Better (and Cheaper!) Content Storage with Amazon S3, CloudFront, and ColdFusion

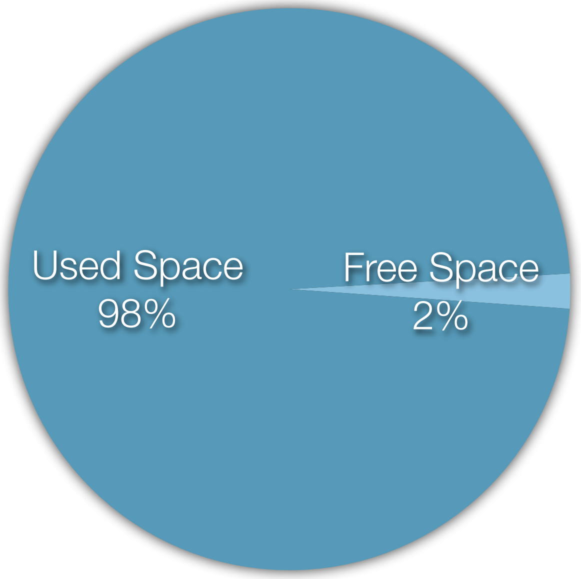
Brian Klaas
Johns Hopkins Bloomberg School of Public Health
bklaas@jhsph.edu
@brian_klaas

The Problem

I need to upload a file

How big will the files be?
How many files?
What’s the projected growth?

Is the drive big enough?



Storage is cheap.

But you've got to manage it.

Set up redundant storage

- SAN
- High–Speed I/O to servers
- Multiple nodes in the data center
- Authentication and authorization

What happens when they stop uploading Word files and start uploading PDFs? ZIP files? MP4s?

You can't delete anything.

EVER

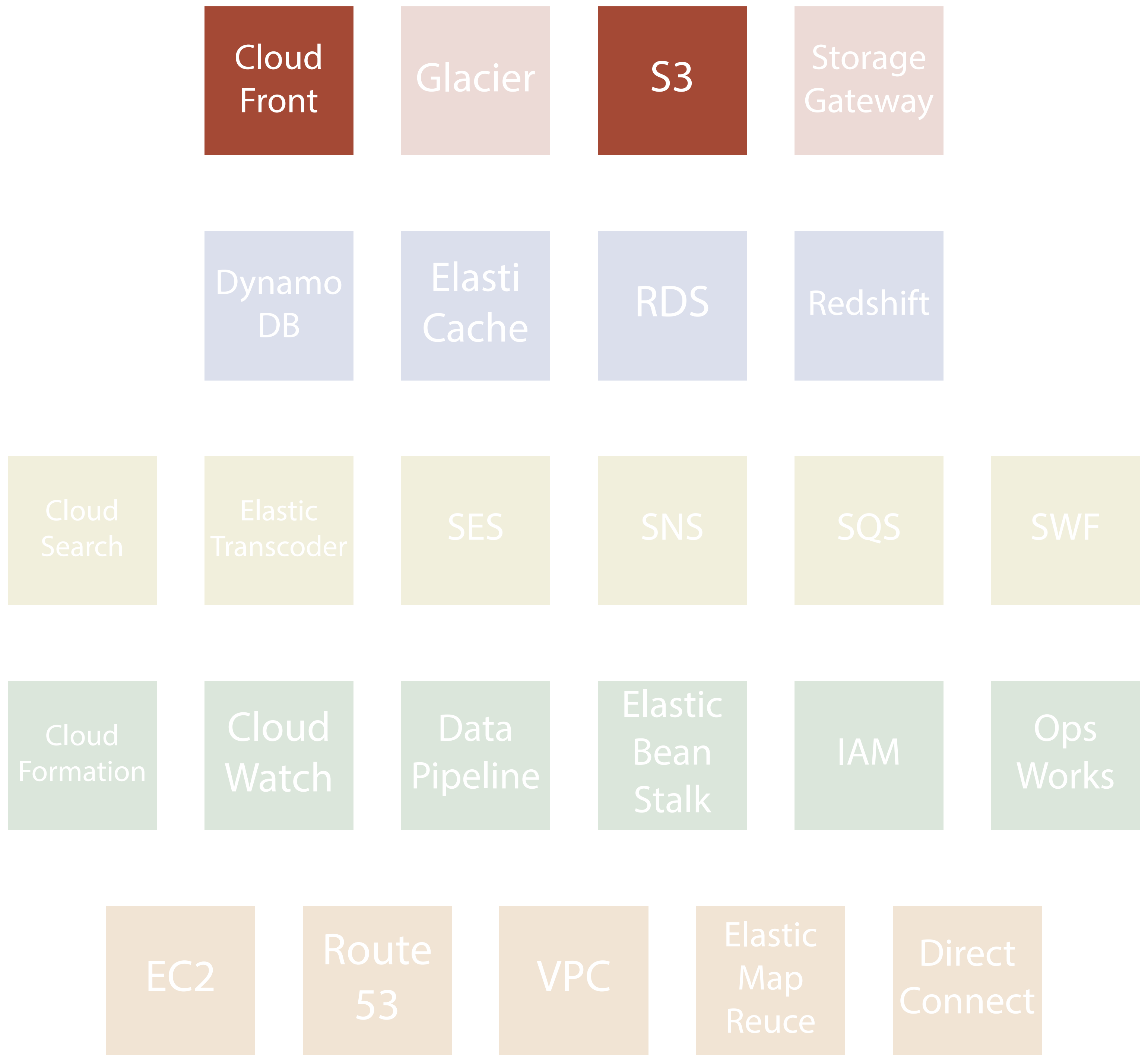
What about bandwidth costs?

What about my customers
in China?
in Australia?
in Brazil?
in Ireland?

$

A solution from AWS

AWS + S3 + CloudFront


Cloud Front
Glacier
S3
Storage Gateway
Dynamo DB
Elasti Cache
RDS
Redshift
Cloud Search
Elastic Transcoder
SES
SNS
SQS
SWF
Cloud Formation
Cloud Watch
Data Pipeline
Elastic Bean Stalk
IAM
Ops Works
EC2
Route 53
VPC
Elastic Map Reuce
Direct Connect

S3: Store all the things

$$\begin{tikzcd}\text{1 byte} \arrow[d] \\ \text{5 terabytes}\end{tikzcd}$$

Regions

- US Standard (NoVA or Washington)
- US West (Oregon)
- US West (NorCal)
- EU (Ireland)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- South America (São Paulo)

Versioning

Static website hosting

99.999999999% durability*

99.99% availability

*Yes, stuff can get lost.

