amazon web services

Masterclass

Amazon S3 – beyond simple storage

Ian Massingham – Technical Evangelist
@IanMmmm

Masterclass

A technical deep dive beyond the basics

Help educate you on how to get the best from AWS technologies

Show you how things work and how to get things done

Broaden your knowledge in ~45 minutes

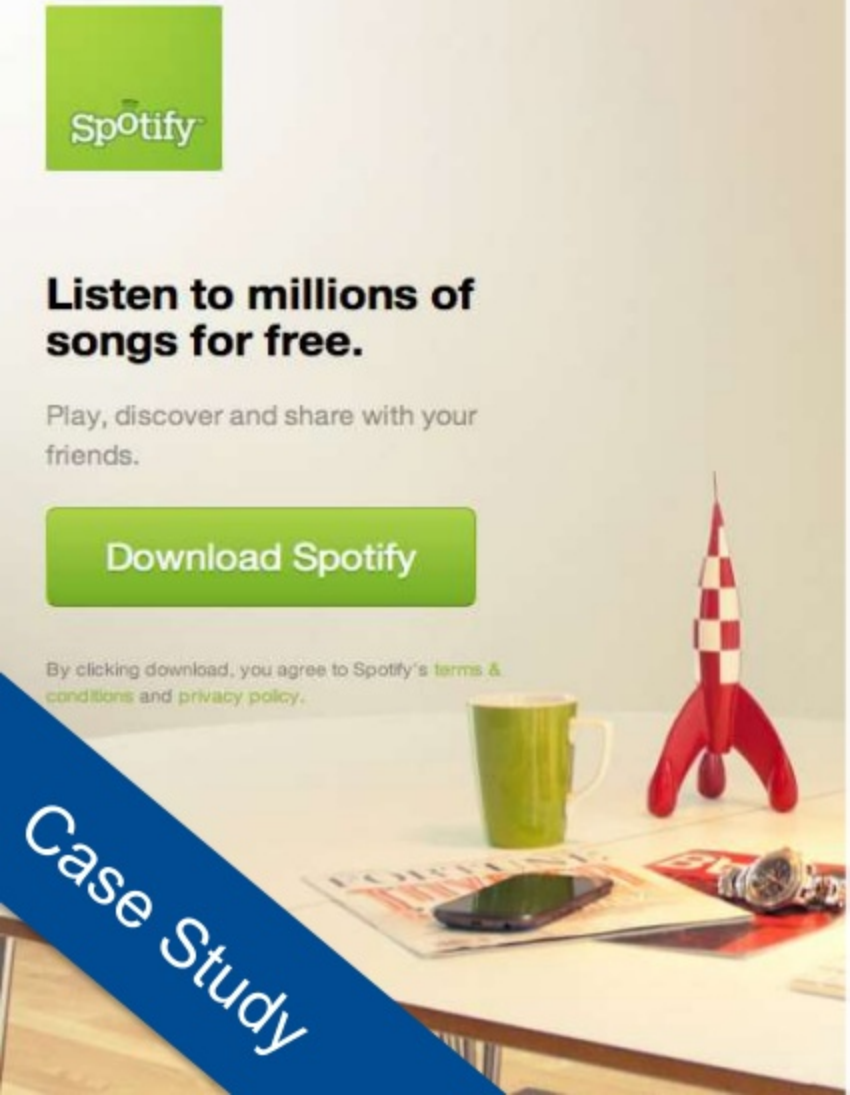
Amazon S3

It’s more than just a ‘simple’ storage platform

A sophisticated 21st century distributed system

A bedrock architectural component for many applications


Spotify
Listen to millions of songs for free.
Play, discover and share with your friends.
Download Spotify
By clicking download, you agree to Spotify's terms & conditions and privacy policy.
Case Study



"Spotify needed a storage solution that could scale very quickly without incurring long lead times for upgrades. This led us to cloud storage, and in that market, Amazon Simple Storage Service (Amazon S3) is the most mature large-scale product.

Amazon S3 gives us confidence in our ability to expand storage quickly while also providing high data durability."

Emil Fredriksson, Operations Director

Find out more at : aws.amazon.com/solutions/case-studies/spotify/

Get Minecraft!

or play the demo

Already bought the game?
Download it here

craft is a game about breaking and placing blocks. At first, people built structures to protect against
l monsters, but as the game grew players worked together to create wonderful, imaginative things.

out adventuring with friends or watching the sun rise over a blocky ocean. It's pretty. Brave
e things in The Nether, which is more scary than pretty. You can also visit a land of
more like your cup of tea.

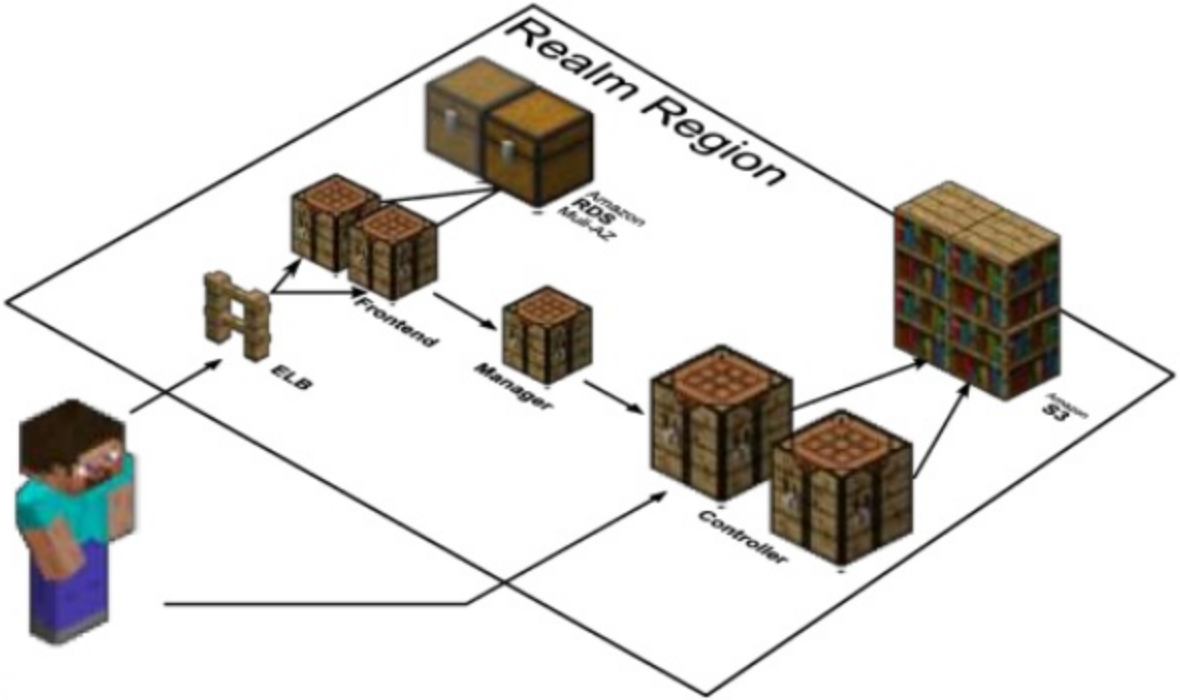
You can
ac. Minecraft: Pocket Edition is available for iOS and Android, and Minecraft:
Xbox 360 E
XBLA Marketplace. So far 14,037,668 people bought the PC/Mac version
of the game. Pr

Case Study

Case Study

A safe, simple way to enjoy Minecraft with friends

Playing Minecraft over the internet is a lot of fun, but it can be tricky to set up. To make things easier, we're creating Minecraft Realms. It's a paid service that lets users quickly set up an online world which can be accessed by an approved list of players.



Case Study

Minecraft Realms and AWS S3

Minecraft worlds and game state are stored in Amazon S3. The system takes advantage of S3 versioning, giving the owner / administrator of a realm the ability to roll back to a previous state. The team implemented efficient transfers to S3 by using S3's multipart upload capabilty.

Find out more at : aws.typepad.com/aws/2014/01/hosting-minecraft-realms-on-aws.html

You put it in S3

AWS stores with 99.999999999% durability


Highly scalable web
access to objects
You put it in S3
AWS stores with 99.999999999% durability
Multiple redundant
copies in a region

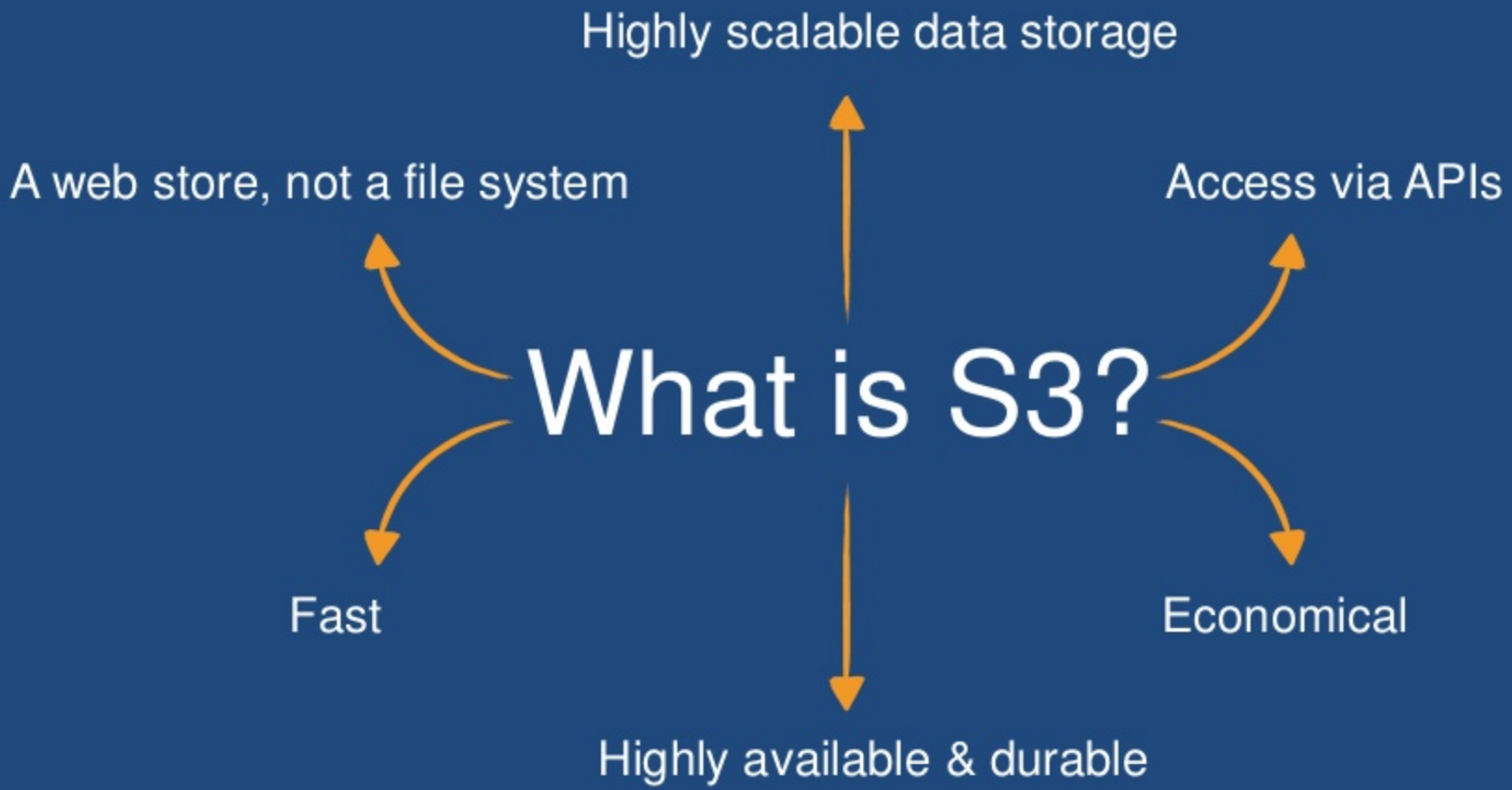
but it’s more than just a

simple storage service

Objects in S3

Trillions of Objects
(000,000,000,000s)

Servicing over 2 million
requests per Second


Highly scalable data storage
A web store, not a file system
Access via APIs
What is S3?
Fast
Economical
Highly available & durable

A web store, not a file system

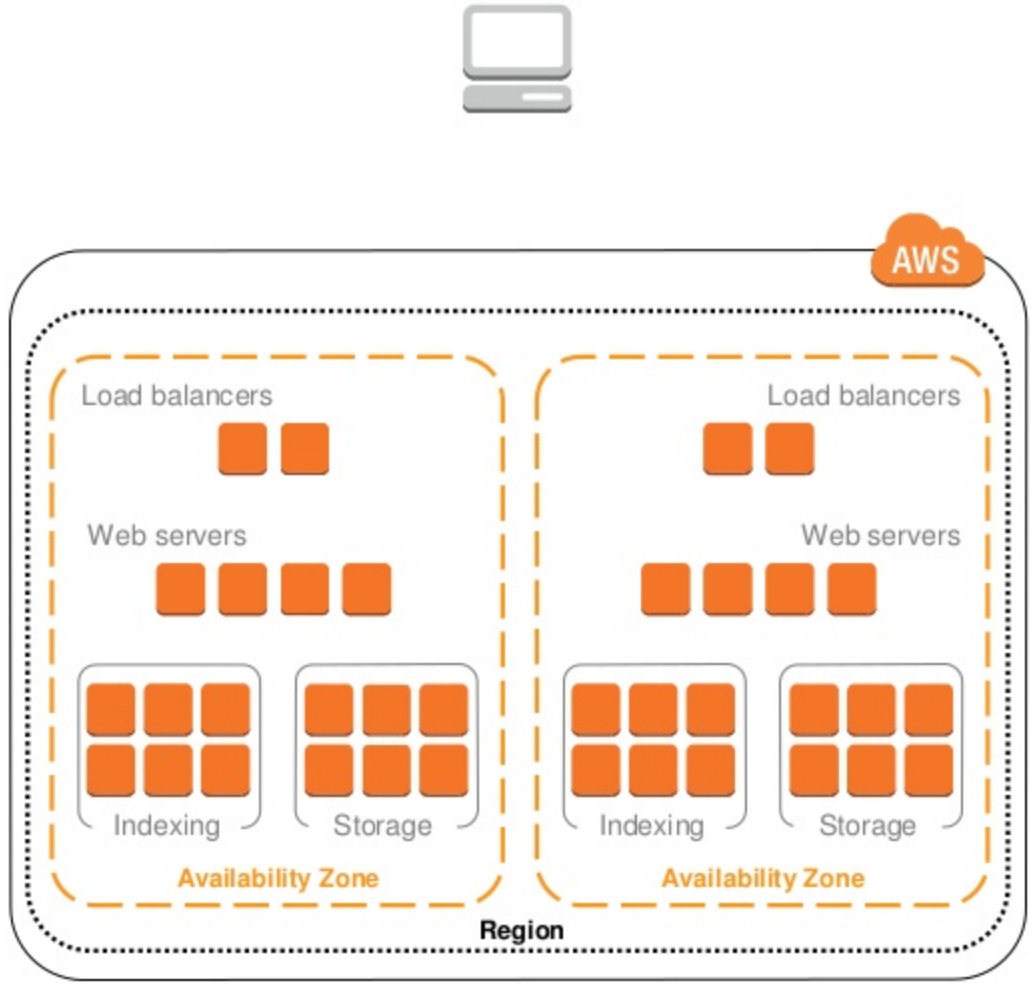
Write once,
read many
(WORM)

