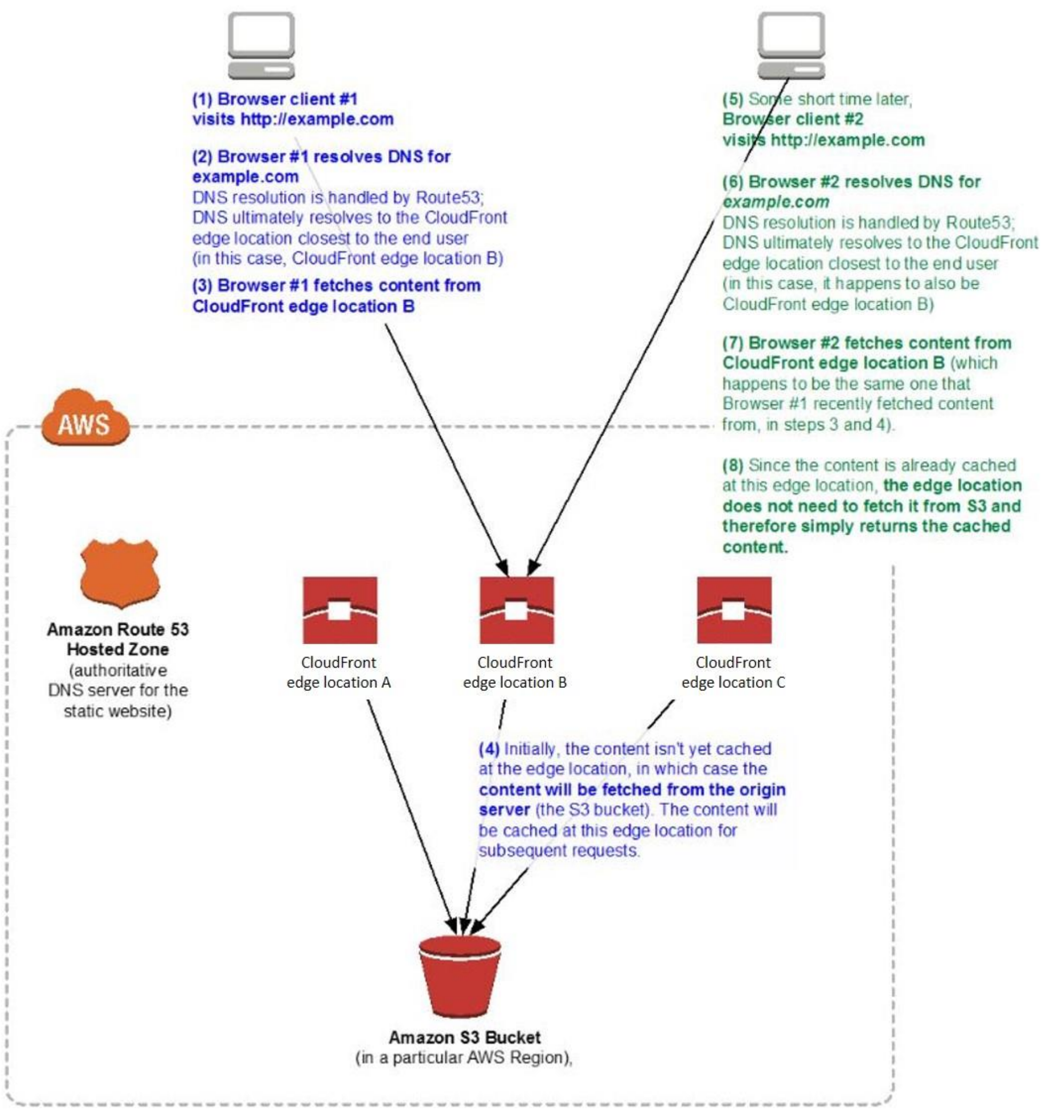

(1) Browser client #1 visits http://example.com
(2) Browser #1 resolves DNS for example.com DNS resolution is handled by Route53; DNS ultimately resolves to the CloudFront edge location closest to the end user (in this case, CloudFront edge location B)
(3) Browser #1 fetches content from CloudFront edge location B
(5) Some short time later, Browser client #2 visits http://example.com
(6) Browser #2 resolves DNS for example.com DNS resolution is handled by Route53; DNS ultimately resolves to the CloudFront edge location closest to the end user (in this case, it happens to also be CloudFront edge location B)
(7) Browser #2 fetches content from CloudFront edge location B (which happens to be the same one that Browser #1 recently fetched content from, in steps 3 and 4).
(8) Since the content is already cached at this edge location, the edge location does not need to fetch it from S3 and therefore simply returns the cached content.
AWS
Amazon Route 53 Hosted Zone (authoritative DNS server for the static website)
CloudFront edge location A
CloudFront edge location B
CloudFront edge location C
(4) Initially, the content isn't yet cached at the edge location, in which case the content will be fetched from the origin server (the S3 bucket). The content will be cached at this edge location for subsequent requests.
Amazon S3 Bucket (in a particular AWS Region),

Figure 8: How Amazon CloudFront caches content

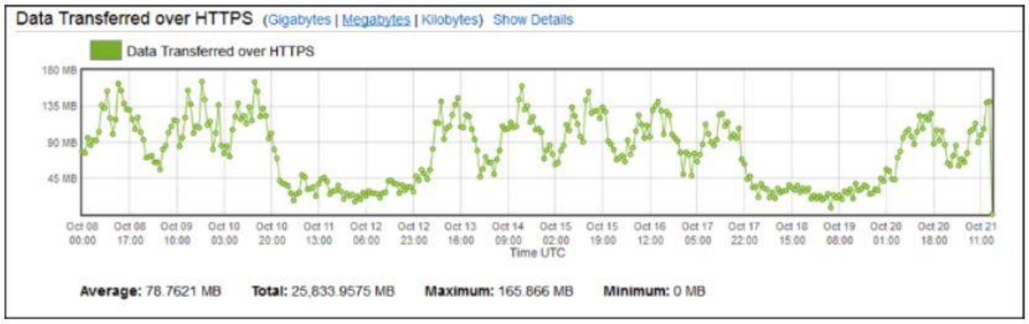
Amazon CloudFront Reports

Amazon CloudFront provides you with a rich set of reports. The reports help you understand how your visitors are using your website. You can track trends and keep a close eye on your website's performance and effectiveness. The reports are great for gaining insights by answering questions such as:

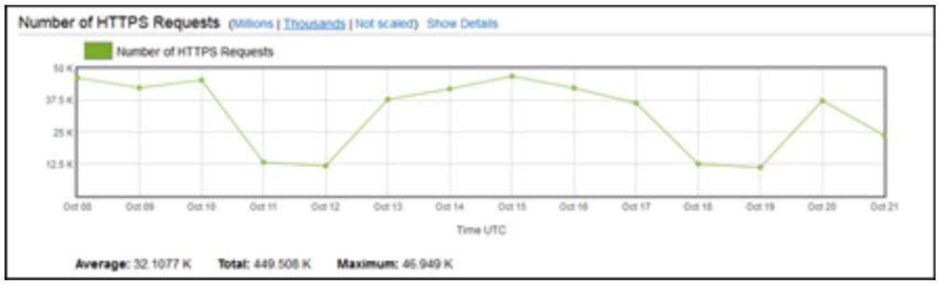
- What is the overall health of my website?
- How many visitors are viewing my website?
- Which browsers, devices, and operating systems are they using?

- Which countries are they coming from?
- Which websites are the top referrers to my site?
- What assets are the most popular ones on my site?
- How often is CloudFront caching taking place?