$0.095 per GB

$0.01 per 10,000 GET

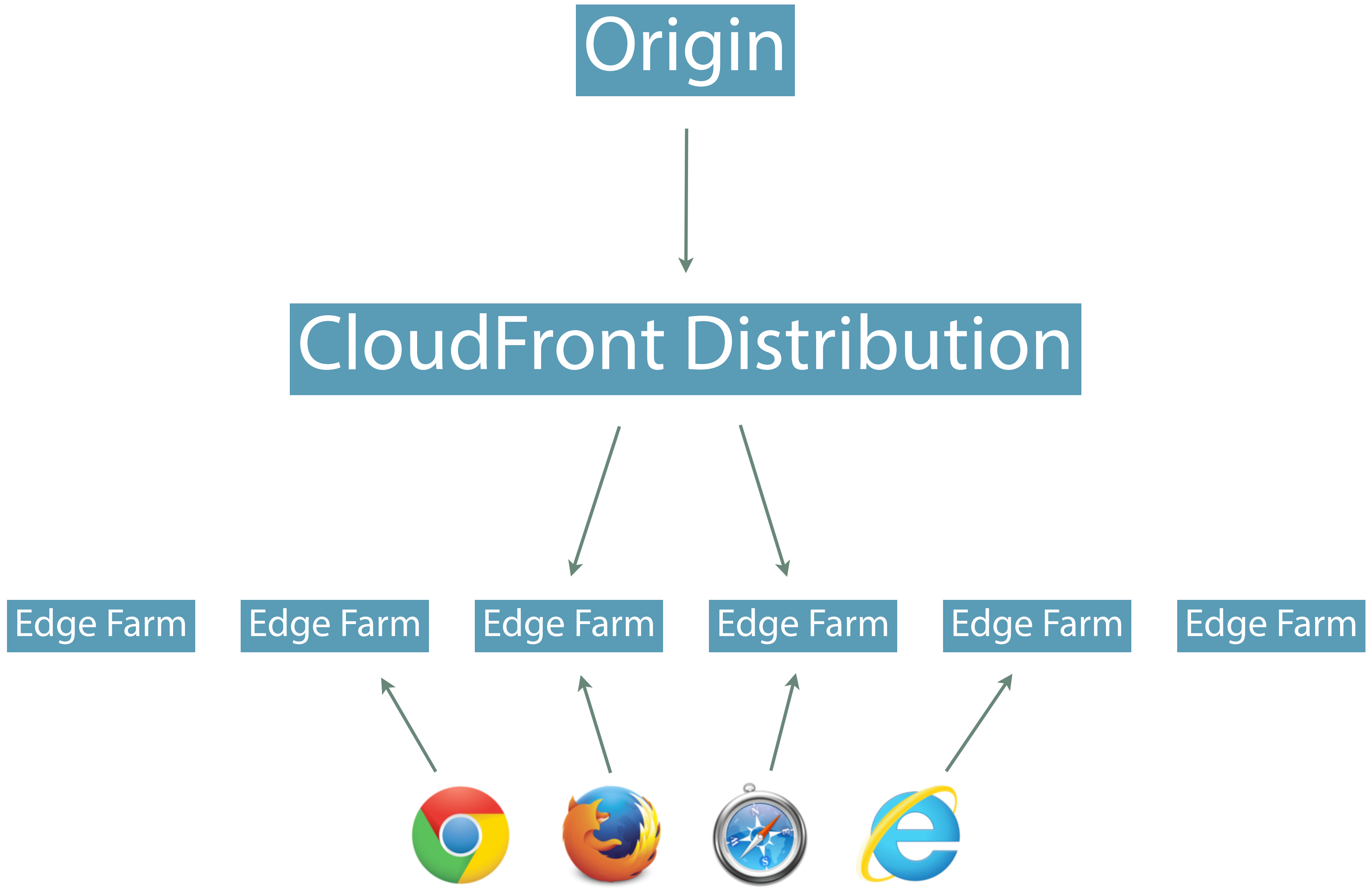
$0.01 per 1,000 PUT

$0.12 per GB out after 1GB

CloudFront: Distribute all the things

Would you prefer...?

$$\begin{tikzcd}
\text{Beijing} \arrow[d] & & \text{Beijing} \arrow[d] \\
\text{Hong Kong} \arrow[d] & & \text{Hong Kong} \\
\text{Seattle} \arrow[d] & & \\
\text{Washington, DC} \arrow[d] & & \\
\text{Raleigh} & &
\end{tikzcd}$$


Origin
CloudFront Distribution
Edge Farm
Edge Farm
Edge Farm
Edge Farm
Edge Farm
Edge Farm

Conditional GET

Download

Streaming (via FMS)

Current CloudFront Server Farms

United States

- Ashburn, VA (2)
- Dallas/Fort Worth, TX (2)
- Hayward, CA
- Jacksonville, FL
- Los Angeles, CA (2)
- Miami, FL
- New York, NY (3)
- Newark, NJ
- Palo Alto, CA
- San Jose, CA
- Seattle, WA
- South Bend, IN
- St. Louis, MO

Europe

- Amsterdam, The Netherlands (2)
- Dublin, Ireland
- Frankfurt, Germany (2)
- London, England (2)
- Madrid, Spain
- Milan, Italy
- Paris, France (2)
- Stockholm, Sweden