Eventually
consistent

A web store, not a file system

Write once, read many (WORM)

Eventually consistent

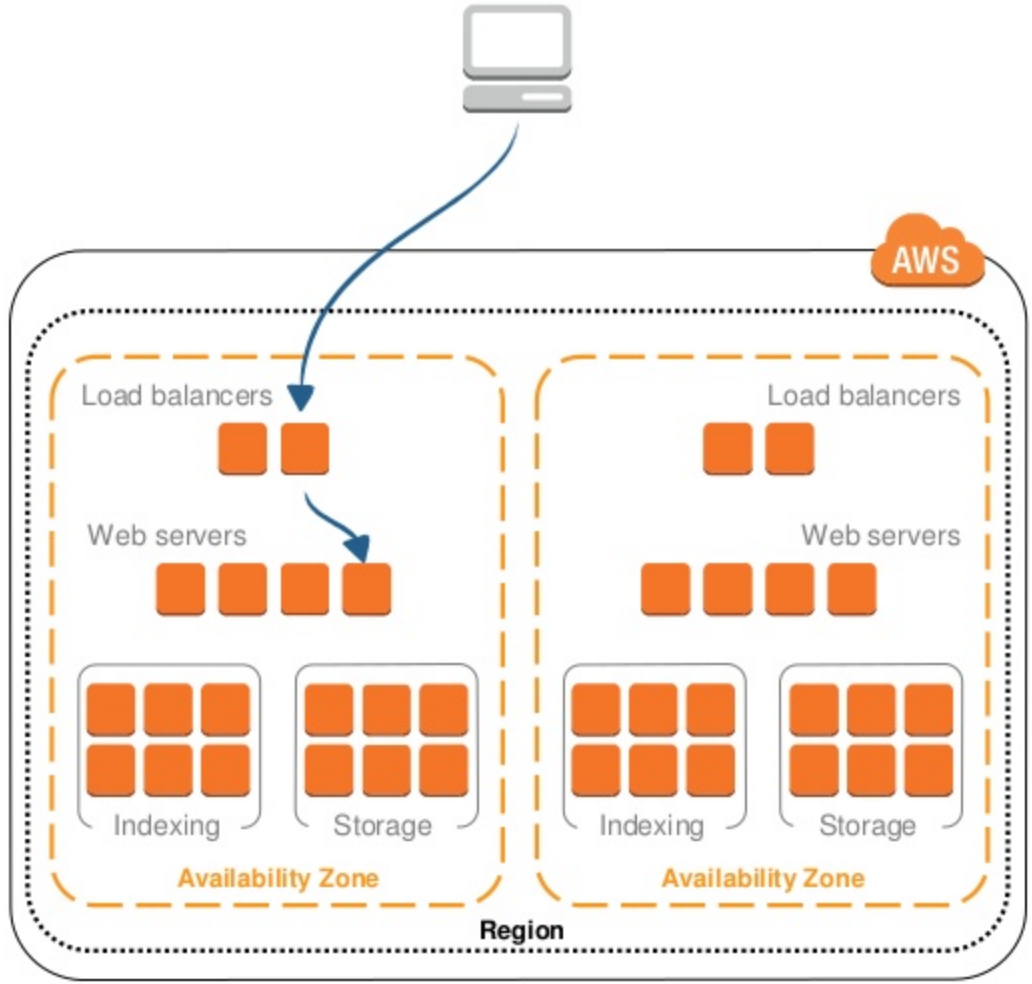


Conceptual diagram only

A web store, not a file system

Write once, read many (WORM)

Eventually consistent

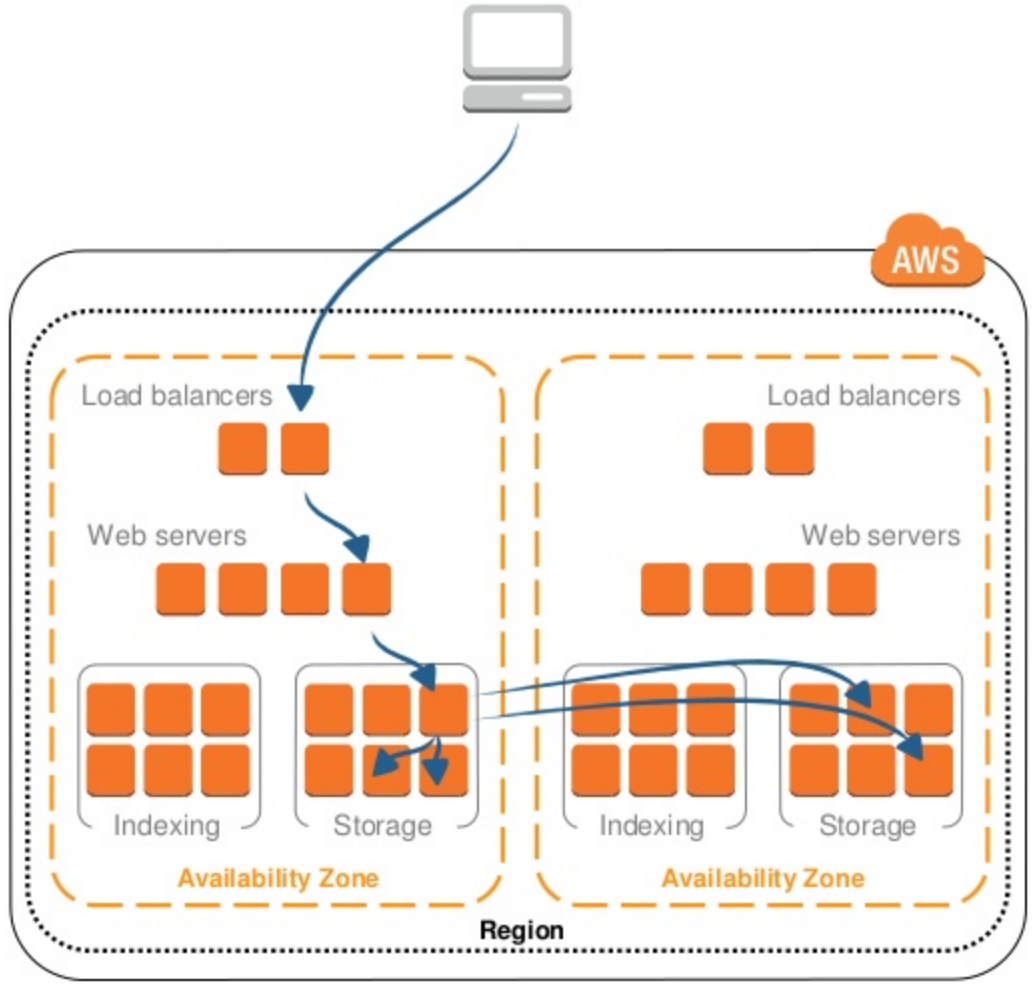


Conceptual diagram only

A web store, not a file system

Write once, read many (WORM)

Eventually consistent

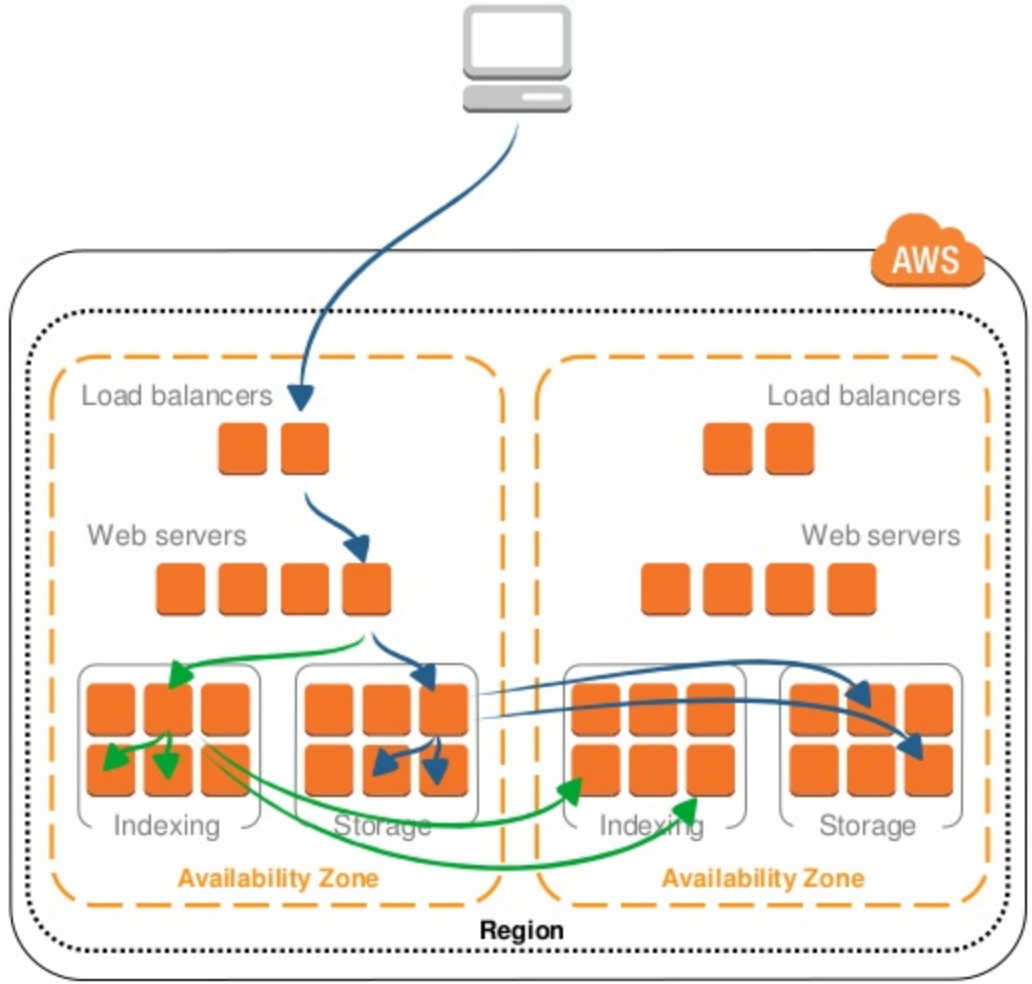


Conceptual diagram only

A web store, not a file system

Write once, read many (WORM)

Eventually consistent

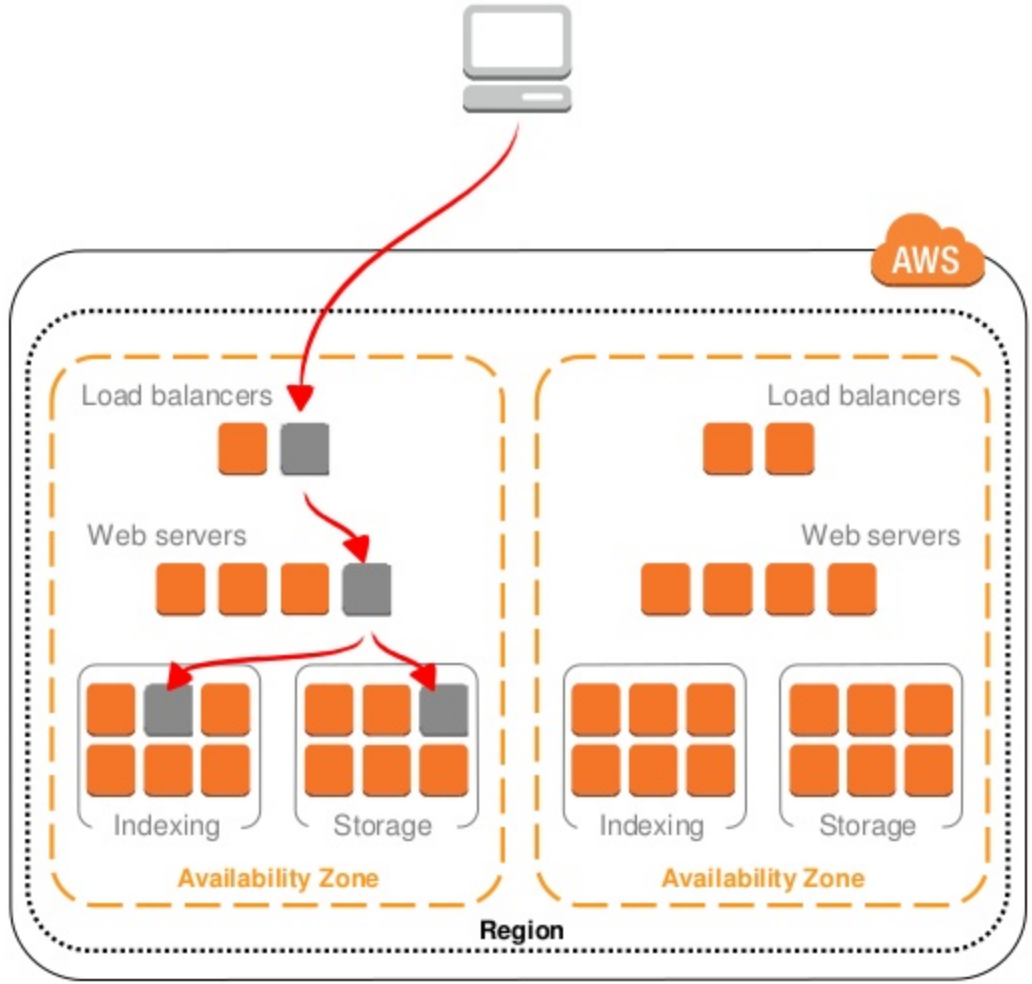


Conceptual diagram only

A web store, not a file system

Write once, read many (WORM)