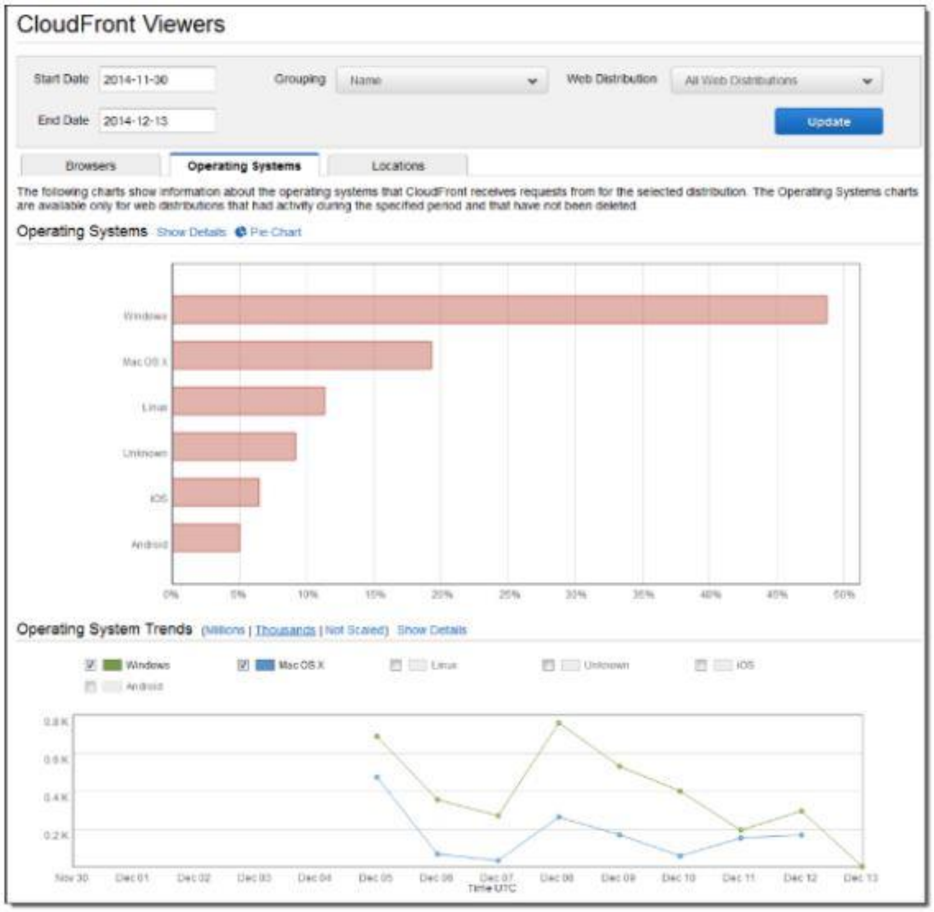
Amazon CloudFront reports are complementary with other online analytics tools, and we encourage the use of multiple reporting tools. Note that some analytics tools may require you to embed client-side JavaScript in your HTML pages; CloudFront reporting does not require any changes to your web pages. Figure 9 shows a sampling of types of Amazon CloudFront reports.



Average: 78.7621 MB **Total:** 25,833.9575 MB **Maximum:** 165.866 MB **Minimum:** 0 MB



Average: 32.1077 K **Total:** 449.508 K **Maximum:** 46.949 K



CloudFront Viewers

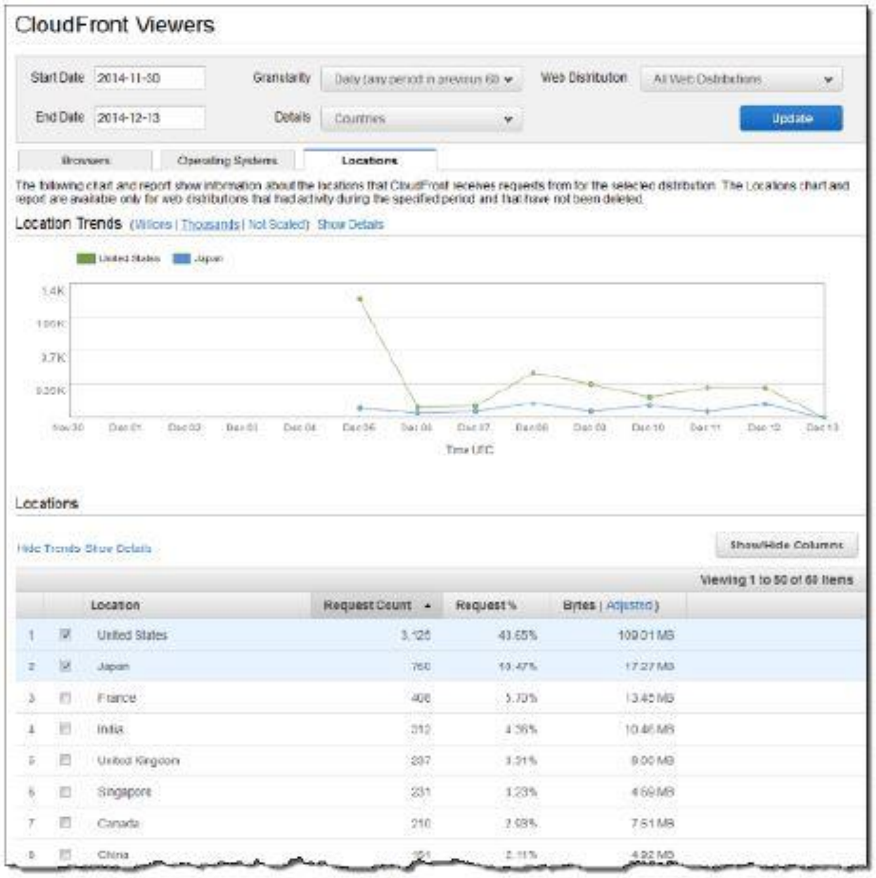
Start Date: 2014-11-30

End Date: 2014-12-13

Grouping: Name

Web Distribution: All Web Distributions

The following charts show information about the operating systems that CloudFront receives requests from for the selected distribution. The Operating Systems charts are available only for web distributions that had activity during the specified period and that have not been deleted.



CloudFront Viewers

Start Date 2014-11-30

End Date 2014-12-13

Granularity Daily (any period in previous 60

Details Countries

Web Distribution All Web Distributions

Browsers | Operating Systems | Locations

The following chart and report show information about the locations that CloudFront receives requests from for the selected distribution. The Locations chart and report are available only for web distributions that had activity during the specified period and that have not been deleted.

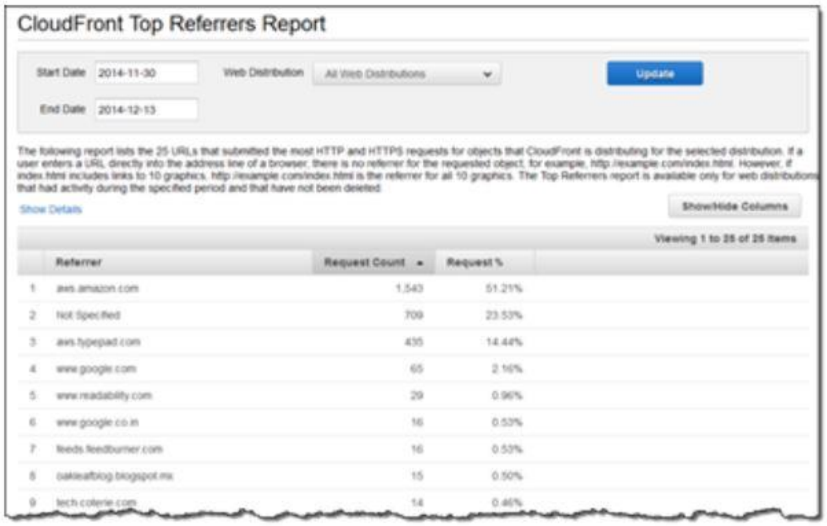
Location Trends (Millions | Thousands | Not Scaled) Show Details

Locations

Hide Trends Show Details

Viewing 1 to 50 of 60 items

		Location	Request Count ▲	Request %	Bytes (Adjusted)
1	☑	United States	3,126	43.65%	109.01 MB
2	☑	Japan	750	10.47%	17.27 MB
3	☐	France	408	5.70%	13.40 MB
4	☐	India	312	4.36%	10.46 MB
5	☐	United Kingdom	237	3.31%	9.00 MB
6	☐	Singapore	231	3.23%	4.59 MB
7	☐	Canada	210	2.93%	7.51 MB
8	☐	China	[illegible]	2.11%	4.92 MB



CloudFront Top Referrers Report

Start Date 2014-11-30

End Date 2014-12-13

Web Distribution All Web Distributions

The following report lists the 25 URLs that submitted the most HTTP and HTTPS requests for objects that CloudFront is distributing for the selected distribution. If a user enters a URL directly into the address line of a browser, there is no referrer for the requested object, for example, http://example.com/index.html. However, if index.html includes links to 10 graphics, http://example.com/index.html is the referrer for all 10 graphics. The Top Referrers report is available only for web distributions that had activity during the specified period and that have not been deleted.