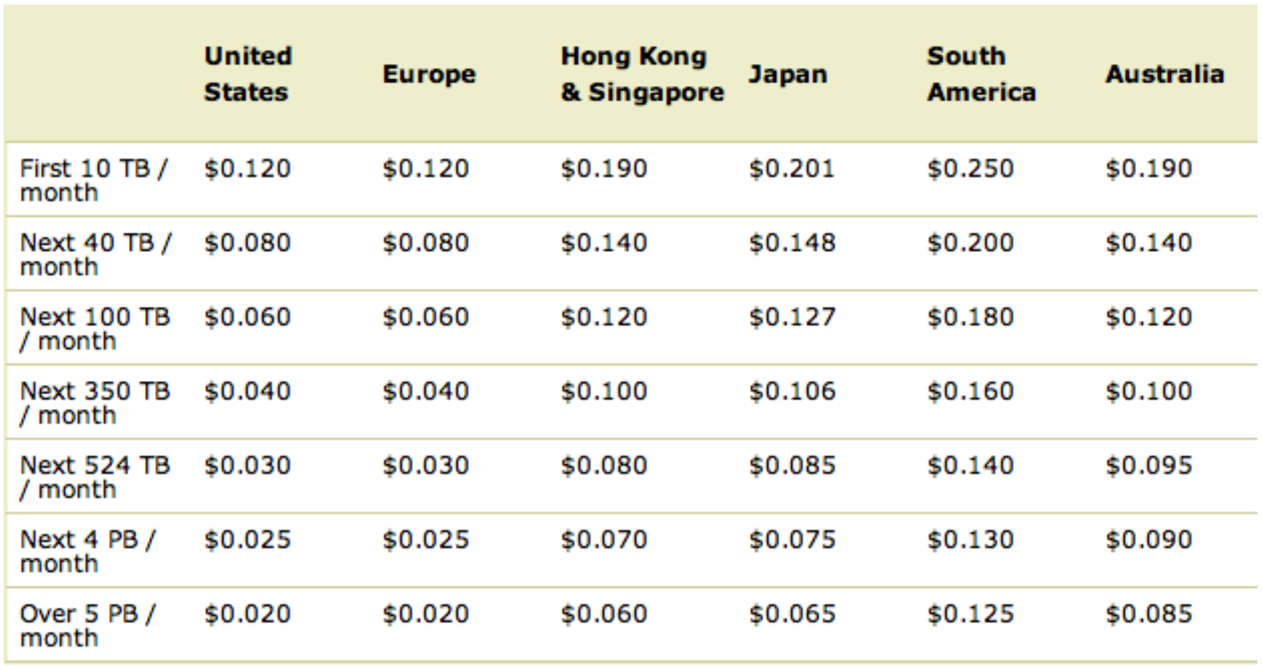
Asia

- Hong Kong, China (2)
- Osaka, Japan
- Singapore (2)
- Sydney, Australia
- Tokyo, Japan (2)

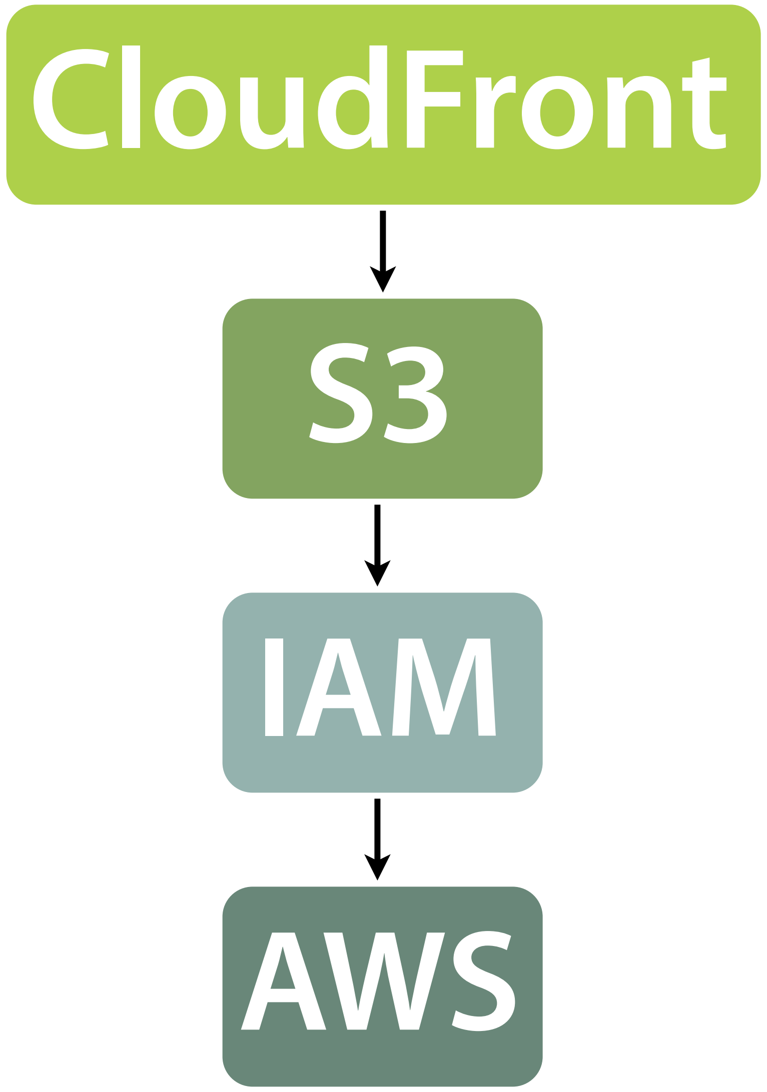
South America

- São Paulo, Brazil

CloudFront Pricing



	United States	Europe	Hong Kong & Singapore	Japan	South America	Australia
First 10 TB / month	$0.120	$0.120	$0.190	$0.201	$0.250	$0.190
Next 40 TB / month	$0.080	$0.080	$0.140	$0.148	$0.200	$0.140
Next 100 TB / month	$0.060	$0.060	$0.120	$0.127	$0.180	$0.120
Next 350 TB / month	$0.040	$0.040	$0.100	$0.106	$0.160	$0.100
Next 524 TB / month	$0.030	$0.030	$0.080	$0.085	$0.140	$0.095
Next 4 PB / month	$0.025	$0.025	$0.070	$0.075	$0.130	$0.090
Over 5 PB / month	$0.020	$0.020	$0.060	$0.065	$0.125	$0.085