Eventually consistent

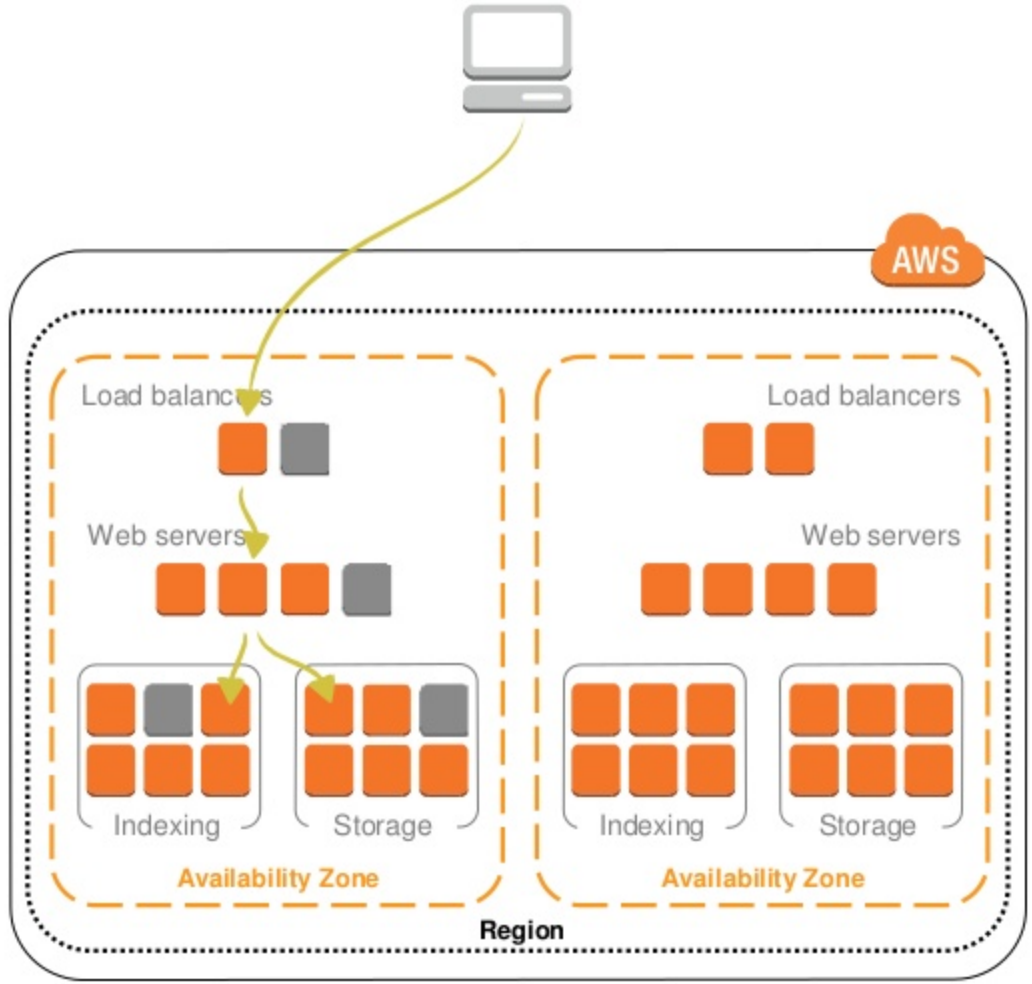


Conceptual diagram only

A web store, not a file system

Write once, read many (WORM)

Eventually consistent

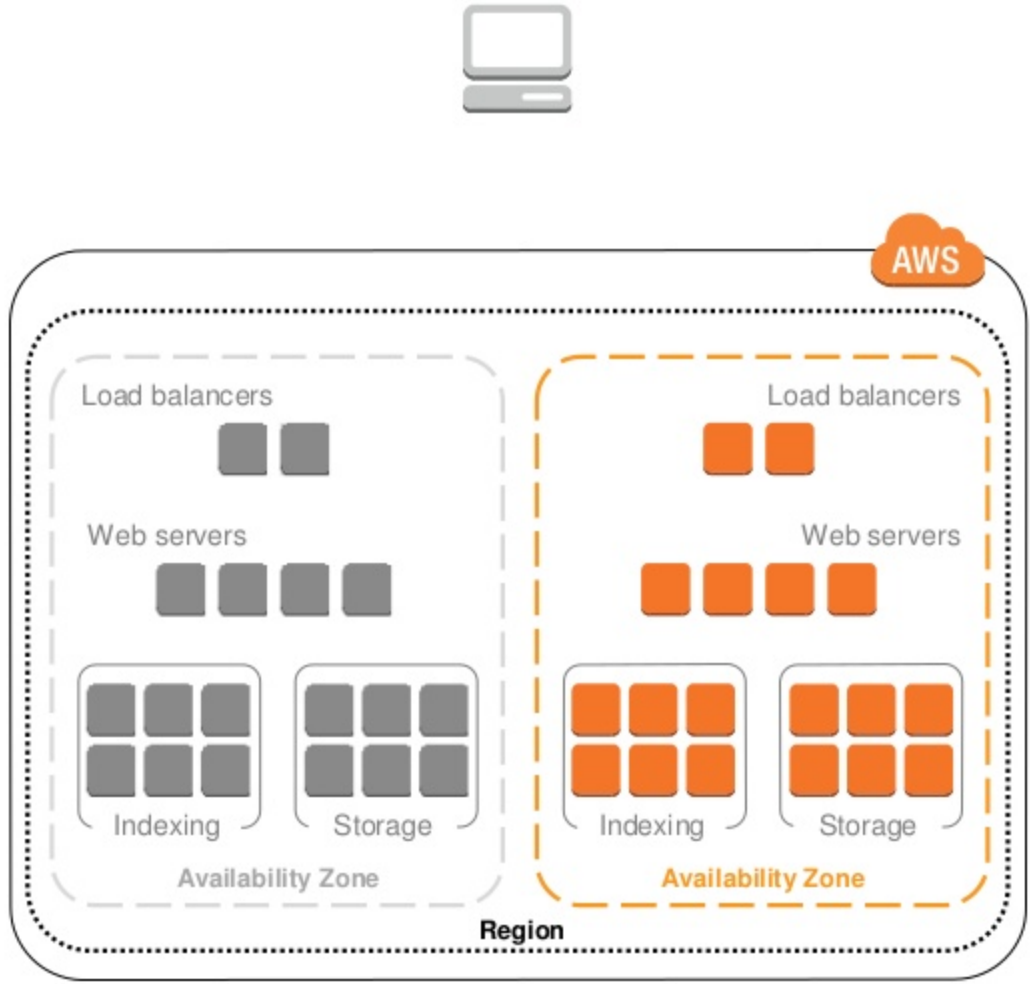


Conceptual diagram only

A web store, not a file system

Write once, read many (WORM)

Eventually consistent

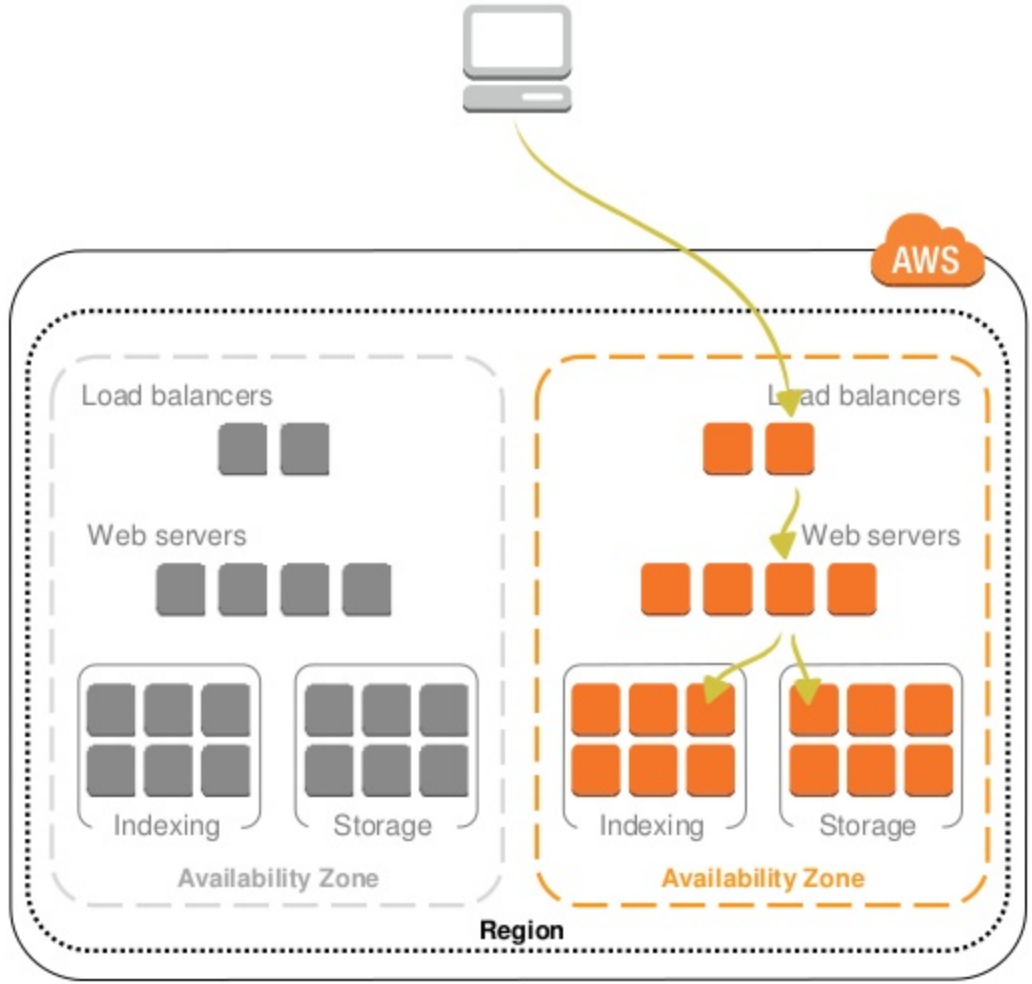


Conceptual diagram only

A web store, not a file system

Write once, read many (WORM)

Eventually consistent



Conceptual diagram only

A web store, not a file system

Conceptual diagram only

A web store, not a file system

New objects

Synchronously stores your data across multiple facilities before returning SUCCESS

*Read-after-write consistency**

Updates

Write then read: could report key does not exist

Write then list: might not include key in list

Overwrite then read: old data could be returned

Deletes

Delete then read: could still get old data

Delete then list: deleted key could be included in list

*except US-STANDARD region

A regional service

Your data never leaves a region unless you move it

Storage classes

Controlling the way S3 holds your data

Amazon S3 storage classes

Standard

Designed to provide 99.999999999% durability and 99.99% availability of objects over a given year

Designed to sustain the concurrent loss of data in two facilities

Amazon S3 storage classes

Standard

Designed to provide 99.999999999% durability and 99.99% availability of objects over a given year

Designed to sustain the concurrent loss of data in two facilities

Reduced Redundancy Storage

Reduces costs by storing data at lower levels of redundancy than the Standard storage

Designed to provide 99.99% durability and 99.99% availability of objects over a given year

Amazon S3 storage classes

Standard

Designed to provide 99.999999999% durability and 99.99% availability of objects over a given year

Designed to sustain the concurrent loss of data in two facilities

Reduced Redundancy Storage

Reduces costs by storing data at lower levels of redundancy than the Standard storage

Designed to provide 99.99% durability and 99.99% availability of objects over a given year

Glacier

Suitable for archiving data, where data access is infrequent and a retrieval time of several hours is acceptable

Uses the very low-cost Amazon Glacier storage service, but managed through Amazon S3