Show Details

Viewing 1 to 25 of 25 items

	Referrer	Request Count ▲	Request %
1	aws.amazon.com	1,543	51.21%
2	Not Specified	709	23.53%
3	aws.typepad.com	435	14.44%
4	www.google.com	65	2.16%
5	www.readability.com	29	0.96%
6	www.google.co.in	16	0.53%
7	feeds.feedburner.com	16	0.53%
8	oakleafblog.blogspot.mx	15	0.50%
9	tech.coterie.com	14	0.46%

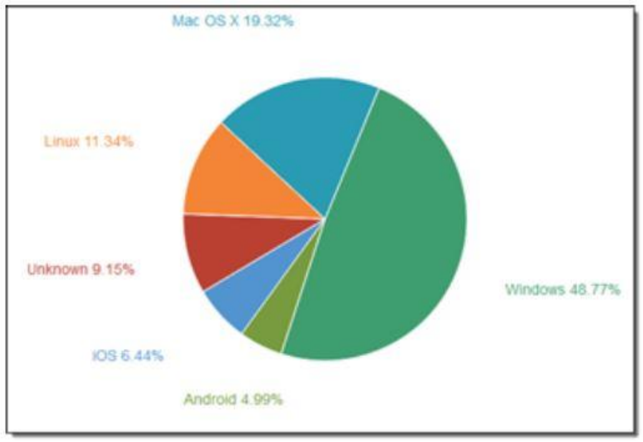


Figure 9: Various types of Amazon CloudFront reports

For any website serving a global user base, we highly recommend using CloudFront.[19] Amazon S3 is solid on its own, and adding CloudFront helps deliver a higher-quality user experience.

Let's now turn our attention to advanced topics. The remaining sections will cover:

- Estimating and tracking your AWS spend
- Integration with your continuous deployment process
- Deploying variations (for A/B tests or Blue/Green deployments)
- Analyzing and archiving web access logs
- Securing administrative access to your AWS resources
- Auditing administrative actions performed in your AWS account

Estimating and Tracking AWS Spend

With AWS, there is no upper limit to the amount of Amazon S3 storage or network bandwidth you can consume. You pay as you go and only pay for actual usage. Because you're not using web servers in this architecture, there are no licensing costs or concern for server scalability or utilization.

Estimating AWS Spend

To estimate your monthly costs, you can use the AWS Simple Monthly Calculator.[20] Pricing sheets for Amazon Route 53, Amazon S3, and Amazon CloudFront are all available online, with costs clearly itemized.[21] The following table provides a breakdown of the key cost components for a website served out of Amazon CloudFront and an Amazon S3 bucket in the US West Oregon Region (us-west-2):

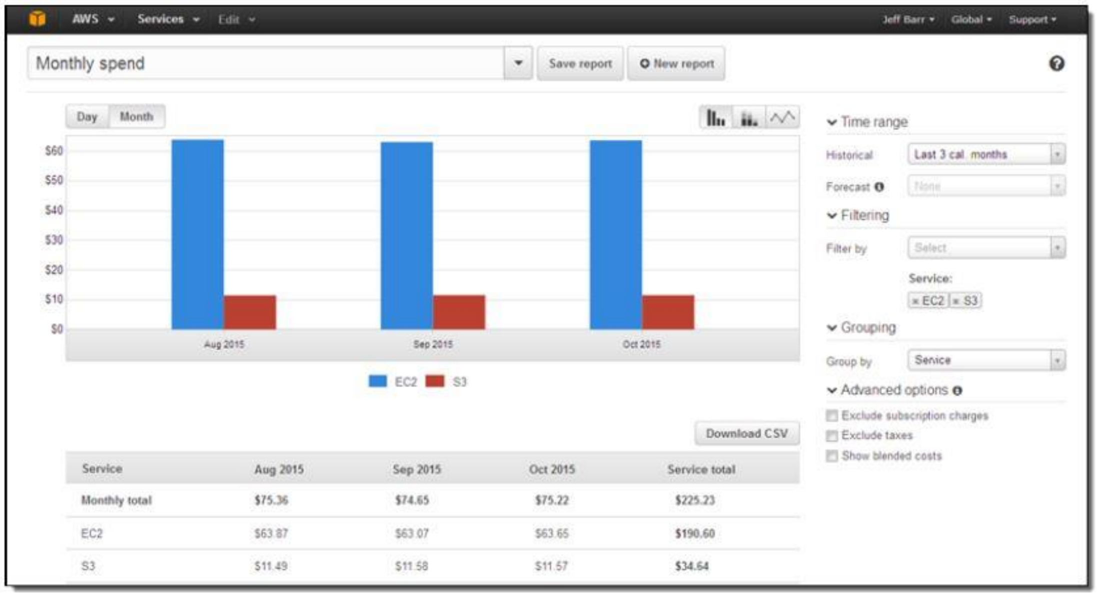
Component cost	Monthly cost	Notes
Amazon Route 53 hosted zone	$0.50/month	
DNS queries to the Amazon Route 53 hosted zone	$0.400 per million queries for the first 1 Billion queries	Most static websites will be well below 1 Billion queries / month.
Amazon S3 storage costs	$0.03 for the first TB of data stored	Most static websites will be well below a TB of storage. Therefore, unless your website contains thousands of large video files, your Amazon S3 monthly storage costs will typically be rounded up, a penny.
GET requests to Amazon S3	Every 10,000 GET is $0.004	There is cost to use CloudFront, but because it also cuts down on the number of Amazon S3 requests, it saves on S3 request costs.[22] For popular websites, using CloudFront can reduce overall AWS costs.
HTTP requests to CloudFront	Every 10,000 HTTP requests is $0.0075	Popular websites will cost more to run of course. Note that if a web page references other assets such as images, JavaScript scripts, and CSS files this will result in multiple HTTP requests to CloudFront/S3.
Data transfer from CloudFront to end users	$0.085/month for the first 10 TB	Most static websites will be well below 10 TB of outbound data transfer. Popular websites will cost more to run of course.
Transferring data from Amazon S3 to CloudFront	No charge	
API costs to deploy the website	Negligible	

** As of January 2016 pricing. Also keep in mind that AWS pricing is region-specific.

Plug your own assumptions into the AWS Simple Monthly Calculator. You might be pleasantly surprised. Most static websites will cost less than *a dollar a month*.

Tracking AWS Spend

It is a good idea to track your AWS spend on a regular basis. The AWS Cost Explorer is a helpful tool to help you track cost trends by service type.[23] It's integrated in the AWS Billing console and runs in your browser. The Monthly Spend by Service chart allows you to see where your money is going. The Daily Spend report helps you track on your spending as it happens. If you configured tags for your Amazon S3 bucket, you can filter your reports against specific tags for cost allocation purposes.[24]



Time range
Historical: Last 3 cal. months
Forecast: None
Filtering
Filter by: Select
Service: EC2, S3
Grouping
Group by: Service
Advanced options
☐ Exclude subscription charges
☐ Exclude taxes
☐ Show blended costs

Service	Aug 2015	Sep 2015	Oct 2015	Service total
Monthly total	$75.36	$74.65	$75.22	$225.23
EC2	$63.87	$63.07	$63.65	$190.60
S3	$11.49	$11.58	$11.57	$34.64

Figure 10: Sample from AWS Cost Explorer

Generally speaking, your AWS spend should be in the range of a few dollars at most, and the vast majority of static websites will not require AWS spend optimization.[25]

Are you curious how your costs changed since your last marketing blitz? Simply define a custom filter based on time and services. After you configure it just the way you want, remember to save the report for future use.[26]

Integration with Your Continuous Deployment Process

Fresh content helps to keep your website attractive. A carefully thought out deployment process will make it easy to publish fresh content. For example, earlier you learned how to use the AWS Command Line tool to upload content from your local hard drive to Amazon S3.

Your website content should be managed using version control software (such as Git, Subversion, or Microsoft Team Foundation Server) to make it possible to revert to older versions of your files.[27] AWS offers a managed source control service called AWS CodeCommit that makes it easy to host secure and private Git repositories. Regardless of which version control system your team uses, consider tying it to a continuous build/integration tool so that your website will automatically update whenever the content changes.

For example, if your team is using a Git-based repository for version control, a Git post-commit hook can notify your continuous integration tool (e.g., Jenkins) of any content updates. At that point, your continuous integration tool can perform the actual deployment to synchronize the content with S3 (using either the AWS CLI or the Amazon S3 API), and notify the user of the deployment status. Setting up continuous deployment will streamline the process for keeping content fresh, and we highly recommend it as a best practice.