CloudFront
S3
IAM
AWS

AWS is HTTP–based Development

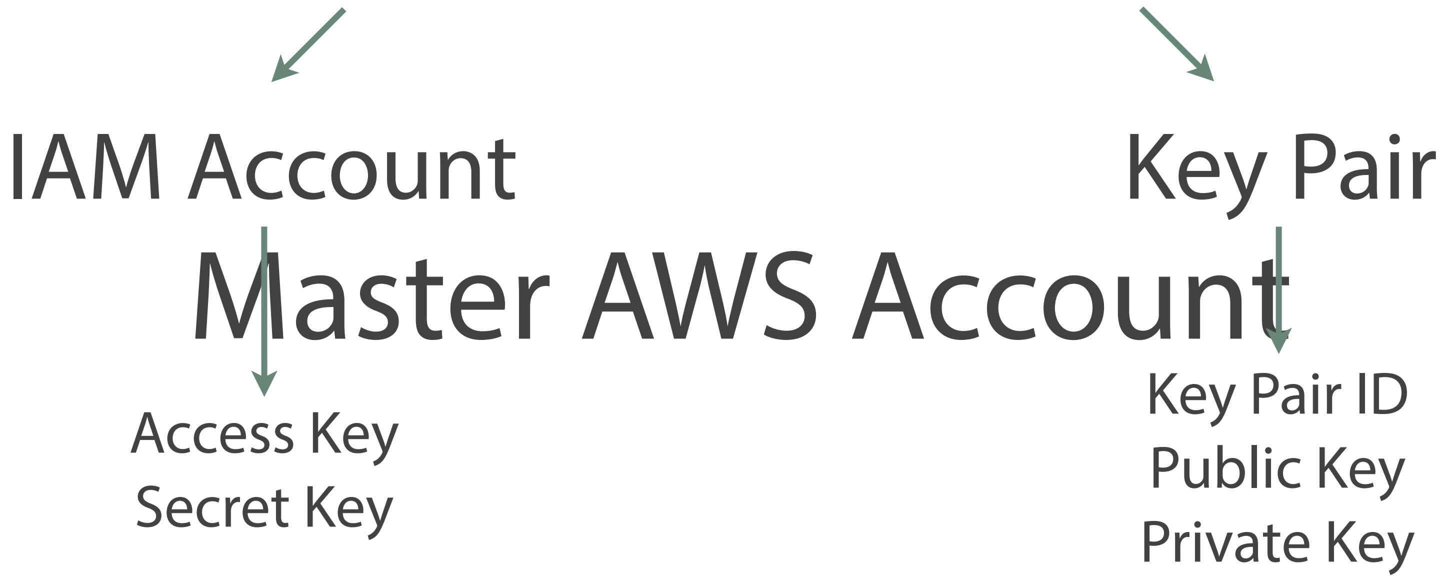
```
PUT /photos/puppy.jpg HTTP/1.1
Content-Type: image/jpeg
Content-Length: 94328
Host: johnsmith.s3.amazonaws.com
Date: Tue, 27 Mar 2007 21:15:45 +0000

Authorization: AWS AKIAIOSFODNN7EXAMPLE:
MyyxeRY7whkBe+bq8fHCL/2kKUg=
```

AWS SDKs for:

- Java*
- PHP
- Ruby
- Node.js
- Python
- .NET
- Android
- iOS

*ColdFusion

AWS Account Security

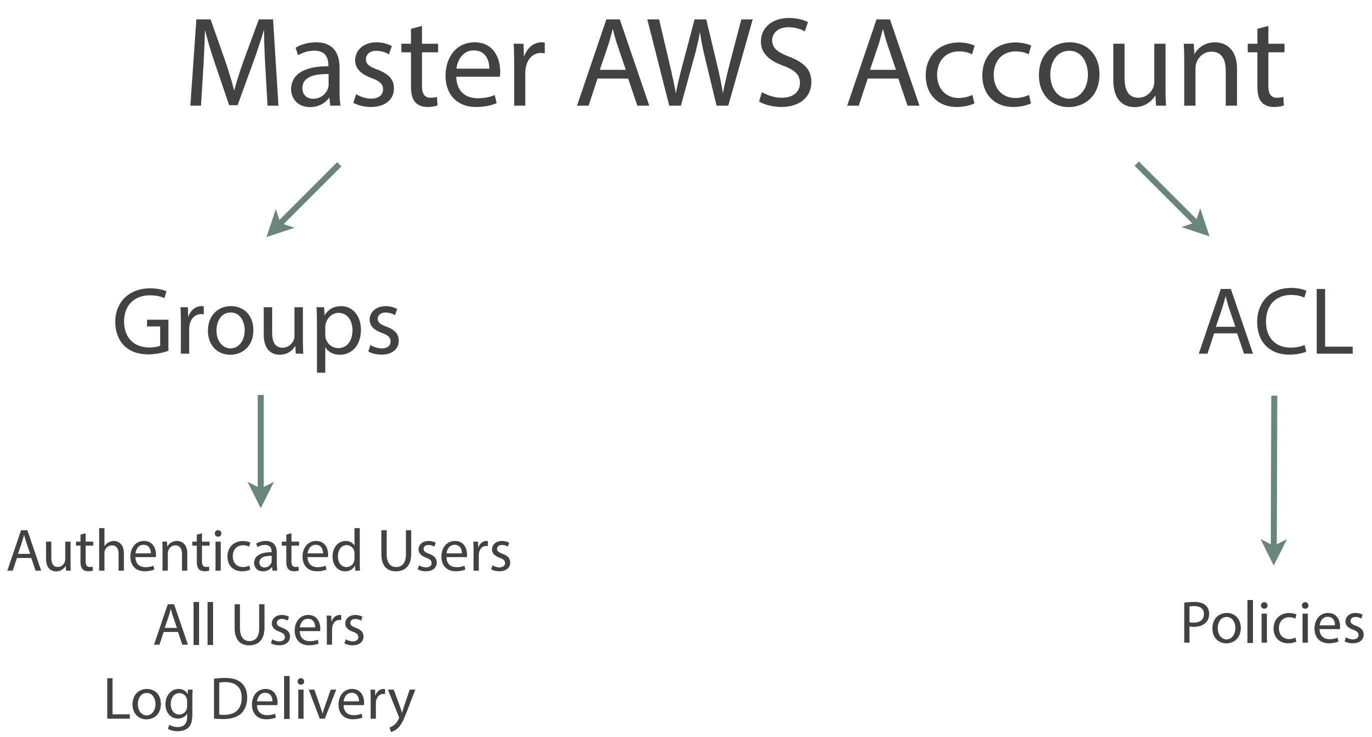

IAM Account
Key Pair
Master AWS Account
Access Key
Secret Key
Key Pair ID
Public Key
Private Key

IAM accounts can create other IAM accounts

Master AWS account can create key pairs

S3: IAM accounts

CloudFront: Key pairs


Master AWS Account
Groups
ACL
Authenticated Users
All Users
Log Delivery
Policies