Amazon S3 storage classes

Standard

Objects you want to have high durability

e.g. master copy of movie media

Reduced Redundancy

Objects you can afford to lose or can recreate

e.g. different encodings of movie media

Glacier

Objects you want to keep in archive for a long time

e.g. digital archive of old movies & broadcasts

Namespaces

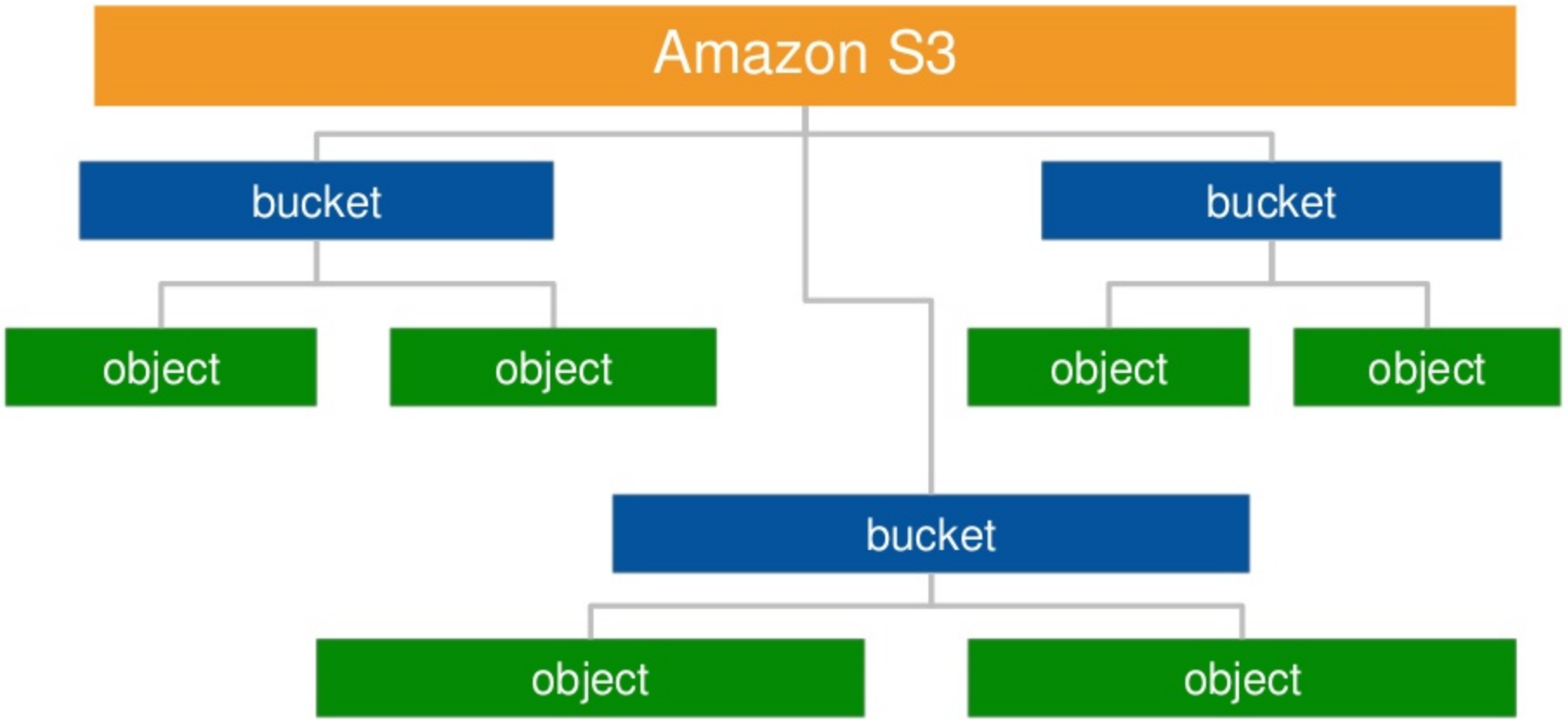
Object naming, buckets and keys

Amazon S3 namespace

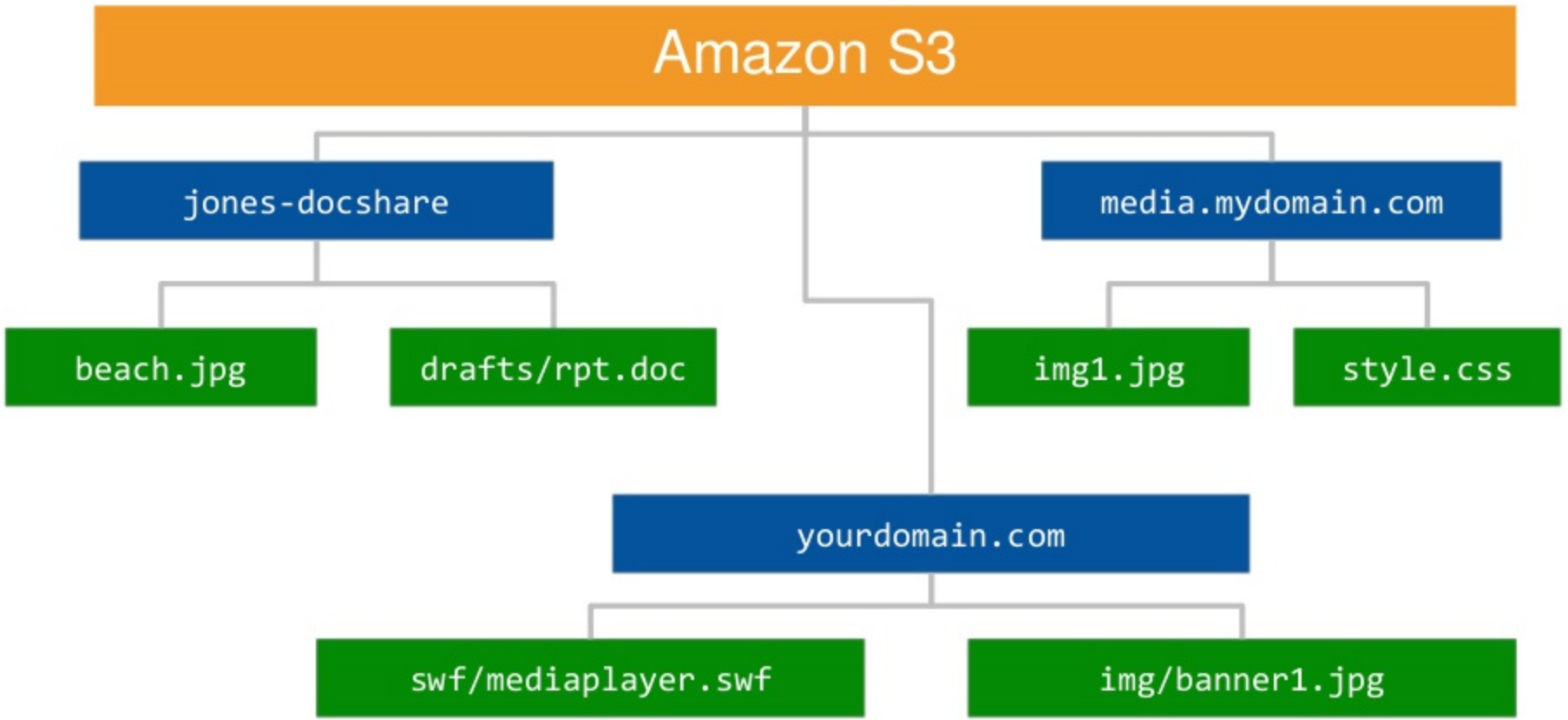
Globally unique

bucket name + object name (key)

Amazon S3 namespace



Amazon S3 namespace

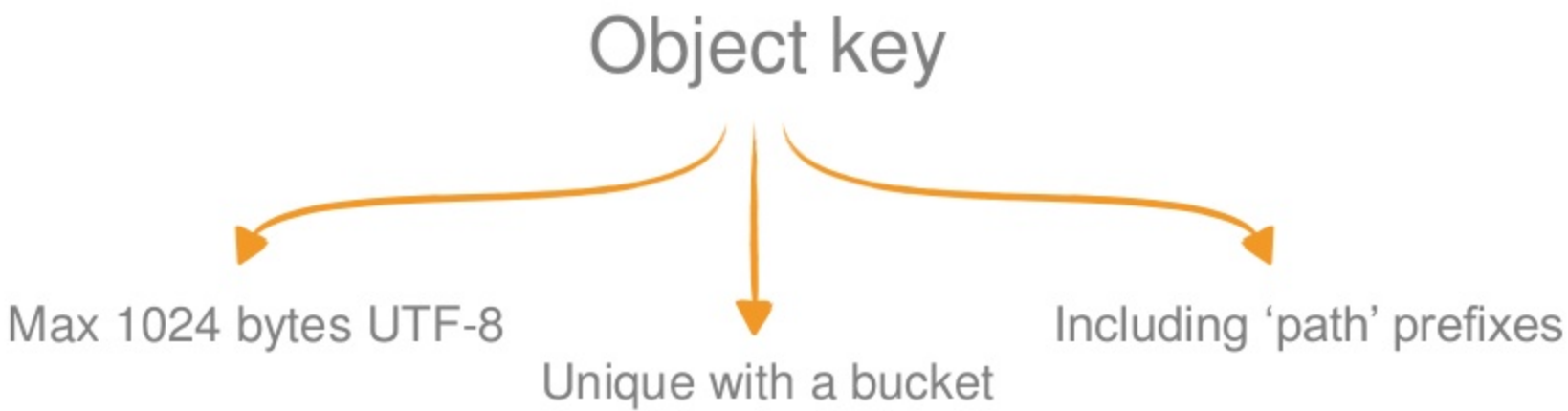


Amazon S3 namespace

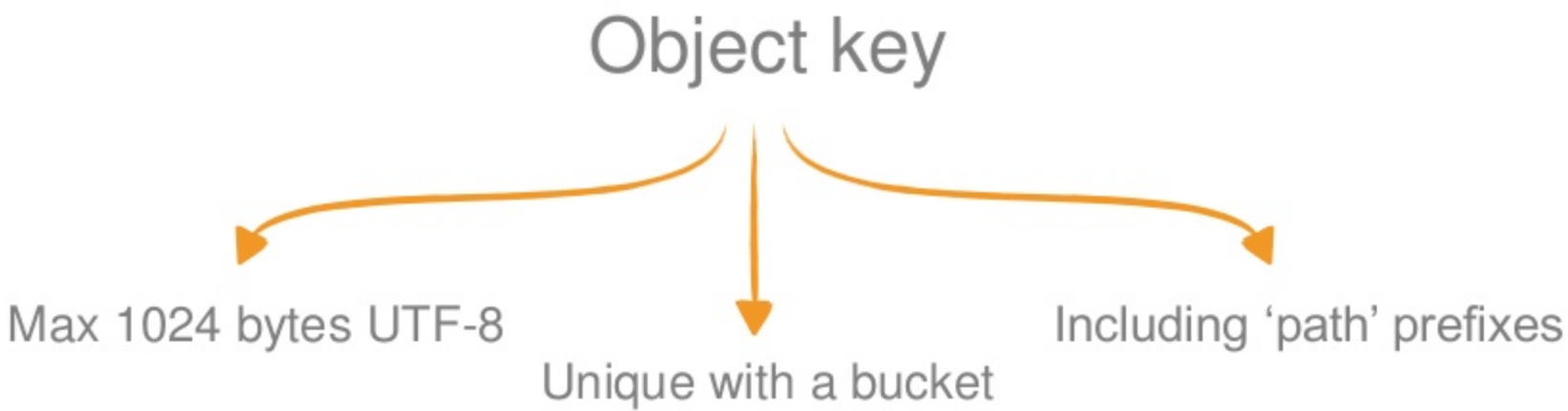
Object key

Unique with a bucket

Amazon S3 namespace



Amazon S3 namespace



assets/js/jquery/plugins/jtables.js

this is an object key

Tip

Throughput optimisation

S3 automatically partitions based upon key prefix:

Bucket:	Object keys:
mynewgame	2134857/gamedata/start.png 2134857/gamedata/resource.rsrc 2134857/gamedata/results.txt 2134858/gamedata/start.png 2134858/gamedata/resource.rsrc 2134858/gamedata/results.txt 2134859/gamedata/start.png 2134859/gamedata/resource.rsrc 2134859/gamedata/results.txt

Tip

Throughput optimisation

S3 automatically partitions based upon key prefix:

Bucket:

mynewgame

Object keys:

2134857/gamedata/start.png
2134857/gamedata/resource.rsrc
2134857/gamedata/results.txt
2134858/gamedata/start.png
2134858/gamedata/resource.rsrc
2134858/gamedata/results.txt
2134859/gamedata/start.png
2134859/gamedata/resource.rsrc
2134859/gamedata/results.txt

Tip

Throughput optimisation

S3 automatically partitions based upon key prefix:

Bucket:

`mynewgame`

Partition:

`mynewgame/2`

Object keys:

```
2134857/gamedata/start.png
2134857/gamedata/resource.rsrc
2134857/gamedata/results.txt
2134858/gamedata/start.png
2134858/gamedata/resource.rsrc
2134858/gamedata/results.txt
2134859/gamedata/start.png
2134859/gamedata/resource.rsrc
2134859/gamedata/results.txt
```

Tip

Throughput optimisation

S3 automatically partitions based upon key prefix:

Bucket:

`mynewgame`

Object keys:

```
7584312/gamedata/start.png
7584312/gamedata/resource.rsrc
7584312/gamedata/results.txt
8584312/gamedata/start.png
8584312/gamedata/resource.rsrc
8584312/gamedata/results.txt
9584312/gamedata/start.png
9584312/gamedata/resource.rsrc
9584312/gamedata/results.txt
```

Reversed game ID

Tip

Throughput optimisation

S3 automatically partitions based upon key prefix:

Bucket:

`mynewgame`

Partitions:

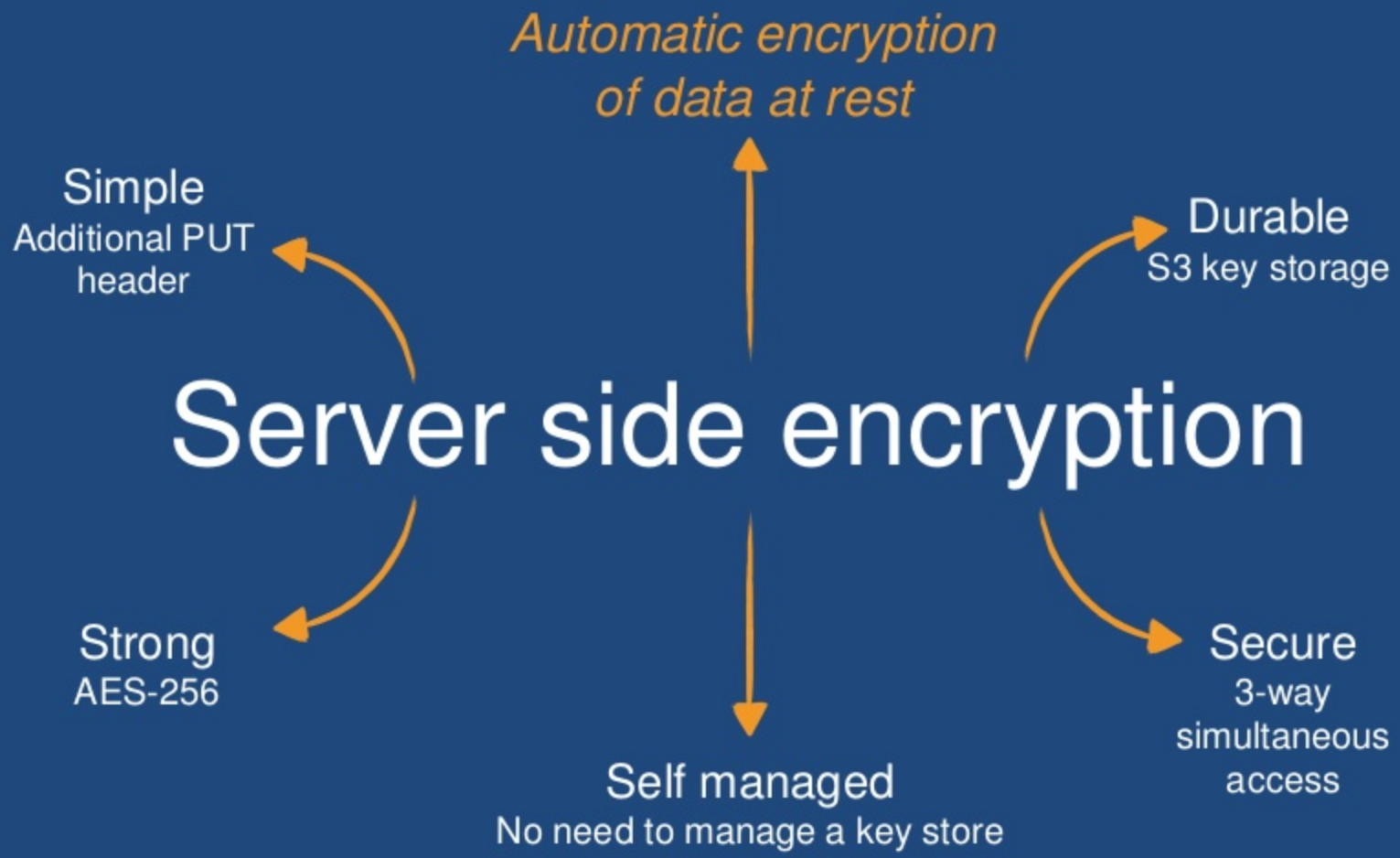
`mynewgame/7`
`mynewgame/8`
`mynewgame/9`

Object keys:

```
7584312/gamedata/start.png
7584312/gamedata/resource.rsrc
7584312/gamedata/results.txt
8584312/gamedata/start.png
8584312/gamedata/resource.rsrc
8584312/gamedata/results.txt
9584312/gamedata/start.png
9584312/gamedata/resource.rsrc
9584312/gamedata/results.txt
```