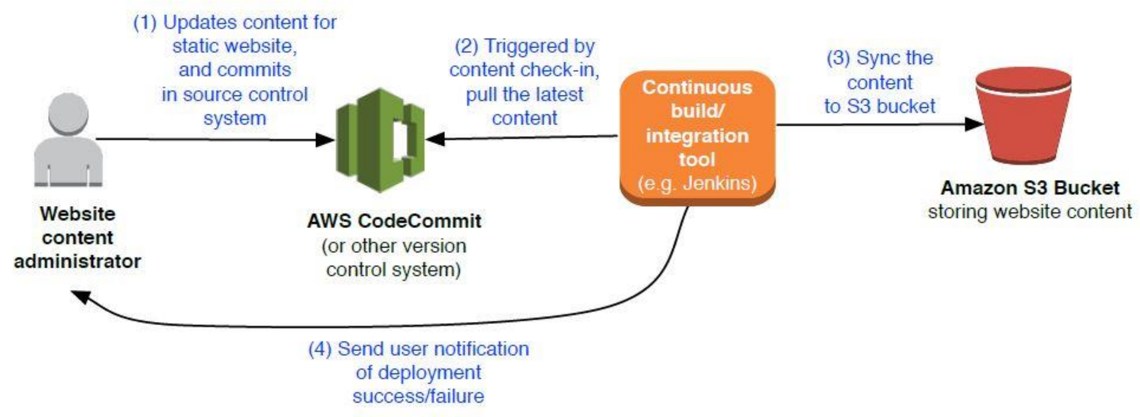

(1) Updates content for static website, and commits in source control system
(2) Triggered by content check-in, pull the latest content
(3) Sync the content to S3 bucket
Continuous build/ integration tool (e.g. Jenkins)
Website content administrator
AWS CodeCommit (or other version control system)
Amazon S3 Bucket storing website content
(4) Send user notification of deployment success/failure

Figure 11: Example continuous deployment process

If you do not want to use version control, then be sure to periodically download your website and back up that snapshot. The AWS CLI lets you download your entire website with a single command:

```
aws s3 sync s3://bucket /my_local_backup_directory
```

Deploying Variations: A/B Tests and Blue/Green Deployments

A/B Tests

Use A/B tests to test different variations of your site to see which one gives you the best results. For example, the A and B variants could display different landing pages or different order forms. Experiment with alternate designs. A/B tests are a great way to understand your users and fine-tune your website over time.

Blue/Green Deployments

A blue/green deployment serves a different purpose than an A/B test. It's a release technique that helps to manage risk and encourage experimentation. For example, say you make significant JavaScript code changes, and you're not sure if the changes will break your website. You can have a version 1 (blue) of your website, and a version 2 (green) with the JavaScript code changes. Using a blue/green deployment model, you would run two production environments side-by-side at the same time.

Initially the blue version would receive all the web traffic. Then over time, gradually shift traffic over to the green version. Start by just shifting 1 percent of the traffic over to green, then run validation tests to ensure that the changes haven't degraded the user experience. Once your team gains confidence with the changes, gradually shift more and more traffic from blue to green. Continue to monitor the health of your site and increase from 5 percent to 10 percent to 15 percent, and so on, until finally 100 percent of the traffic is routed to the green version. If you need to reverse course at any point, you can revert back to the blue version. The blue/green technique will help you feel more confident as you iterate on your site.

Deploying A/B Tests and Blue/Green Deployments

While they serve different purposes, both A/B tests and blue/green deployments involve deploying multiple variations at the same time. The approach we recommend is to create different Amazon CloudFront distributions for each variation, each with its own DNS.

Different CloudFront distributions can point to the same Amazon S3 bucket so there is no need to have multiple S3 buckets. Each variation would store its assets under different folders in the same S3 bucket. Configure the CloudFront behaviors to point to the respective Amazon S3 folders for each A/B or blue/green variation. If you want to test out multiple variations simultaneously such as A/B/C/D tests, you can just create four different CloudFront distributions pointing to the same Amazon S3 bucket.

The other key part of this strategy is an Amazon Route 53 feature called weighted routing. Weighted routing allows you to associate multiple resources with a single DNS name and dynamically resolve DNS based on their relative assigned weights. So if you want to split your traffic 70/30 for an A/B test, set the relative weights to be 70 and 30. For blue/green deployments, an automation script can call the Amazon Route 53 API to gradually shift the relative weights from blue to green after automated tests validate that the green version is healthy.

Figure 12 below illustrates sample blue/green deployments.

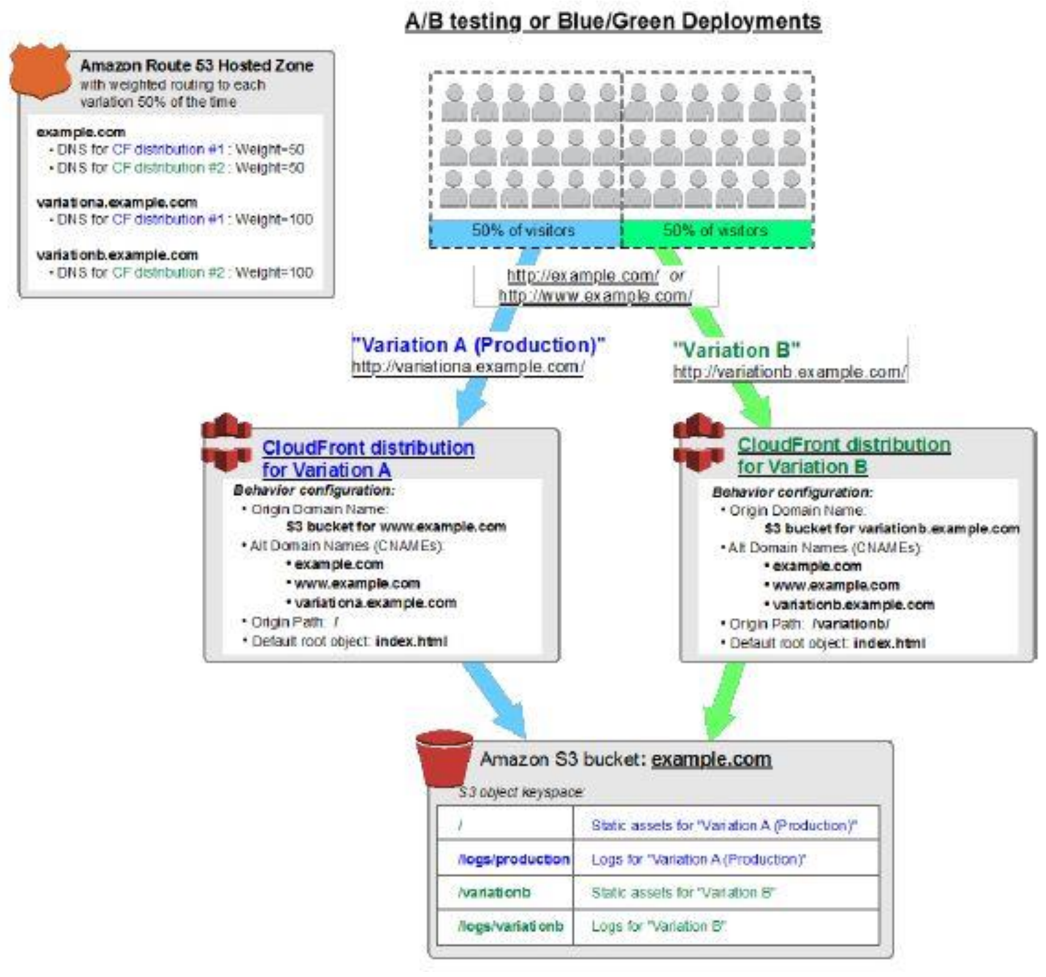

A/B testing or Blue/Green Deployments
Amazon Route 53 Hosted Zone
with weighted routing to each variation 50% of the time
example.com
• DNS for CF distribution #1 : Weight=50
• DNS for CF distribution #2 : Weight=50
variationa.example.com
• DNS for CF distribution #1 : Weight=100
variationb.example.com
• DNS for CF distribution #2 : Weight=100
50% of visitors
50% of visitors
http://example.com/ or
http://www.example.com/
"Variation A (Production)"
http://variationa.example.com/
"Variation B"
http://variationb.example.com/
CloudFront distribution for Variation A
Behavior configuration:
• Origin Domain Name:
S3 bucket for www.example.com
• Alt Domain Names (CNAMEs):
• example.com
• www.example.com
• variationa.example.com
• Origin Path: /
• Default root object: index.html
CloudFront distribution for Variation B
Behavior configuration:
• Origin Domain Name:
S3 bucket for variationb.example.com
• Alt Domain Names (CNAMEs):
• example.com
• www.example.com
• variationb.example.com
• Origin Path: /variationb/
• Default root object: index.html
Amazon S3 bucket: example.com
S3 object keyspace:
/
Static assets for "Variation A (Production)"
/logs/production
Logs for "Variation A (Production)"
/variationb
Static assets for "Variation B"
/logs/variationb
Logs for "Variation B"

Figure 12: Using multiple Amazon CloudFront distributions for Blue/Green Deployments

Access Logs

Access logs can help you troubleshoot or analyze traffic coming to your site. Both Amazon CloudFront and Amazon S3 give you the option of turning on access logs. There's no extra charge to enable logging, other than the storage of the actual logs. The access logs are delivered on a best-effort basis; they are usually delivered within a few hours after the events are recorded.