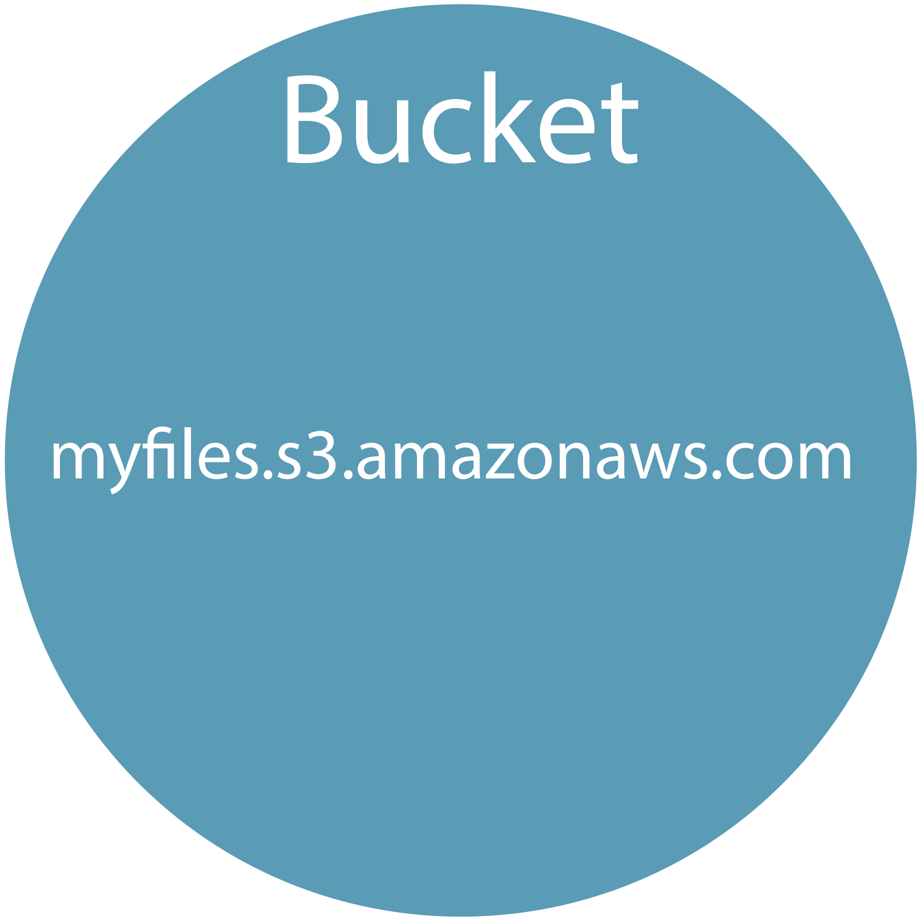
Sample Policy

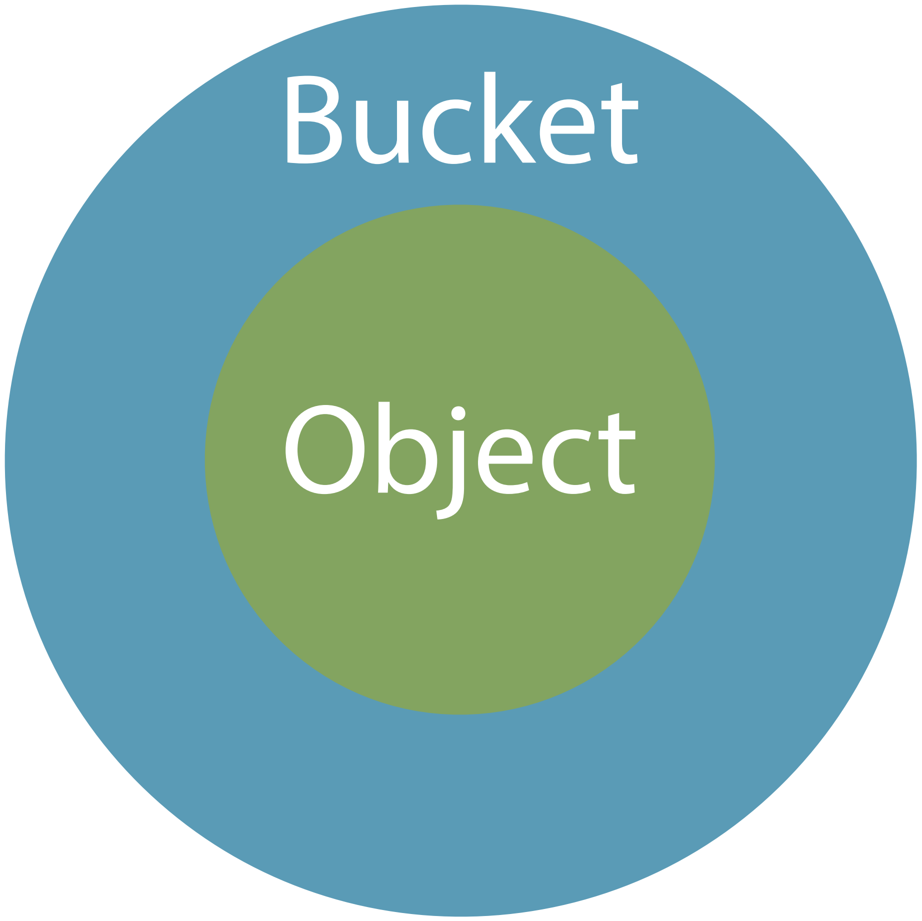
```
{
  "Version":"2008-10-17",
  "Statement":[{
   "Sid":"AddPerm",
      "Effect":"Allow",
       "Principal": {
            "AWS": "*"
         },
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::bucket/*"
      ]
    }
  ]
}
```

Requests from a Specific Domain Policy

```
{
  "Version":"2008-10-17",
  "Id":"http referer policy example",
  "Statement":[
    {
      "Sid":"Allow get requests referred by www.mysite.com 
and mysite.com",
      "Effect":"Allow",
      "Principal":"*",
      "Action":"s3:GetObject",
      "Resource":"arn:aws:s3:::example-bucket/*",
      "Condition":{
        "StringLike":{
          "aws:Referer":[
            "http://www.mysite.com/*",
            "http://mysite.com/*"
          ]
        }
      }
    }
  ]
}
```