Encryption

Securing data at rest

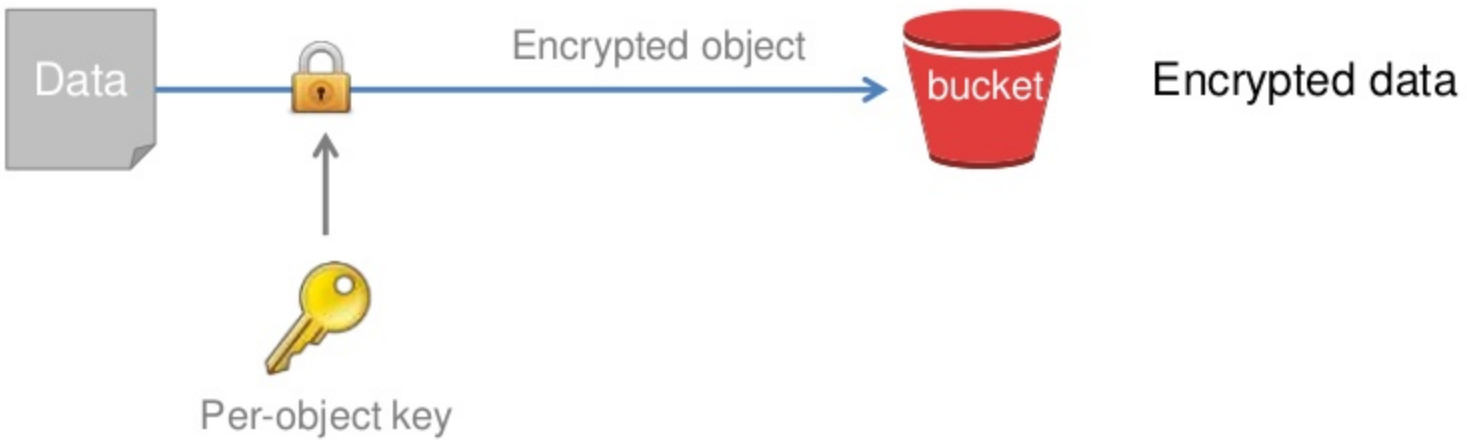

Automatic encryption
of data at rest
Simple
Additional PUT
header
Durable
S3 key storage
Server side encryption
Strong
AES-256
Secure
3-way
simultaneous
access
Self managed
No need to manage a key store