Analyzing Logs

Amazon S3 access logs are deposited in your S3 bucket as plain text files. Each record in the log files provides details about a single Amazon S3 access request, such as the requester, bucket name, request time, request action, response status, and error code, if any. You could open individual log files in a text editor.

However, it's generally far better and easier to use one of the many third-party tools that can interpret the Amazon S3 access log format.

CloudFront logs are deposited in your Amazon S3 bucket as GZIP-compressed text files. CloudFront logs follow the standard W3C extended log file format and can be analyzed using any of the myriad log analyzers that are available.

If you prefer to build out a custom analytics solution, AWS Lambda and Amazon Elasticsearch Service (Amazon ES) can help. AWS Lambda functions can be hooked to an Amazon S3 bucket to detect when new log files are available for processing.[28] AWS Lambda function code can process the log files and send the data to an Amazon ES cluster. Users can then analyze the logs by querying against Elasticsearch Amazon ES or using the Kibana visual dashboard. Both AWS Lambda and Amazon ES are managed services, and there are no servers to manage.

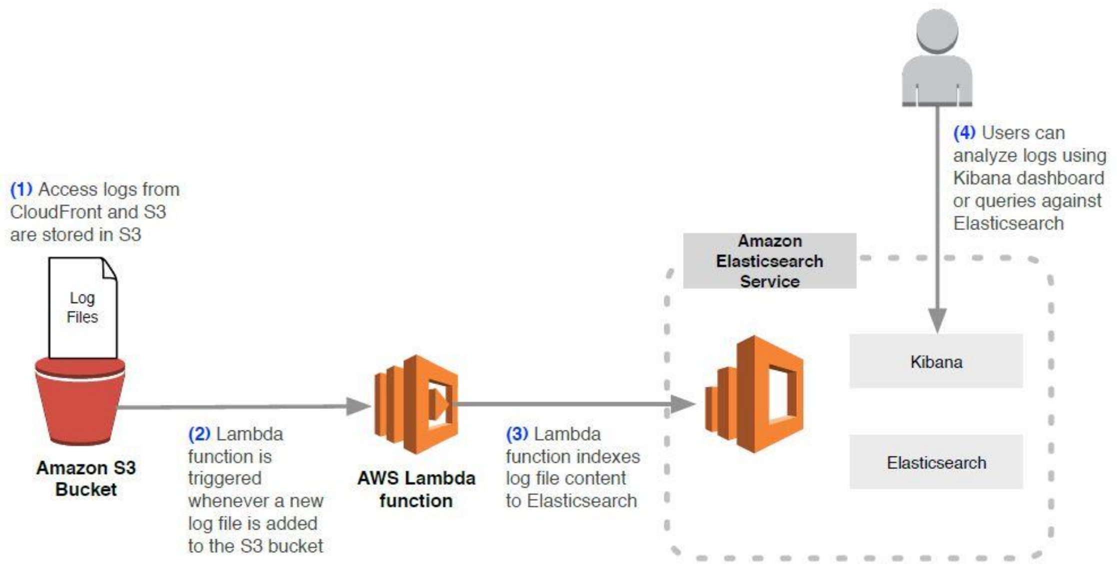

(1) Access logs from CloudFront and S3 are stored in S3
Log Files
Amazon S3 Bucket
(2) Lambda function is triggered whenever a new log file is added to the S3 bucket
AWS Lambda function
(3) Lambda function indexes log file content to Elasticsearch
Amazon Elasticsearch Service
Kibana
Elasticsearch
(4) Users can analyze logs using Kibana dashboard or queries against Elasticsearch

Figure 13: Using AWS Lambda to send logs from Amazon S3 to Amazon Elasticsearch Service

Archiving and Purging Logs

So how long should you preserve logs? Historical logs can help you track multi-year trends or perform security audits. Amazon S3 buckets don't have a storage cap, and you're free to retain logs for as long as you want. As a practical matter however, you should archive and/or purge older logs.

A best practice is to archive files into Amazon Glacier, which you can think of as a lower-cost distant cousin of Amazon S3. Amazon Glacier is suitable for long-term storage of infrequently accessed files. Like Amazon S3, Amazon Glacier is also designed for 99.999999999% data durability, and you have practically unlimited storage. The difference is in retrieval time. Amazon S3 supports immediate file retrieval. With Amazon Glacier, after you initiate a file retrieval request, there will be a 3-5 hour delay before you can start downloading the files. As long as you understand that tradeoff, Amazon Glacier is a terrific choice for archiving old logs. In the US West (Oregon) Region (us-west-2) for example, Amazon Glacier storage costs are just $0.007 per GB each month—significantly cheaper than S3, disks, or tape drives.

The easiest way to archive data into Amazon Glacier is to use Amazon S3 lifecycle policies.[29] The lifecycle policies can be applied to an entire S3 bucket or to specific objects within the bucket (e.g., only the log files). A minute of configuration in the Amazon S3 console can reduce your storage costs significantly in the long run. This is called *data tiering*. Here's an example of data tiering:

- Lifecycle policy #1: After X days, automatically move logs from Amazon S3 into Amazon Glacier.
- Lifecycle policy #2: After Y days, automatically delete logs from Amazon Glacier.

Data tiering is illustrated in Figure 14.

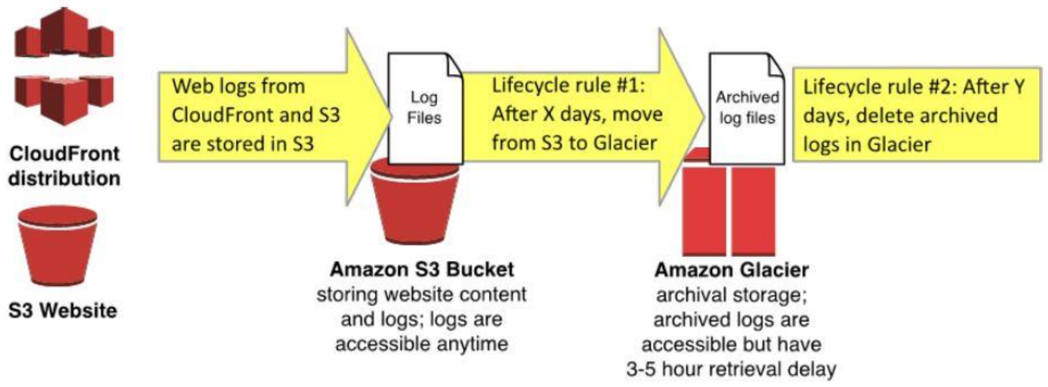

CloudFront distribution
S3 Website
Web logs from CloudFront and S3 are stored in S3
Log Files
Lifecycle rule #1: After X days, move from S3 to Glacier
Archived log files
Lifecycle rule #2: After Y days, delete archived logs in Glacier
Amazon S3 Bucket
storing website content and logs; logs are accessible anytime
Amazon Glacier
archival storage; archived logs are accessible but have 3-5 hour retrieval delay

Figure 14: Data tiering using Amazon S3 lifecycle policies

Securing Administrative Access to Your Website Resources

Under the AWS shared responsibility model, the responsibility for a secure website is shared between AWS and the customer (you). AWS provides a global secure infrastructure and foundation compute, storage, networking, and database services, as well as higher level services. AWS also provides a range of security services and features that you can use to secure your assets.

As an AWS customer, you're responsible for protecting the confidentiality, integrity, and availability of your data in the cloud, and for meeting your specific business requirements for information protection. We strongly recommend working closely with your Security and Governance teams to implement the recommendations in this whitepaper as well as the ones covered in the AWS Security Best Practices whitepaper.[30]

A benefit of using Amazon S3 and Amazon CloudFront as a serverless architecture is that the security model is also simplified. You have no operating system to harden, servers to patch, or software vulnerabilities to generate concern. Also, Amazon S3 offers security options such as server-side data encryption and access control lists.[31] This results in a significantly reduced surface area for potential attacks.

Managing Administrator Privileges

Which team members should be able to modify your Amazon S3 bucket? Who should be able to tune your Amazon CloudFront distribution or manage Amazon Route 53?

Enforcing the principle of *least privilege* is a critical part of a security and governance strategy.[32] In most organizations, the team in charge of DNS configurations is separate from the team that manages web content. You should grant users appropriate levels of permissions to access the resources they need, but no more than that. In AWS, you can use AWS Identity and Access Management (IAM) to lock down permissions.