Working with S3


Bucket
myfiles.s3.amazonaws.com


Bucket
Object

Everything is an object

Objects have metadata

Everything in S3 is private by default.

p.

```
http://mybucket.s3.amazonaws.com/
path/to/file.png
```

Sample .NET Request

```
BasicAWSCredentials basicCredentials = new
BasicAWSCredentials("*** Access Key ID ***", "***
Security Access Key ***");

AmazonS3Client s3Client = new
AmazonS3Client(basicCredentials);

var response = s3Client.ListBuckets();
```

Sample Java Request

```
AWSCredentials myCredentials = new
BasicAWSCredentials(myAccessKeyID, mySecretKey);

AmazonS3 s3client = new
AmazonS3Client(myCredentials);

List <Buckets> = s3client.listBuckets();
```

~~C:/~~

s3://

Basic ColdFusion Integration

```
<cffile action="read"
file="s3://somebucket/somefile.txt"
variable="fileData" />

<cffile action="write"
file="s3://somebucket/somefile.txt"
output="#someStuff#" />
```

Basic ColdFusion Integration

```
<cffile action="delete"
file="s3://somebucket/somefile.txt" />

<cffile action="copy"
source="s3://somebucket/somefile.txt"
destination="s3://anotherbucket/
someCopy.txt" />
```

Basic ColdFusion Integration

```
<cfdirectory action=“create”
directory=“s3://somebucket/
someDirectory” />

<cfdirectory action=“list”
directory=“s3://somebucket/
someDirectory” />
```

ColdFusion Example

```
<cfif not directoryExists("s3://somebucket.s3.amazonaws.com")>
    <cfset perms = [
        {group="all", permission="read"},
{id="canonicalIDofYourAWSAccount", permission="full_control"}
    ]>
    <cfdirectory action="create" directory="s3://
somebucket.s3.amazonaws.com" storeacl="#perms#">
</cfif>

<cfset fileWrite("s3://somebucket.s3.amazonaws.com/myFile.txt",
"#someOutput#")>

<cfset files = directoryList("s3://somebucket.s3.amazonaws.com")>
```

Tags Which Support S3

- cffile*
- cfdirectory
- cfdocument
- cfftp
- cffeed
- cfimage
- cfloop†

*Except rename
† Looping over directory information

Functions Which Support S3

- fileOpen
- fileClose
- fileCopy
- fileDelete
- fileExists
- fileisEOF
- fileMove
- fileWrite
- fileRead
- fileReadBinary
- fileReadLine
- fileSetLastModified
- getFileInfo
- getDirectoryFromPath
- directoryCreate
- directoryDelete
- directoryExists
- directoryList
- imageNew
- imageRead
- imageWrite
- imageWriteBase64
- isImageFile
- isPDFFile

Don’t you need credentials?

Setting AWS IAM credentials

1. In the individual S3 call
2. In application.cfc

Setting AWS IAM credentials

```
<cffile action="read"
file="s3://
accessKeyId:awsSecretKey@somebucket/
somefile.txt" variable="fileData" />
```

Setting AWS IAM credentials

In application.cfc:

```
this.s3.accessKeyId="accessKey";
this.s3.awsSecretKey="secretKey";
```

ColdFusion Example

```
<cfif not directoryExists("s3://somebucket.s3.amazonaws.com")>
    <cfset perms = [
        {group="all", permission="read"},
{id="canonicalIDofYourAWSAccount", permission="full_control"}
    ]>
    <cfdirectory action="create" directory="s3://
somebucket.s3.amazonaws.com" storeacl="#perms#">
</cfif>

<cfset fileWrite("s3://somebucket.s3.amazonaws.com/myFile.txt",
"#someOutput#")>

<cfset files = directoryList("s3://somebucket.s3.amazonaws.com")>
```

Everything in S3 is private by default.

ColdFusion Example

```
<cfif not directoryExists("s3://somebucket.s3.amazonaws.com")>
    <cfset perms = [
        {group="all", permission="read"},
{id="canonicalIDofYourAWSAccount", permission="full_control"}
    ]>
    <cfdirectory action="create" directory="s3://
somebucket.s3.amazonaws.com" storeacl="#perms#">
</cfif>

<cfset fileWrite("s3://somebucket.s3.amazonaws.com/myFile.txt",
"#someOutput#")>

<cfset files = directoryList("s3://somebucket.s3.amazonaws.com")>
```

Get ACL with `storeGetACL()`

Set ACL with `storeSetACL()`

Setting permissions with ACLs

```
<cfset permissions = storeGetACL(fileOnS3) />
<cfset arrayAppend(permissions,
{group="all",permission="read"}) />
<cfset storeSetACL(fileOnS3, "#permissions#") />
```

Get object metadata with `storeGetMetadata()`

Set object metadata with `storeSetMetadata()`

Setting content type

```
<cfset metadataStruct.content_type=
"video/webm" />
<cfset storeSetMetadata(s3File,
"#metadataStruct#") />
```

Standard Keys in S3 Metadata

- last_modified
- date
- owner
- etag
- content_length
- content_type
- content_encoding
- content_disposition
- content_language
- content_md5
- md5_hash

Multiparting Large Uploads

```
this.s3.minsizeformultipart=10;
```

Files above this size (in MB) will be split and sent in parallel.

*CF10–only

ACF 9.0.1/2 bug

Solution: Get your current ACL,
make the metadata change, then set the ACL again

S3: The Next Level

Custom Signing Requests

Custom Signing Requests

```
Signature = Base64(
  HMAC-SHA1( YourSecretAccessKeyID, UTF-8-Encoding-
  Of( StringToSign ) )
  );

StringToSign = HTTP-Verb + "\n" +
     Content-MD5 + "\n" +
     Content-Type + "\n" +
     Date + "\n" +
     CanonicalizedAmzHeaders +
     CanonicalizedResource;
```

Expiring the URL

Expiring the URL

1. Specify the object URL
2. Specify an expiration time for the request
3. Sign the request with your IAM secret key using HMAC encoding
4. Make the HTTP call

```
http://bucket.s3.amazonaws.com/someObject?
AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=117736
3698&Signature=vjSAMPLENmGa%2ByT272YEAiv4%3D
```

Changing file properties on a per-request basis

- File name
- Content–disposition
- MIME–type
- Adding custom metadata

Changing file properties per request

1. Specify the object URL
2. Specify your request headers
3. Sign the request with your IAM secret key using HMAC encoding
4. Make the HTTP call

S3RequestSigningUtils on GitHub

github.com/brianklaas/ctlS3Utils

*Requires CF10

Direct Upload to S3 from the Browser

www.bennadel.com/blog/2500-Uploading-Files-To-Amazon-S3-Using-A-Form-Post-And-ColdFusion.htm

Upload to S3 from the Browser

- Can upload only one file at a time.
- Must specify the policy for the upload.
- You still need to generate an authorization signature on the server and embed that as a form parameter.
- Provide a success URL in the form. AWS posts the bucket and key to that URL on successful upload.
- Failures are returned as the typical AWS error XML structure.