Server side encryption

Object put

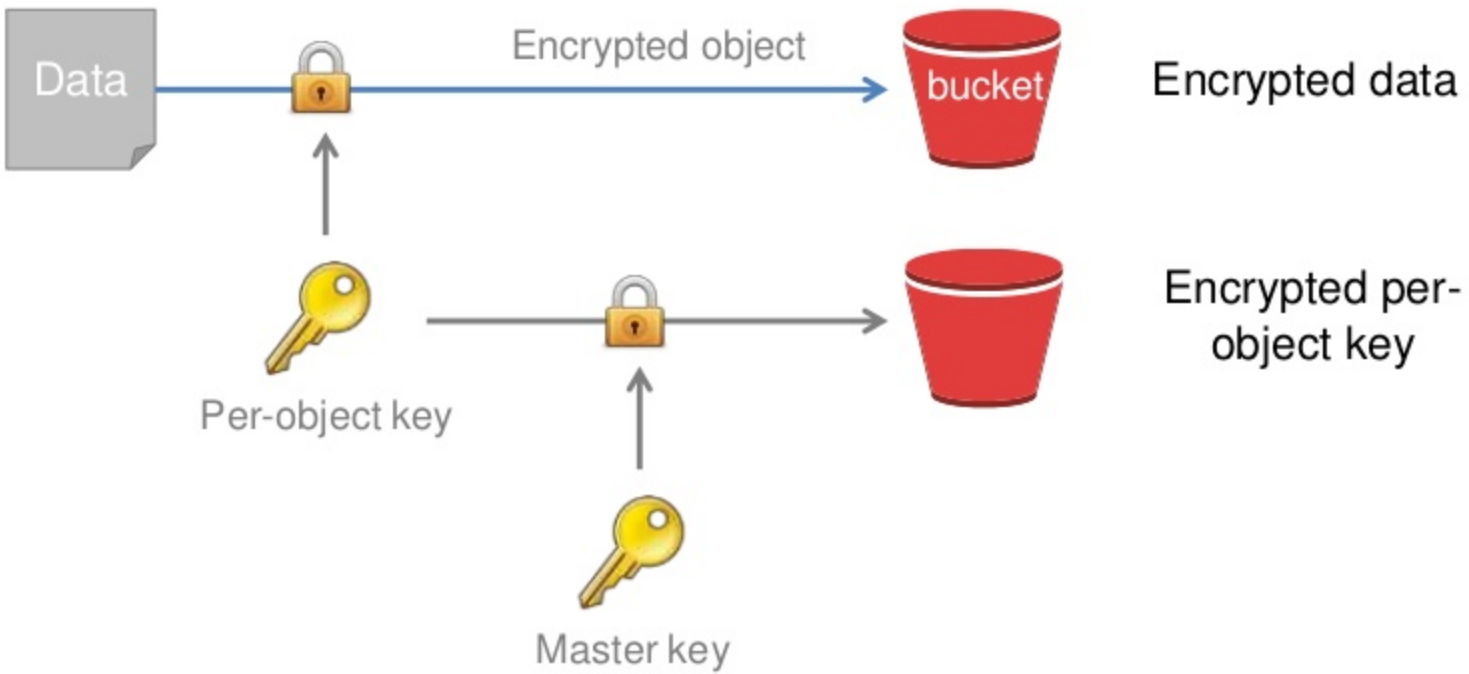
Server side encryption

Object put



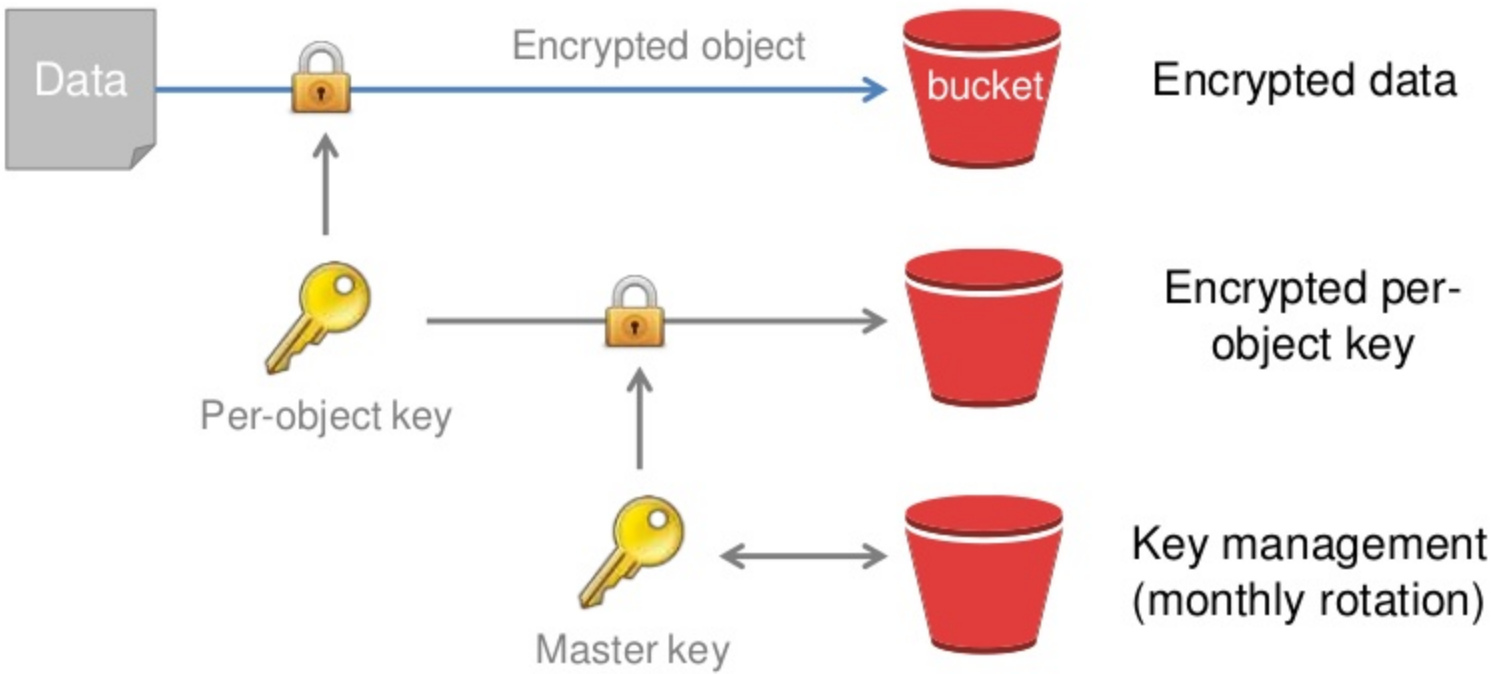
Object put

Server side encryption

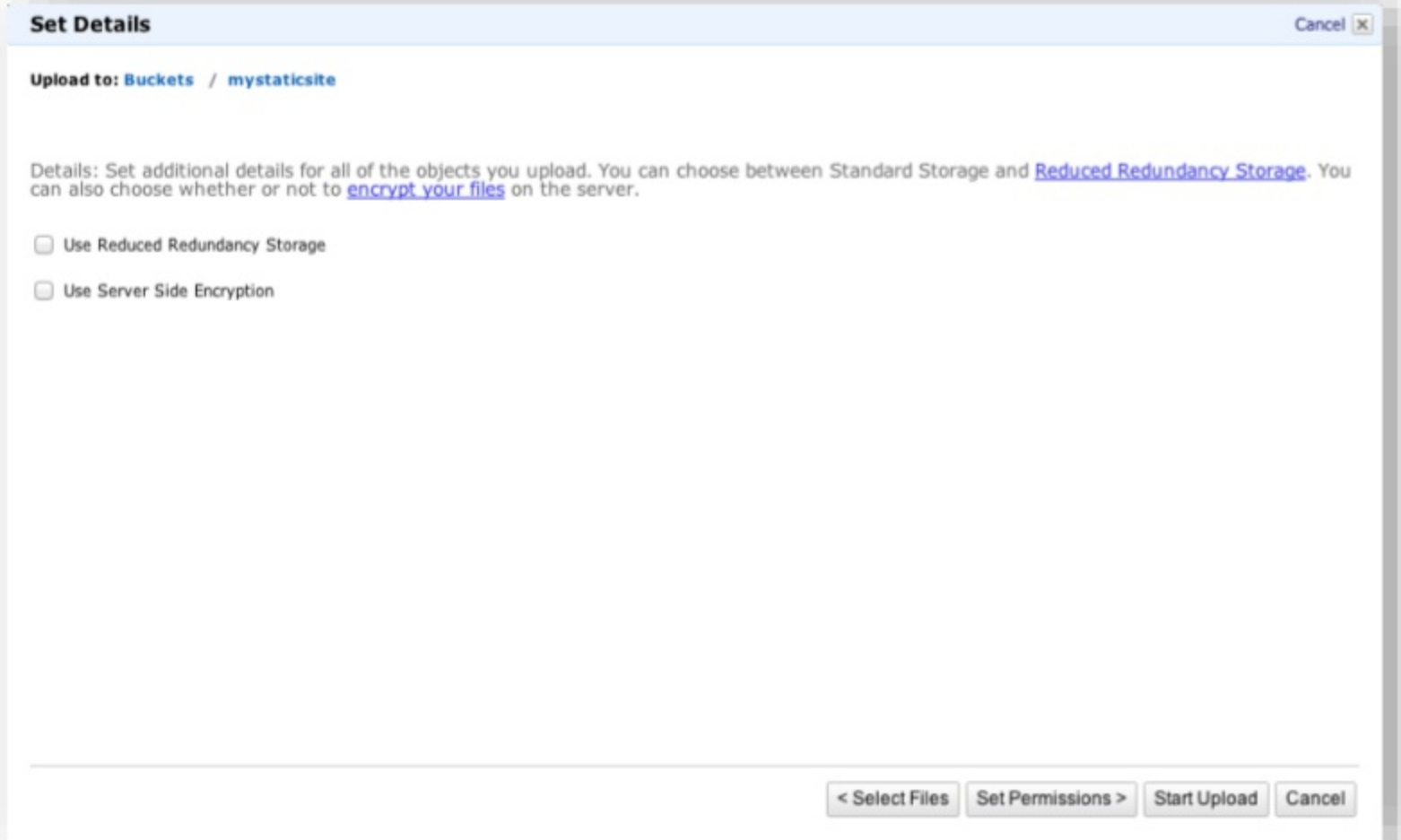


Server side encryption

Object put



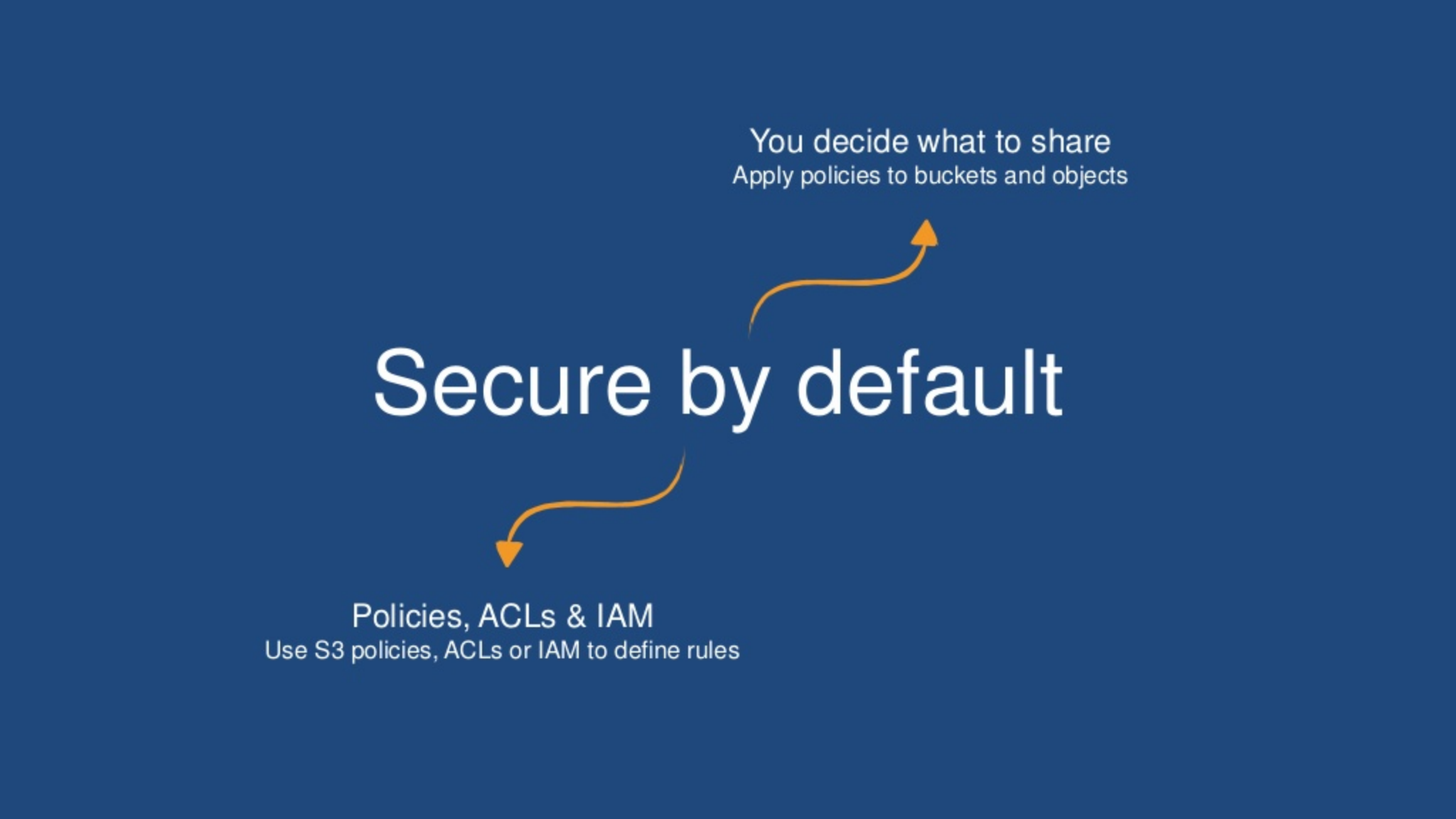
Console



docs.aws.amazon.com/AmazonS3/latest/dev/UsingServerSideEncryption.html

Access controls

You’re in control of who does what

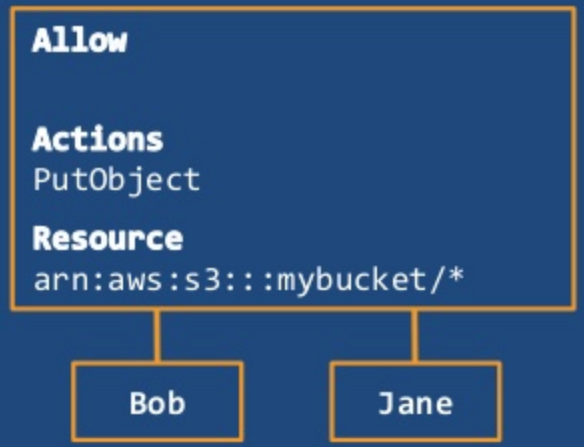

You decide what to share
Apply policies to buckets and objects
Secure by default
Policies, ACLs & IAM
Use S3 policies, ACLs or IAM to define rules

IAM

Fine grained

Administer as part of role based access

Apply policies to S3 at role, user & group level



IAM vs Bucket Policies

Fine grained

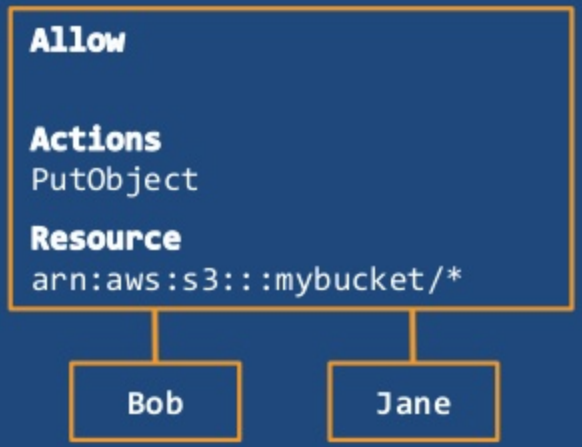
Administer as part of role based access

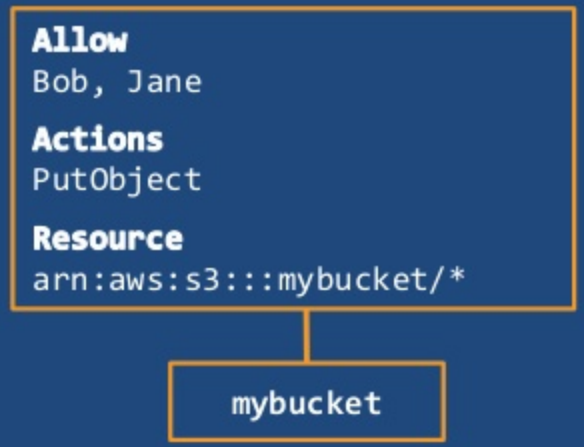
Apply policies to S3 at role, user & group level

Fine grained

Apply policies at the bucket level in S3

Incorporate user restrictions without using IAM



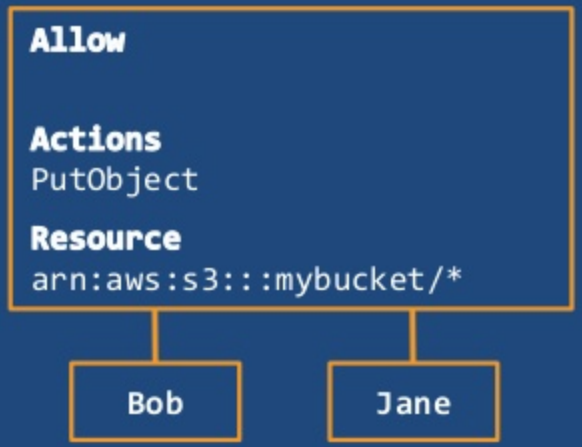


IAM

Fine grained

Administer as part of role based access

Apply policies to S3 at role, user & group level



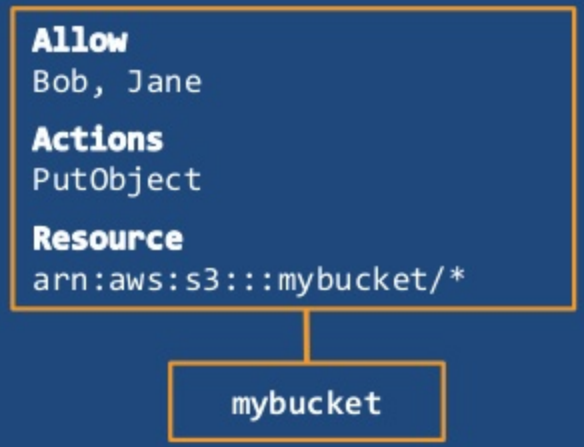
VS

Bucket Policies

Fine grained

Apply policies at the bucket level in S3

Incorporate user restrictions without using IAM



VS

ACLs

Coarse grained

Apply access control rules at the bucket and/or object level in S3

```
{"Statement":[{
        "Effect":"Allow",
        "Principal":{"AWS":["4649-6425", "5243-0045"]},
        "Action":"*",
        "Resource":"/mybucket/*",
         "Condition":{
           "IpAddress":{"AWS:SourceIp":"176.13.0.0/12"}
}}]}
```

Bucket policy


{"Statement":[{
"Effect":"Allow",
"Principal"{"AWS":["4649-6425", "5243-0045"]},
"Action":"*",
"Resource":"/mybucket/*",
"Condition":{
"IpAddress":{"AWS:SourceIp":"176.13.0.0/12"}
}}]}
Accounts to allow
Bucket policy


{"Statement":[{
"Effect":"Allow",
"Principal" {"AWS":["4649-6425", "5243-0045"]},
"Action":"*",
"Resource" "/mybucket/*"
"Condition":{
"IpAddress":{"AWS:SourceIp":"176.13.0.0/12"}
}}]}
Resource
Bucket policy

```
{"Statement":[{
        "Effect":"Allow",
        "Principal" {"AWS":["4649-6425", "5243-0045"]},
        "Action":"*",
        "Resource":"/mybucket/*"
         "Condition":{
           "IpAddress":{"AWS:SourceIp":"176.13.0.0/12"}
}}]}
```

Source address to allow

Bucket policy

Transitions & Lifecycle Management

Automated management of objects

Lifecycle management

Object deletion

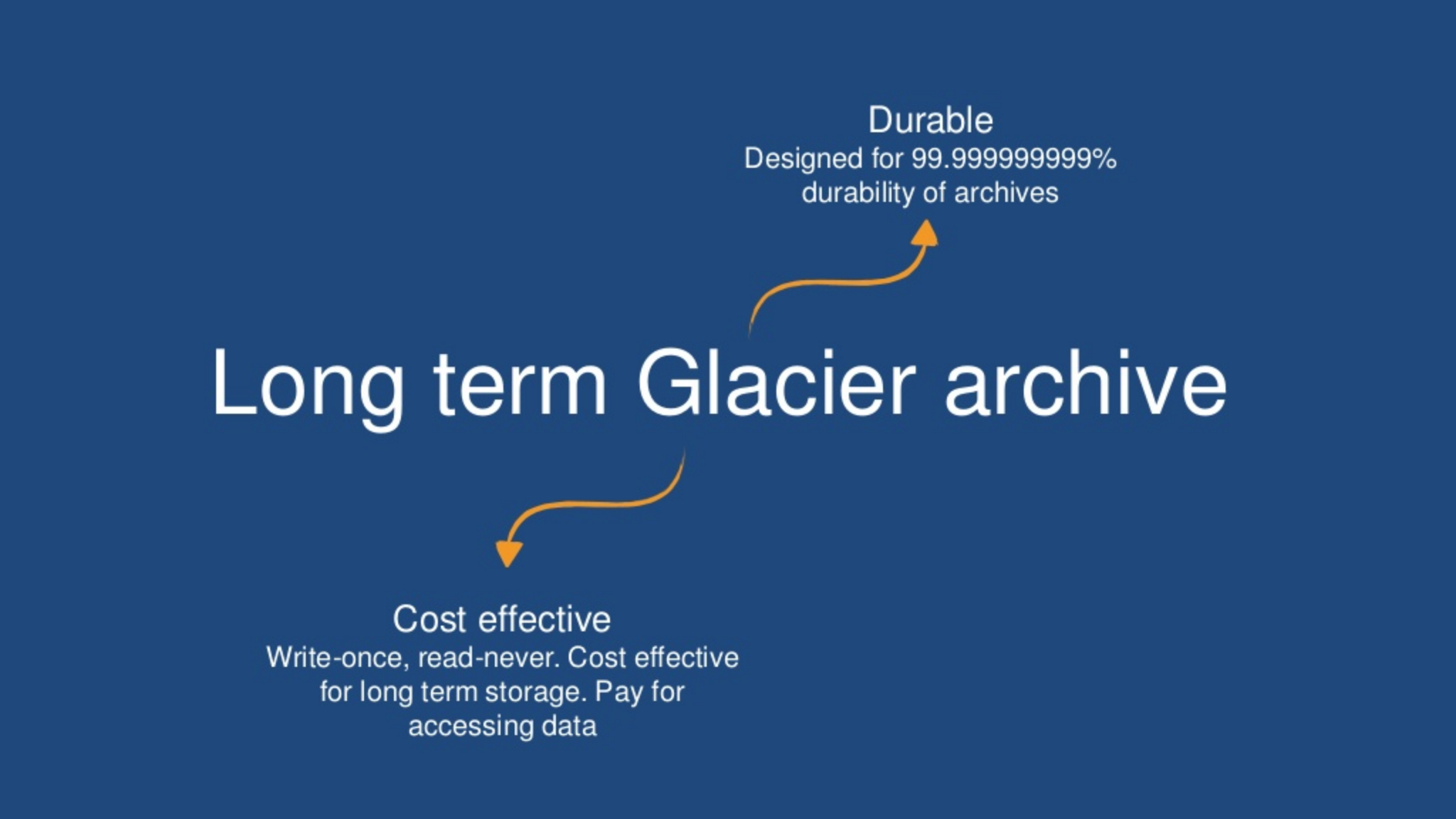
Permanently delete objects from S3

Lifecycle management


Object deletion
Permanently delete objects from S3
Lifecycle management
Object archiving
Move objects to Glacier and out of S3 storage