You can create multiple IAM users under your AWS account, each with their own login and password.[33] An IAM user can be a person, service, or application that needs access to your AWS resources (in this case, Amazon S3 buckets, Amazon CloudFront distributions, and Amazon Route 53 hosted zones).[34] You can then organize them into IAM groups based on functional roles. When an IAM user is placed in an IAM group, it will inherit the group's permissions.

IAM's fine-grained policies allow you to grant administrators the minimal privileges needed to accomplish their tasks. The permissions can be scoped to specific Amazon S3 buckets and Amazon Route 53 hosted zones.

For example, the *separation of duties* might look like this:

IAM configuration can be managed by:

- Super_Admins

Amazon Route 53 configuration can be managed by:

- Super_Admins
- Network_Admins

CloudFront configuration can be managed by:

- Super_Admins
- Network_Admins
- Website_Admin

Amazon S3 configuration can be managed by:

- Super_Admins
- Website_Admin

Amazon S3 content can be updated by:

- Super_Admins
- Website_Admin
- Website_Content_Manager

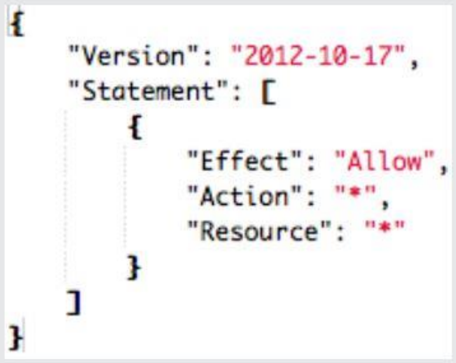
An IAM user can belong to more than one IAM group. For example, if someone needs to manage both Amazon Route 53 and Amazon S3, he or she can belong to both the Network_Admins and the Website_Admins groups.

The best practice is to require all IAM users to rotate their IAM passwords periodically. Multi-factor authentication (MFA) should be enabled for any IAM user account with administrator privileges.

Sample IAM Policies

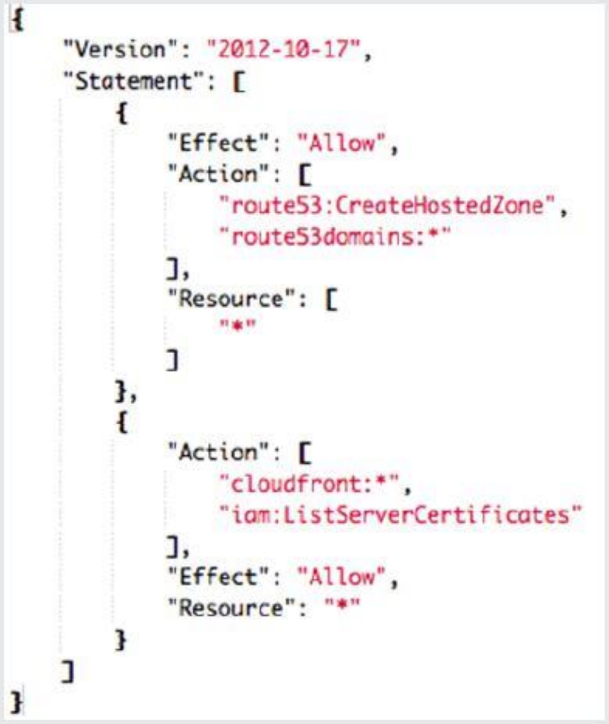
Use the IAM policy simulator tool to test out your IAM policies.[35] Here are some starting points for crafting IAM policies, and be sure to check out the IAM documentation for some great tips.[36]

Super_Admins group:



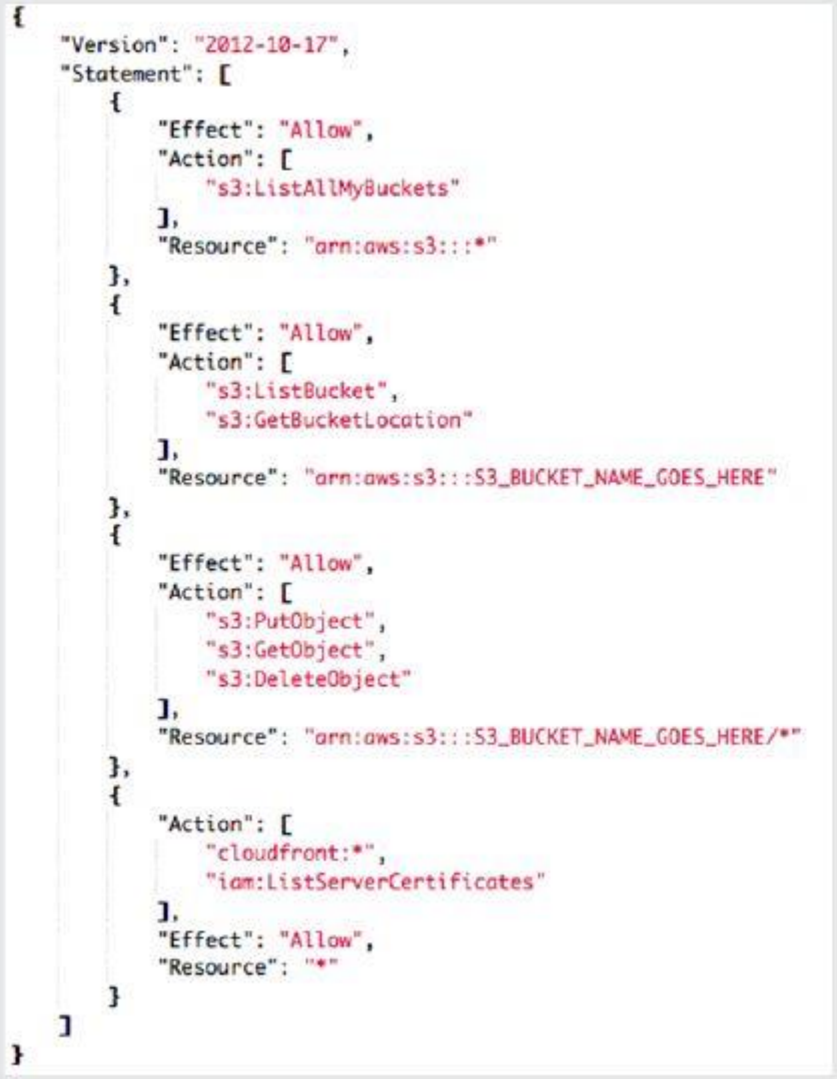
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        }
    ]
}
```

Network_Admins:



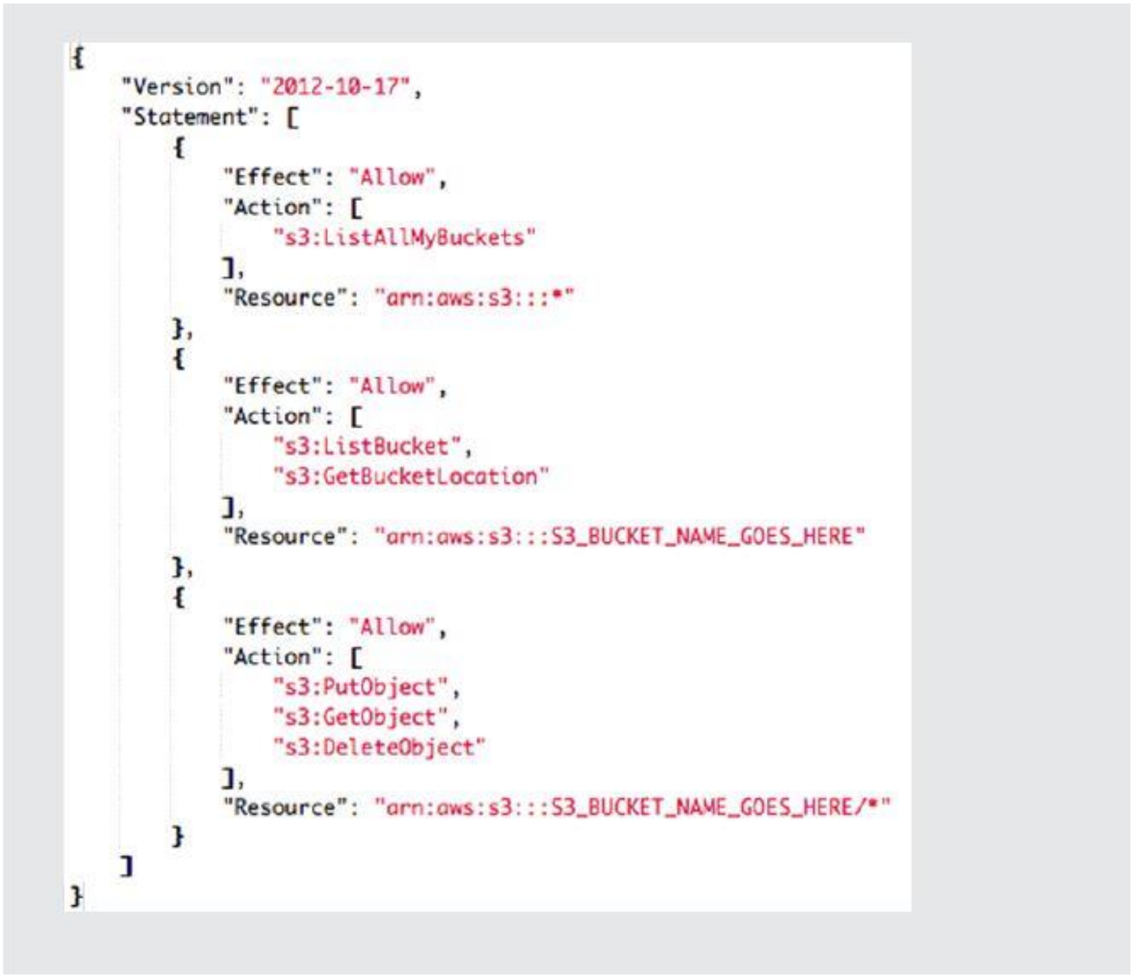
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "route53:CreateHostedZone",
                "route53domains:*"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Action": [
                "cloudfront:*",
                "iam:ListServerCertificates"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Website_Admin group:



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::S3_BUCKET_NAME_GOES_HERE"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::S3_BUCKET_NAME_GOES_HERE/*"
        },
        {
            "Action": [
                "cloudfront:*",
                "iam:ListServerCertificates"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Website_Content_Manager group:



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::S3_BUCKET_NAME_GOES_HERE"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::S3_BUCKET_NAME_GOES_HERE/*"
        }
    ]
}
```