S3: Issues to Consider

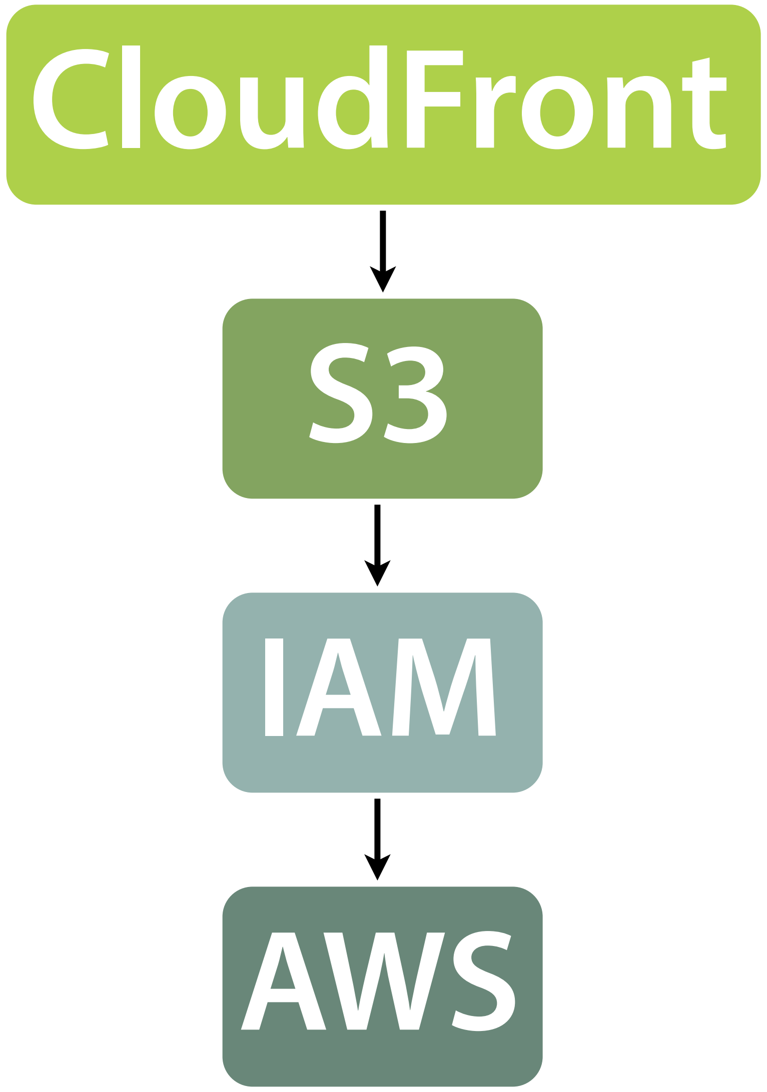
What happens when an upload fails?

S3 is storage, not a file system

Can get basic file info with
<cfhttp url="http://bucket.s3.amazonaws.com/filename" method="head">

What happens when AWS goes down?

Working with CloudFront


CloudFront
S3
IAM
AWS

CloudFront Workflow

1. Put files in your origin (S3).
2. Make the objects accessible.
3. Create a CloudFront distribution pointing at your origin.
4. Use URLs with the CloudFront distribution domain.

Use the full path to the file here, including folder name(s).

```
http://d1xrz3abcdef8.cloudfront.net/image.jpg
```

If S3 is the origin for your CloudFront distribution, the bucket name is embedded in the CloudFront distribution domain name.

Custom domain names

Default TTL = 24 hours

CloudFront: The Next Level

Changing the TTL

Versioning

?ver=2

Custom HTTP headers

?response-content-disposition=attachment;
filename=someNewFileName.mp4

Protecting For–Fee Content

CloudFront distributions which allow signed requests have additional setup parameters.