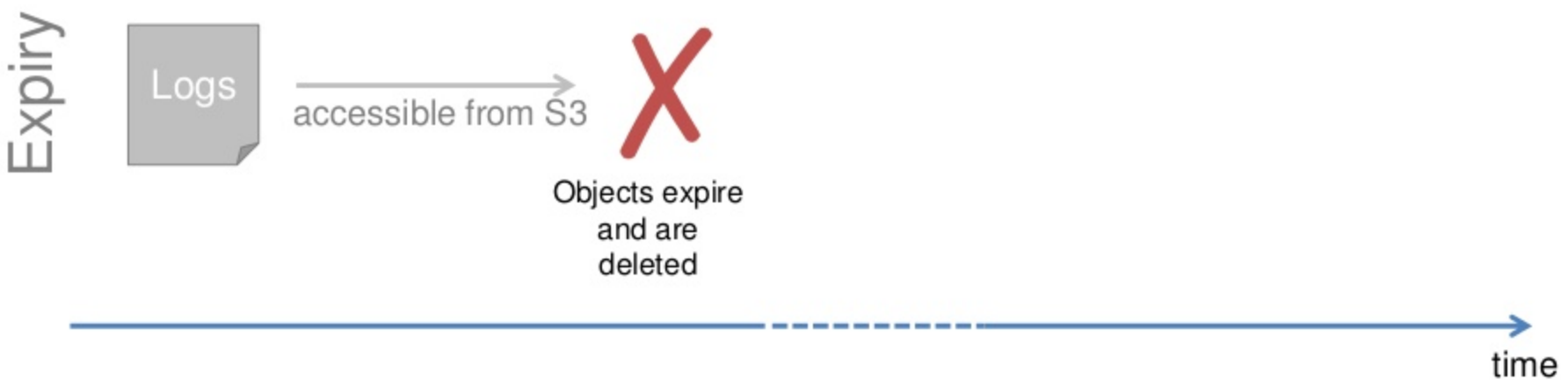
Glacier

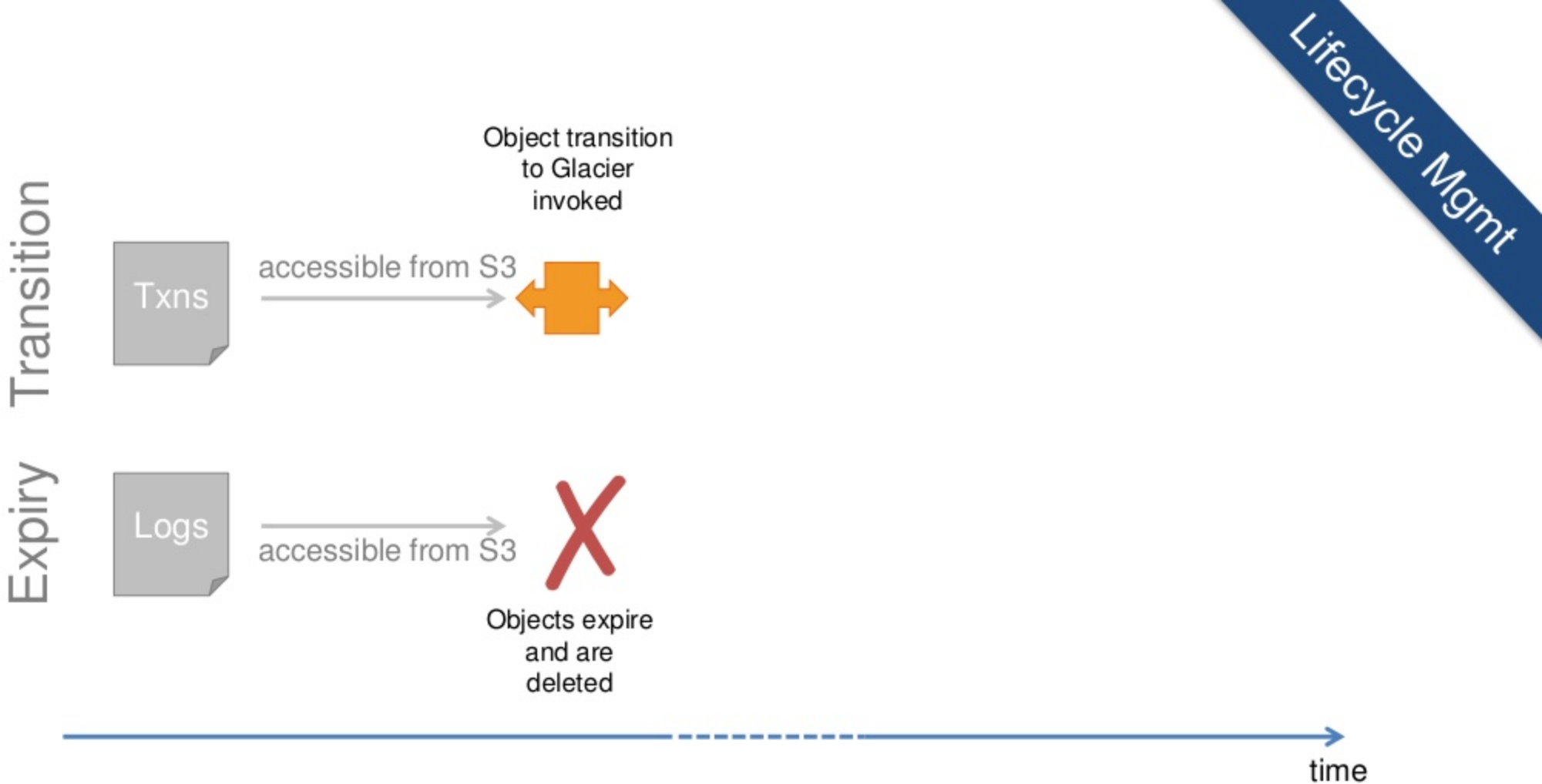
Long term durable archive

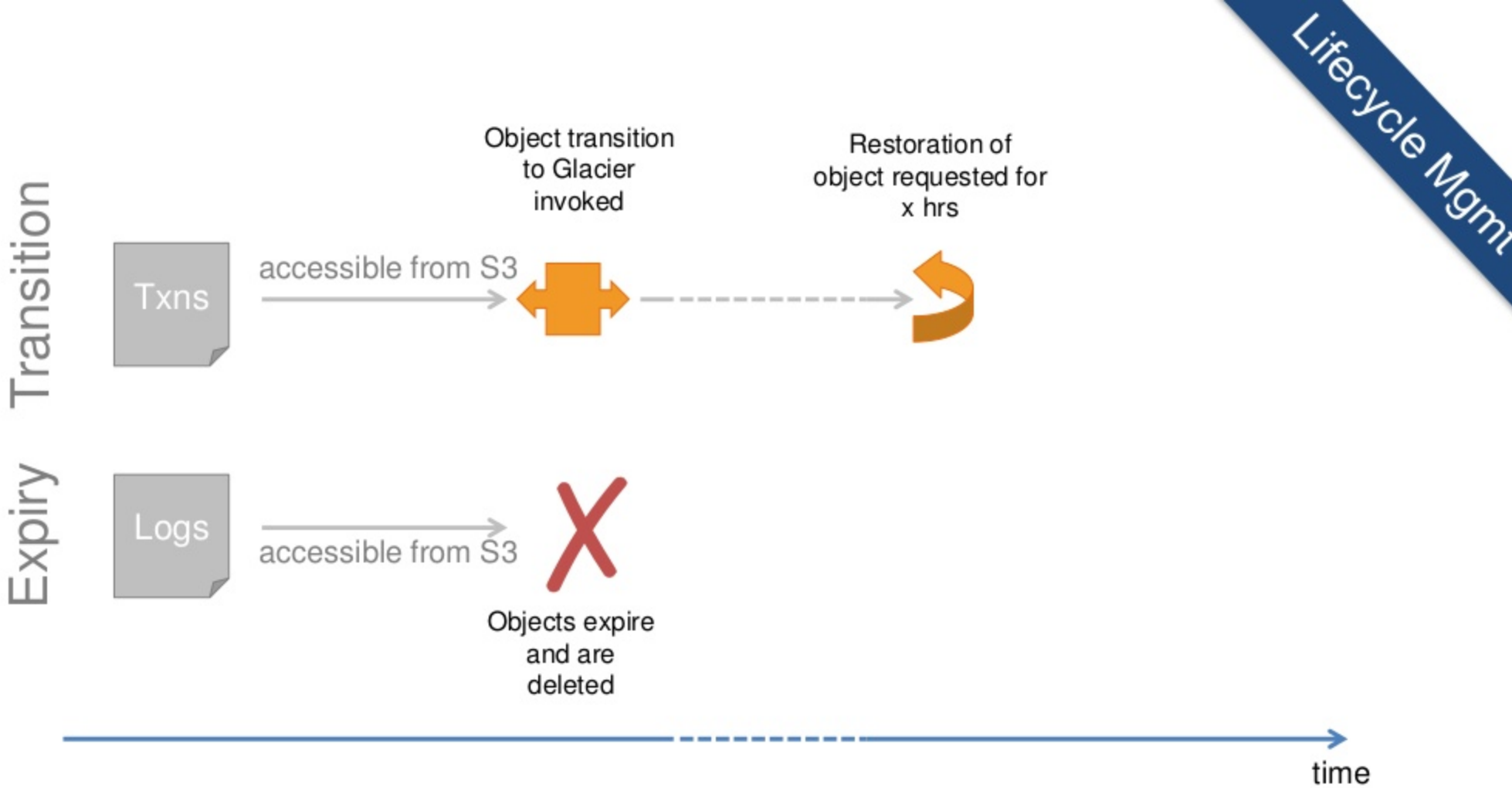

Durable
Designed for 99.999999999% durability of archives
Long term Glacier archive
Cost effective
Write-once, read-never. Cost effective for long term storage. Pay for accessing data

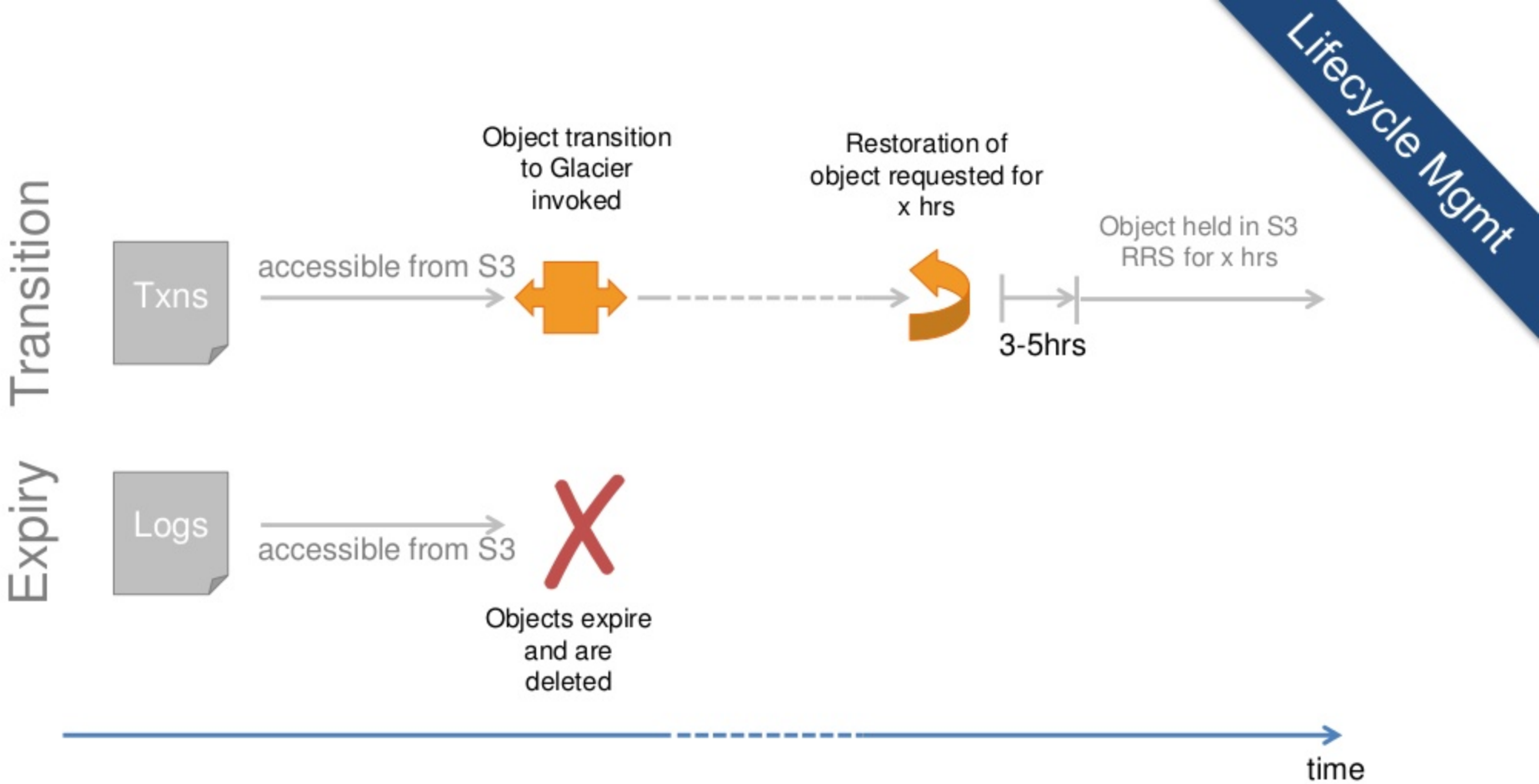
Lifecycle Mgmt

Lifecycle Mgmt




Lifecycle Mgmt
Transition
Object transition to Glacier invoked
Txns
accessible from S3
Expiry
Logs
accessible from S3
Objects expire and are deleted
time

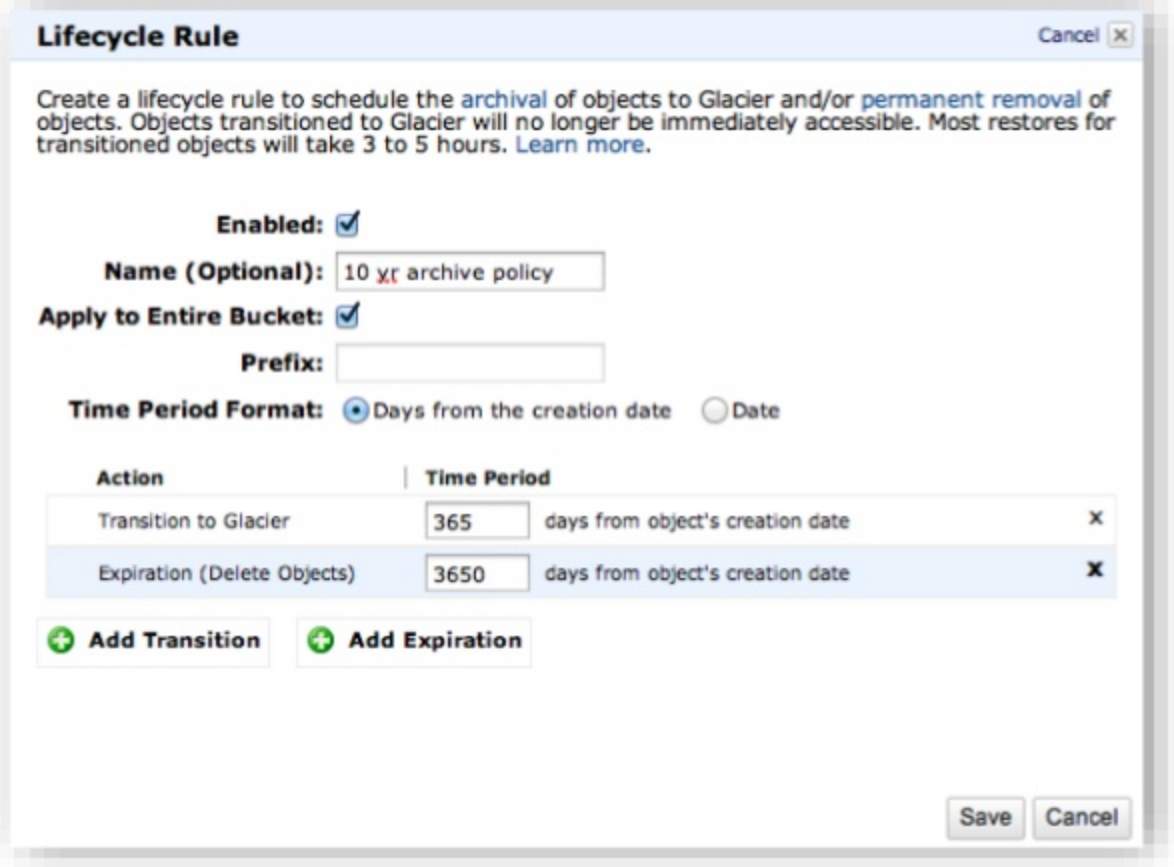

Lifecycle Mgmt
Transition
Expiry
Object transition to Glacier invoked
Restoration of object requested for x hrs
Txns
accessible from S3
Logs
accessible from S3
Objects expire and are deleted
time


Lifecycle Mgmt
Transition
Expiry
Txns
Logs
accessible from S3
accessible from S3
Object transition to Glacier invoked
Restoration of object requested for x hrs
Object held in S3 RRS for x hrs
3-5hrs
Objects expire and are deleted
time