Auditing API Calls Made in Your AWS Account

A single misconfiguration error of Amazon S3, Amazon CloudFront, or Amazon Route 53 can knock your website off line. When changes are made that have an impact on your AWS environment, it's important to be able to identify who made the change, when, and what the change was.

You can use AWS CloudTrail to see an audit trail for API activity in your AWS account. Toggle it on for all AWS Regions, and the audit logs will be deposited to an Amazon S3 bucket. You can use the AWS Management Console to search against API activity history. Or you can use a third-party log analyzer to analyze and visualize the CloudTrail logs.

Do you want to build a custom analyzer instead? You can start by configuring CloudTrail to send the data to Amazon CloudWatch Logs. CloudWatch Logs allows you to create automated rules that notify you of suspicious API activity. CloudWatch Logs also has seamless integration with the Amazon Elasticsearch Service (Amazon ES), and you can configure the data to be automatically streamed over to a managed Amazon ES cluster. Once the data is in Amazon ES, users can query against that data directly or visualize the analytics using a Kibana dashboard.

Controlling How Long Amazon S3 Content Is Cached by Amazon CloudFront

It is important to control how long your Amazon S3 content is cached at the CloudFront edge locations. This helps make sure that website updates appear correctly. If you're ever confused by a situation in which you've updated your website, but you are still seeing stale content when visiting your CloudFront powered website, one likely reason is that CloudFront is still serving up cached content. You can control CloudFront caching behavior with a combination of Cache-Control HTTP headers, CloudFront Minimum TTL specifications, Maximum TTL specifications, content versioning, and CloudFront Invalidation Requests.[37] Using these correctly will help you manage website updates.

So how long does a CloudFront edge location keep an item in its cache before expiring it? CloudFront will typically serve cached content from an edge location until the content expires. After it expires, the next time that content is requested by an end user, CloudFront will go back to the Amazon S3 origin server to fetch the content and then cache it. CloudFront edge locations automatically expire content after Maximum TTL (Maximum Time-To-Live) seconds elapse (by default, this is 24 hours). However, it could be sooner because CloudFront reserves the flexibility to expire content if it needs to, before the Maximum TTL is reached. By default, the Minimum TTL (Minimum Time-to-Live) is set to 0 seconds, and that's configurable as well. So to be more accurate, CloudFront may expire content anytime between the Minimum TTL (default is 0 seconds) and Maximum TTL (default is 24 hours). For example, if Minimum TTL=60s and Maximum TTL=600s, then content will be cached for *at least* 60 seconds and *at most* 600 seconds.

For example, say you deploy updates to your marketing website, with the latest and greatest product images. After uploading your new images to Amazon S3, you immediately browse to your website DNS, and you still see the old images! It is likely that one and possibly more CloudFront edge locations are still holding onto cached versions of the older images, and serving the cached versions up to your website visitors. If you're the patient type, you can wait for CloudFront to expire the content, but it could take up to Maximum TTL seconds for that to happen. Often a stale website is simply not acceptable.

There are several ways to tackle this, each with its pros and cons. The first approach is you can set the Maximum TTL to be a relatively low value. The tradeoff is that cached content expires faster because of the low Maximum TTL value. This results in more frequent requests to your Amazon S3 bucket because the CloudFront caches need to be repopulated more often. In addition, the Maximum TTL setting applies across the board to all CloudFront files, and for some websites you might want to control cache expiration behaviors based on file types.

The second approach is to implement content versioning. Every time you update website content, embed a version identifier in the file names. It can be a timestamp, a sequential number, or any other way that allows you to distinguish two versions of the same file. For example, instead of `banner.jpg`, call it `banner_20150401_v1.jpg`. When you update the image, name the new version `banner_20150612_v1.jpg` and update all files that need to link to the new image.[38]

In the following example, the banner and logo images are updated and receive new file names. However, because those images are referenced by the HTML files, the HTML markup also needs to be updated, to reference the new image file names. Note that the HTML file names shouldn't have version identifiers in order to provide stable URLs for end users.

Stale website updated April 4, 2015

```
├─ css/
|   ├─ main_20150401_v1.css
|   └─ navigation_20150401_v1.css
├─ images/
|   ├─ banner_20150401_v1.jpg
|   └─ logo_20150401_v1.jpg
```

```
├ index.html/
├ scripts/
│   ├ script1_20150204_v1.js
│   └ script2_20150204_v1.js
├ section1.html
└ section2.html
```

Website with images updated on June 15, 2015

```
├ css/
│   ├ main_20150401_v1.css
│   └ navigation_20150401_v1.css
├ images/
│   ├ banner_20150612_v1.jpg
│   └ logo_20150612_v1.jpg
├ index.html
├ scripts/
│   ├ script1_20150204_v1.js
│   └ script2_20150204_v1.js
├ section1.html
└ section2.html
```

Content versioning has a clear benefit: it sidesteps CloudFront expiration behaviors altogether. Since new file names are involved, CloudFront will immediately fetch the new files from Amazon S3 (and afterwards, cache them). Non-HTML website changes are reflected immediately. Additionally, it's nice to be able to roll back and roll forward between different versions of your website.

The main challenge is that content update processes will need to be "version-aware." File names will need to versioned. Files with references to changed files will also need to be updated. For example, if an image is updated, the following need to be updated as well:

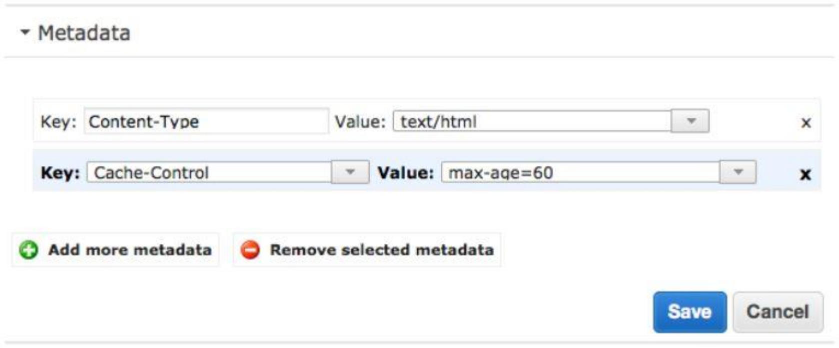
- The image file name
- Content in any HTML, CSS, and JavaScript files referencing the older image file name

- The file names of any referencing files (with the exception of HTML files)

A few static site generator tools can automatically rename file names with version identifiers, but most tools aren't version-aware. Manually managing version identifiers can be cumbersome and error-prone. If your website would benefit from content versioning, it might be worth investing in a few automation scripts to streamline your content update process.