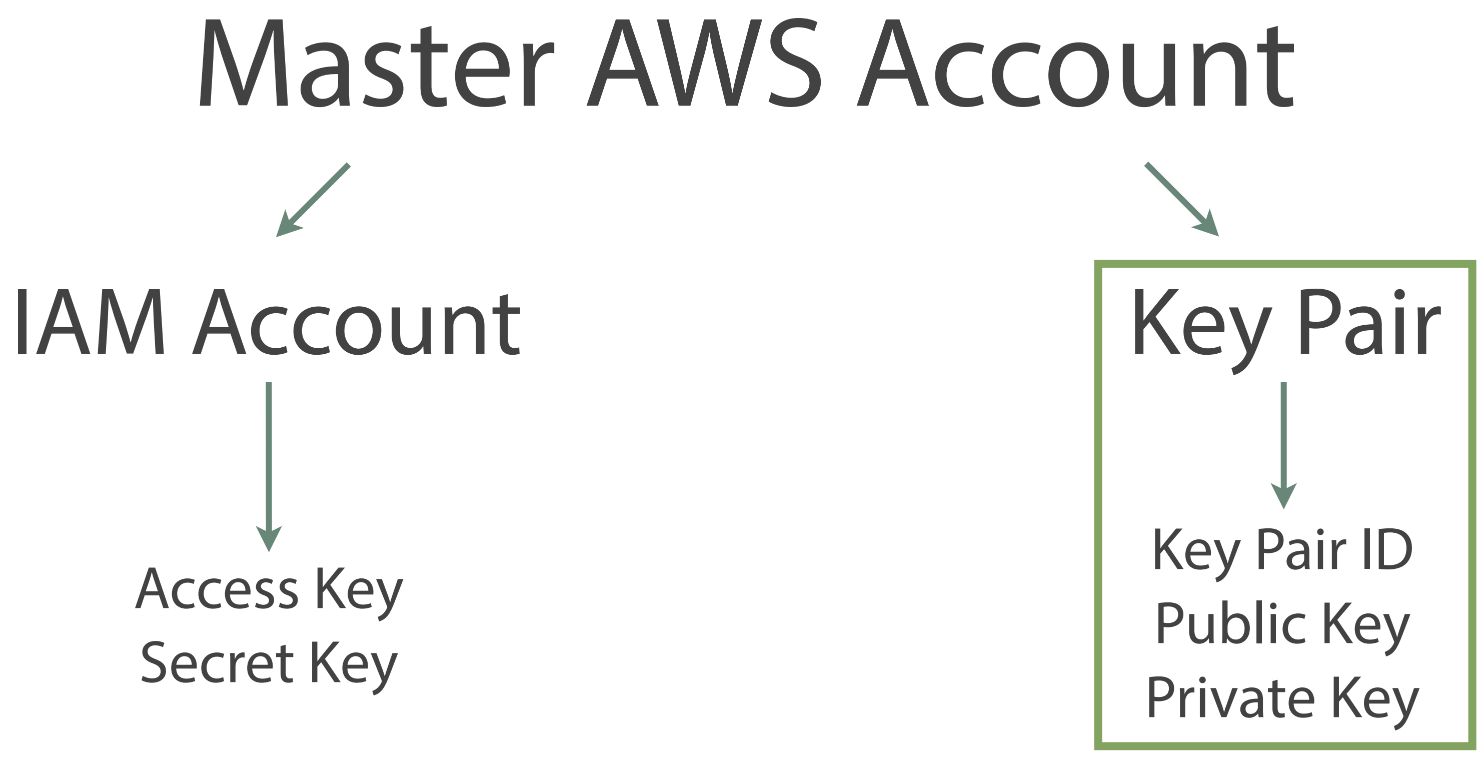
(See the docs.)

Signing the Request

1. Specify the object URL
2. Specify an expiration time for the request
3. Sign the request with the CloudFront private key from a trusted signer using HMAC encoding
4. Make the HTTP call

```
https://d1y9l76vzamjxr.cloudfront.net/sample.pdf?ver=2&
response-content-disposition=attachment%3Bfilename
%3DtotalExample.mp4
&Expires=1384467600
&Signature=DAQ5Eoy-
cKV0pqqaOxh199mHofuYEud3q4MzCHH4IKJqn25P8NJG1RbWPzkw0Dlp~PaUUl09Enq
O7BEvKnjcx-pivmDkpZ9N9-
wq44Ez09q088LaTuP31aXMSoS1~AMyR1zsvTuwkrmwrIucXPiOYiiQaidtgyFdtSRiV
uJaLaY_
&Key-Pair-Id=APKAIZXDNAH35HHVJAZQ
```

CloudFront Private Key?


Master AWS Account
IAM Account
Access Key
Secret Key
Key Pair
Key Pair ID
Public Key
Private Key

Trusted Signer?

S3: Origin Access Identity

CloudFront key pair
+ Trusted signer
+ Origin Access Identity
= Signed CloudFront URLs

Use a library for this.

(Seriously.)

CloudFront Libraries

- .NET — ThreeSharp
- Java — JetS3t
- PHP — AWS Samples
- Ruby — RightScale gem

+ regular AWS SDKs

CTL CloudFrontUtils on GitHub

github.com/brianklaas/ctlCloudFrontUtils

*Requires AWS SDK and CF10

CTL CloudFrontUtils on GitHub

```
<cfset cfUtilsObj = new ctlCloudFrontUtils(initArgs) />

<cfset argCol = structNew() />
<cfset argCol.originFilePath = "sample.pdf" />
<cfset argCol.expiresOnDate = DateAdd("n", 1, Now()) />
<cfset argCol.objectVersion = 3 />
<cfset argCol.isAttachment = true />
<cfset argCol.fileNameToUse = "lecture1a.pdf" />

<cfset signedURL =
cfUtilsObj.createSignedURL(argumentCollection =
argCol) />
```

The Full Rundown on Setting Up a CloudFront Distribution for Signed URLs and Using the CTL CloudFront Utils

www.iterateme.com/blog/index.cfm/2013/2/11/Creating-Signed-URLs-for-Amazon-CloudFront-in-ColdFusion

CloudFront: Issues to Consider

Everything fails.

Multiple distributions, balanced.

*Requires external load balancing.

Use a custom domain name.

S3 as a CDN vs. CloudFront

Go Do

Session Evaluation

ncdevcon.com/sessions/

Thank you!

Brian Klaas
Johns Hopkins Bloomberg School of Public Health
bklaas@jhsph.edu
@brian_klaas
www.iterateme.com
github.com/brianklaas

Resources Used in Building this Presentation

- AWS Management Console
  http://aws.amazon.com/console/
- AWS Documentation
  http://aws.amazon.com/documentation/
- AWS SDK's
  http://aws.amazon.com/tools/
- ColdFusion ACL Object Information
  http://help.adobe.com/en_US/ColdFusion/9.0/Developing/WSd160b5fdf5100e8f79a619d71281e7d6c97-8000.html
- Adobe CF 9.0.1/2 storeSetMetadata() Bug
  http://www.raymondcamden.com/index.cfm/2011/2/7/ColdFusion-S3-Implementation-bug-with-metadata-and-ACLs

Resources Used in Building this Presentation

- Direct Upload to S3 Using Fine Uploader
  http://blog.fineuploader.com/2013/08/16/fine-uploader-s3-upload-directly-to-amazon-s3-from-your-browser/
- Ben Nadel's Example of Direct Upload to S3 Using Plupload
  http://www.bennadel.com/blog/2502-Uploading-Files-To-Amazon-S3-Using-Plupload-And-ColdFusion.htm
- Creating a CloudFront Download Distribution
  http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/CreatingDownloadDistributions.html
- Setting up a CloudFront Distribution for Signed URLs
  http://www.iterateme.com/blog/index.cfm/2013/2/11/Creating-Signed-URLs-for-Amazon-CloudFront-in-ColdFusion

Resources Used in Building this Presentation

- CloudFront Libraries
  http://aws.amazon.com/code/CloudFront
- ColdFusion 10 Documentation on Using S3
  https://learn.adobe.com/wiki/display/coldfusionen/Optimizing
  +ColdFusion+applications