A third approach to managing CloudFront expiration behavior is to specify `Cache-Control` headers for your website content.[39] Remember how CloudFront reserves the flexibility to expire content anytime between the Minimum TTL and Maximum TTL seconds? There's a way you can override that behavior. If you keep the Minimum TTL at the default 0 seconds, then CloudFront will honor any `Cache-Control: max-age` HTTP header that is individually set for your content. So if an image is configured with a `Cache-Control: max-age=60` header, then CloudFront will expire it at the 60 second mark. This gives you more precise control over content expiration for individual files.

You can configure Amazon S3 to return a `Cache-Control` HTTP header with the value of `max-age=<seconds>` when S3 serves up the content. This setting is on a file-by-file basis, and we recommend using different values depending on the file type (HTML, CSS, JavaScript, images, etc.). Since HTML files won't have version identifiers in their file names, we recommend using smaller `max-age` values for HTML files so that CloudFront will expire the HTML files sooner than other content. You can set this by editing the Amazon S3 object metadata using the AWS Management Console.



Key: Content-Type Value: text/html
Key: Cache-Control Value: max-age=60

Figure 15: Setting Cache-Control Values in the Console

In practice, you should automate this as part of your Amazon S3 upload process. If you're using the AWS CLI, as we recommended earlier, a small tweak to your deployment scripts will do the trick:

```
aws s3 sync /path s3://yourbucket/ --recursive \
                  --cache-control max-age=60
```

Lastly, a fourth approach for managing CloudFront expiration behavior is to use CloudFront invalidation requests.[40] Invalidation requests are a way to force CloudFront to expire content. Invalidation requests aren't immediate. It takes several minutes from the time you submit one to the time that CloudFront actually expires the content. For the occasional requests, you can submit them using the AWS Management Console. Otherwise, use the AWS CLI or AWS APIs to script the invalidation. In addition, CloudFront lets you specify which content should be invalidated: you can choose to invalidate your entire Amazon S3 bucket, individual files, or just those matching a wildcard pattern. For example, to invalidate only the images directory, issue an invalidation request for the following:

```
/images/*
```

The best practice is to understand and use the four approaches in tandem: If possible, implement content versioning; it allows you to immediately review changes and gives you the most precise control over the CloudFront and Amazon S3 experience. Set the Minimum TTL to be 0 seconds and the Maximum TTL to be a relatively low value. Also, use `Cache-Control` headers for individual pieces of content. If your website is infrequently updated, then set a large value for `Cache- Control:max-age=<seconds>` and then issue CloudFront invalidation requests every time your site is updated. If the website is updated more frequently, use a relatively small value for `Cache-Control:max-age=<seconds>` and then issue CloudFront invalidation requests only if the `Cache-Control:max-age=<seconds>` settings exceeds several minutes.

Conclusion

This whitepaper started with a look at traditional (non-AWS) architectures for static websites. We then evolved the architecture into a cloud-native architecture based on Amazon S3, Amazon CloudFront, and Amazon Route 53.

The resulting architecture is highly available and scalable, secure, and provides for a responsive user experience at very low cost. By enabling and analyzing the available logs, you can you understand your visitors and how well the website is performing. Using A/B tests and blue/green deployments will help you iterate and continuously improve the website with little risk. Fewer moving parts means less maintenance is required. In addition, the architecture costs only a few dollars a month to run.

Contributors

The following individuals and organizations contributed to this document:

- Jim Tran, AWS Principal Enterprise Solutions Architect
- Bhushan Nene, Senior Manager, AWS Solutions Architecture

Notes

1 http://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html

2 Each S3 object can be 1 byte to 5 TB in file size, and there's no limit to the number of Amazon S3 objects you can store.

3 http://aws.amazon.com/s3/sla/

4 For a list of available AWS Regions, visit: http://aws.amazon.com/about-aws/global-infrastructure/. For an overview of AWS Regions and Availability Zones, visit http://aws.amazon.com/about-aws/global-infrastructure/. If your high-availability requirements require that your website must remain available even in the case of a failure of an entire AWS Region, you may wish to explore the Amazon S3 Cross-Region Replication capability to automatically replicate your S3 data to another S3 bucket in a second AWS Region. See: http://docs.aws.amazon.com/AmazonS3/latest/UG/cross-region-replication.html

5 http://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html

6 http://docs.aws.amazon.com/gettingstarted/latest/swh/website-hosting-intro.html

7 http://docs.aws.amazon.com/gettingstarted/latest/swh/getting-started-configure-route53.html

8 If you're familiar with Microsoft IIS web servers, this is equivalent to "default.html"; for Apache web servers, this is equivalent to "index.html"

9 http://docs.aws.amazon.com/AmazonS3/latest/dev/ServerLogs.html

10 http://docs.aws.amazon.com/AmazonS3/latest/dev/BucketBilling.html

11 http://docs.aws.amazon.com/AmazonS3/latest/dev/IndexDocumentSupport.html

12 http://aws.amazon.com/cli/

13 Refer to http://docs.aws.amazon.com/cli/latest/reference/s3/sync.html for more details on the AWS CLI command. For moving very large quantities of data, there are alternative methods of moving large numbers of large files into S3: https://aws.amazon.com/s3/cloud-data-migration/

14 http://docs.aws.amazon.com/AmazonS3/latest/dev/BucketBilling.html

15 More specifically, the number of bytes that are delivered to the website visitor in the HTTP responses.

16 In addition to ensuring that end-user requests are served by the closest edge location, Amazon CloudFront also keeps persistent connections with your origin servers so that those files can be fetched from the origin servers as quickly as possible. Finally, Amazon CloudFront uses additional optimizations (e.g., wider TCP initial congestion window) to provide higher performance while delivering your content to viewers.

17 http://aws.amazon.com/cloudfront/details/

18 http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/MigrateS3ToCloudFront.html

19 "What about replicating our website content to Amazon S3 buckets in the other AWS Regions, and then serving up websites from the regional S3 bucket closest to the user?" There's no way to route all requests to a single domain to one of several Amazon S3 buckets. An Amazon S3 bucket name must be the

same as the CNAME, and S3 bucket names are globally unique, so no two buckets can have the same CNAME. For use cases where it's fine to direct different sets of users to distinct websites (with different domains/subdomains), you can use two different S3 buckets and leverage the Amazon S3 Cross Region Replication capability: http://docs.aws.amazon.com/AmazonS3/latest/UG/cross-region-replication.html

[20] http://calculator.s3.amazonaws.com/

[21] Any pricing information included in this document is provided only as an estimate of usage charges for AWS services based on the prices effective at the time of this writing. Monthly charges will be based on your actual use of AWS services, and may vary from the estimates provided.

[22] https://aws.amazon.com/cloudfront/pricing/

[23] http://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/cost-explorer-what-is.html

[24] http://docs.aws.amazon.com/AmazonS3/latest/dev/BucketBilling.html

[25] For static websites that host very large numbers of large files that are infrequently accessed (such as static video content), one potential cost optimization to consider is to set up an Amazon S3 Lifecycle Rule to move content from "S3 Standard" storage class to "S3 Standard - Infrequently-Accessed" storage class in order to take advantage of a lower-priced storage tier for infrequently-accessed files. This blog post has more details: https://aws.amazon.com/blogs/aws/aws-storage-update-new-lower-cost-s3-storage-option-glacier-price-reduction/

[26] https://aws.amazon.com/blogs/aws/the-new-cost-explorer-for-aws/

[27] If version control software is not in use at your organization, one alternative approach is to look at the Amazon S3 object versioning feature for versioning your critical files. Note that object versioning introduces storage costs for each version, and requires you to programmatically manage the different versions. For more information, see http://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html.

[28] http://docs.aws.amazon.com/lambda/latest/dg/with-s3.html

[29] https://aws.amazon.com/blogs/aws/amazon-s3-lifecycle-management-update/

[30] https://aws.amazon.com/whitepapers/aws-security-best-practices/

31 http://docs.aws.amazon.com/AmazonS3/latest/dev/UsingServerSideEncryption.html

32 http://blogs.aws.amazon.com/security/post/Tx4BUZIS3E2QG2/Make-a-New-Year-s-Resolution-Adhere-to-IAM-Best-Practices

33 The AWS account is the account that you create when you first sign up for AWS. Each AWS account has root permissions to all AWS resources and services. The best practice is to enable multi-factor authentication (MFA) for your root account, and then lock away the root credentials so that no person or system uses the root credentials directly for day-to-day operations. Instead, create IAM groups and IAM users for the day-to-day operations.

34 through the AWS Management Console, command line tools, or APIs

35 http://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_testing-policies.html

36 http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

37 http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Expiration.html

38 http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/ReplacingObjects.html

39 http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Expiration.html

40 http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Invalidation.html