3-5 hour retrieval time

We assume you won’t access often

Console


Lifecycle Rule
Cancel
Create a lifecycle rule to schedule the archival of objects to Glacier and/or permanent removal of objects. Objects transitioned to Glacier will no longer be immediately accessible. Most restores for transitioned objects will take 3 to 5 hours. Learn more.
Enabled:
Name (Optional):
10 yr archive policy
Apply to Entire Bucket:
Prefix:
Time Period Format:
Days from the creation date
Date
Action
Time Period
Transition to Glacier
365
days from object's creation date
Expiration (Delete Objects)
3650
days from object's creation date
Add Transition
Add Expiration
Save
Cancel

```
using (client = new AmazonS3Client()){
  var lifeCycleConfiguration = new LifecycleConfiguration()
  {
    Rules = new List<LifecycleRule>
    {
        new LifecycleRule
        {
             Id = "Archive and delete rule",
             Prefix = "projectdocs/",
             Status = LifecycleRuleStatus.Enabled,
              Transition = new LifecycleTransition()
              {
                   Days = 365,
                   StorageClass = S3StorageClass.Glacier
              },
              Expiration = new LifecycleRuleExpiration()
              {
                   Days = 3650
              }
        }
    }
  };
```

.net

```
using (client = new AmazonS3Client()){
  var lifeCycleConfiguration = new LifecycleConfiguration()
  {
    Rules = new List<LifecycleRule>
    {
        new LifecycleRule
        {
             Id = "Archive and delete rule",
             Prefix = "projectdocs/",
             Status = LifecycleRuleStatus.Enabled,
              Transition = new LifecycleTransition()
              {
                   Days = 365,
                   StorageClass = S3StorageClass.Glacier
              },
              Expiration = new LifecycleRuleExpiration()
              {
                   Days = 3650
              }
        }
    }
  };
```

Transition to Glacier after 1 year

.net

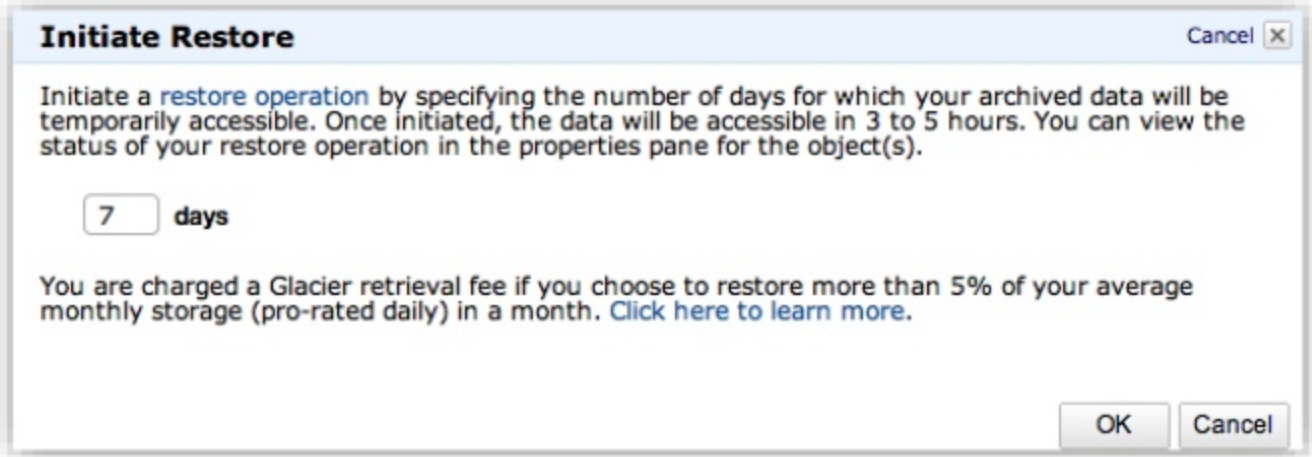
```
using (client = new AmazonS3Client()){
  var lifeCycleConfiguration = new LifecycleConfiguration()
  {
    Rules = new List<LifecycleRule>
    {
        new LifecycleRule
        {
             Id = "Archive and delete rule",
             Prefix = "projectdocs/",
             Status = LifecycleRuleStatus.Enabled,
              Transition = new LifecycleTransition()
              {
                   Days = 365,
                   StorageClass = S3StorageClass.Glacier
              },
              Expiration = new LifecycleRuleExpiration()
              {
                   Days = 3650
              }
        }
    }
  };
```

Delete object after 10 years

.net

Console


Open
Download
Create Folder...
Upload
Make Public
Rename
Delete
Initiate Restore
Cut
Copy
Paste
Properties


Initiate Restore
Cancel
Initiate a restore operation by specifying the number of days for which your archived data will be temporarily accessible. Once initiated, the data will be accessible in 3 to 5 hours. You can view the status of your restore operation in the properties pane for the object(s).
7
days
You are charged a Glacier retrieval fee if you choose to restore more than 5% of your average monthly storage (pro-rated daily) in a month. Click here to learn more.
OK
Cancel

```
POST /ObjectName?restore HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: signatureValue
Content-MD5: MD5

<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-3-01">
   <Days>NumberOfDays</Days>
</RestoreRequest>
```

REST API

```
POST /ObjectName?restore HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: signatureValue
Content-MD5: MD5

<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-3-01">
   <Days>NumberOfDays</Days>
</RestoreRequest>
```

Response codes:

202 Accepted	*Restore request accepted*
200 OK	*Object already restored, number of days updated*
409 Conflict	*Restoration already in progress*

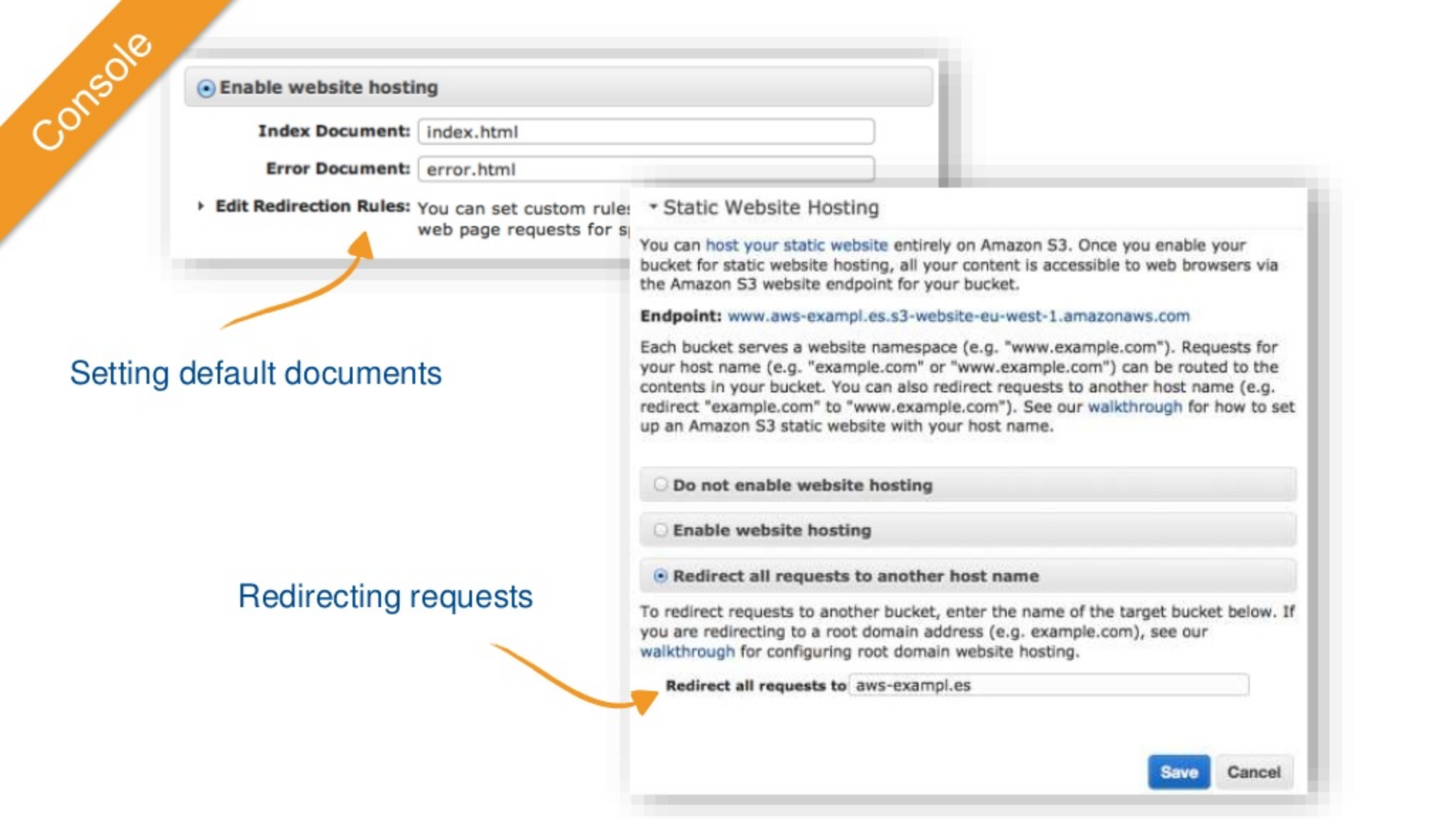
REST API

Website hosting

Static sites straight from S3

It’s a web service…

…so serve up web content


Console
Enable website hosting
Index Document: index.html
Error Document: error.html
Edit Redirection Rules: You can set custom rules
web page requests for s
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: www.aws-exampl.es.s3-website-eu-west-1.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
Redirect all requests to another host name
To redirect requests to another bucket, enter the name of the target bucket below. If you are redirecting to a root domain address (e.g. example.com), see our walkthrough for configuring root domain website hosting.
Redirect all requests to aws-exampl.es
Save
Cancel
Setting default documents
Redirecting requests

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

Bucket policy

Website addressing

{bucket-name}.s3-website-{region}.amazonaws.com

e.g. mybucket.s3-website-eu-west-1.amazonaws.com

Normal addressing

s3-{region}.amazonaws.com/{bucket-name}/{object-key}

e.g. s3-eu-west-1.amazonaws.com/mybucket/img.png

{bucket-name}.s3-{region}.amazonaws.com/{object-key}

e.g. mybucket.s3-eu-west-1.amazonaws.com/img.png

Websites

Record set for:

`aws-exampl.es`

Websites

Record set for:

`aws-exampl.es`

Website bucket name:

`www.aws-exampl.es`

Website bucket name:

`aws-exampl.es`

Websites

Record set for:

aws-exampl.es

Website redirect to:

aws-exampl.es

Website bucket name:

www.aws-exampl.es

Website bucket name:

aws-exampl.es

Websites

Record set for:

aws-exampl.es

A Record 'Alias' to S3 website:

aws-exampl.es @
s3-website-eu-west-1.amazonaws.com

Website redirect to:

aws-exampl.es

Website bucket name:

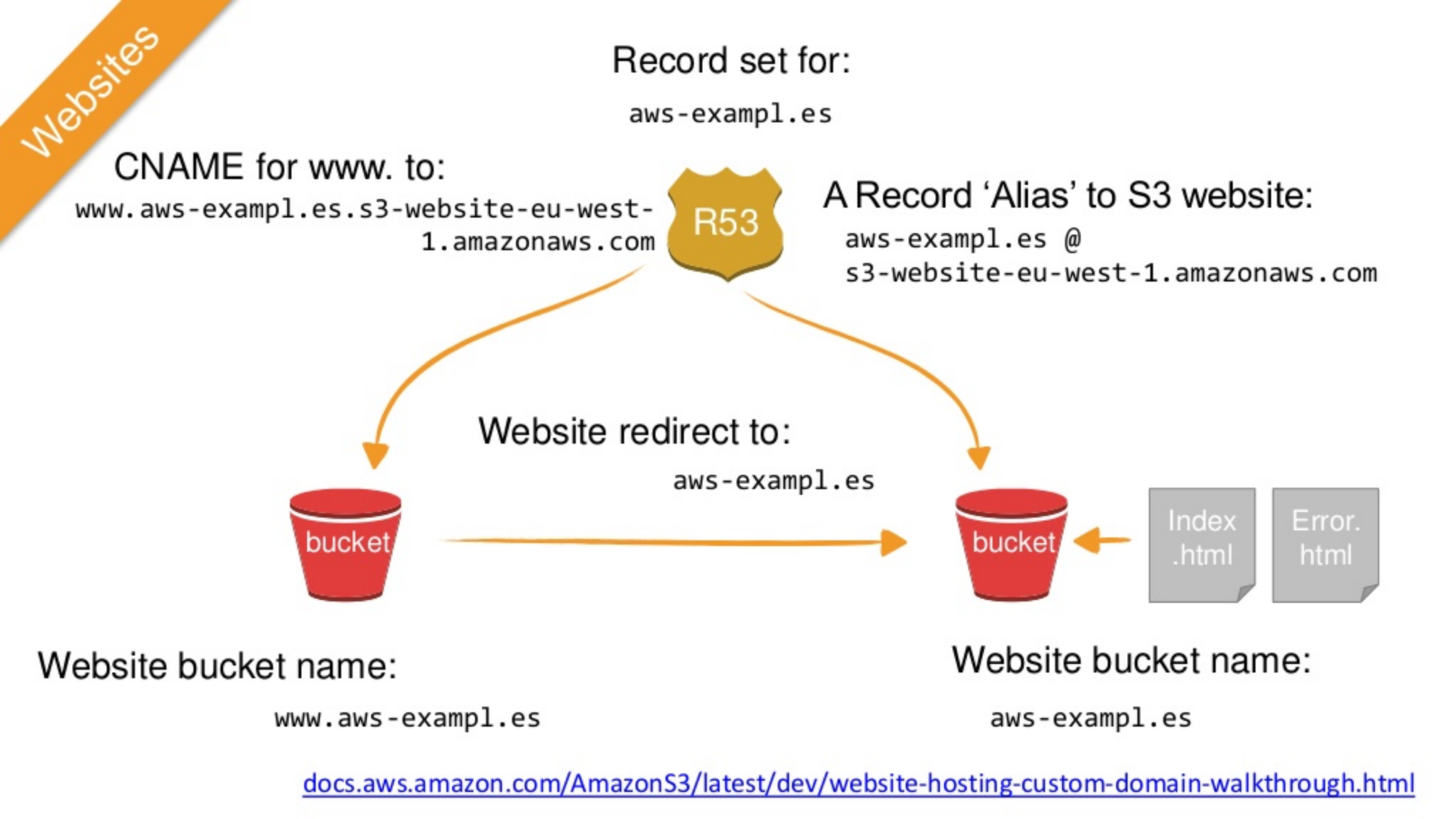
www.aws-exampl.es

Website bucket name:

aws-exampl.es

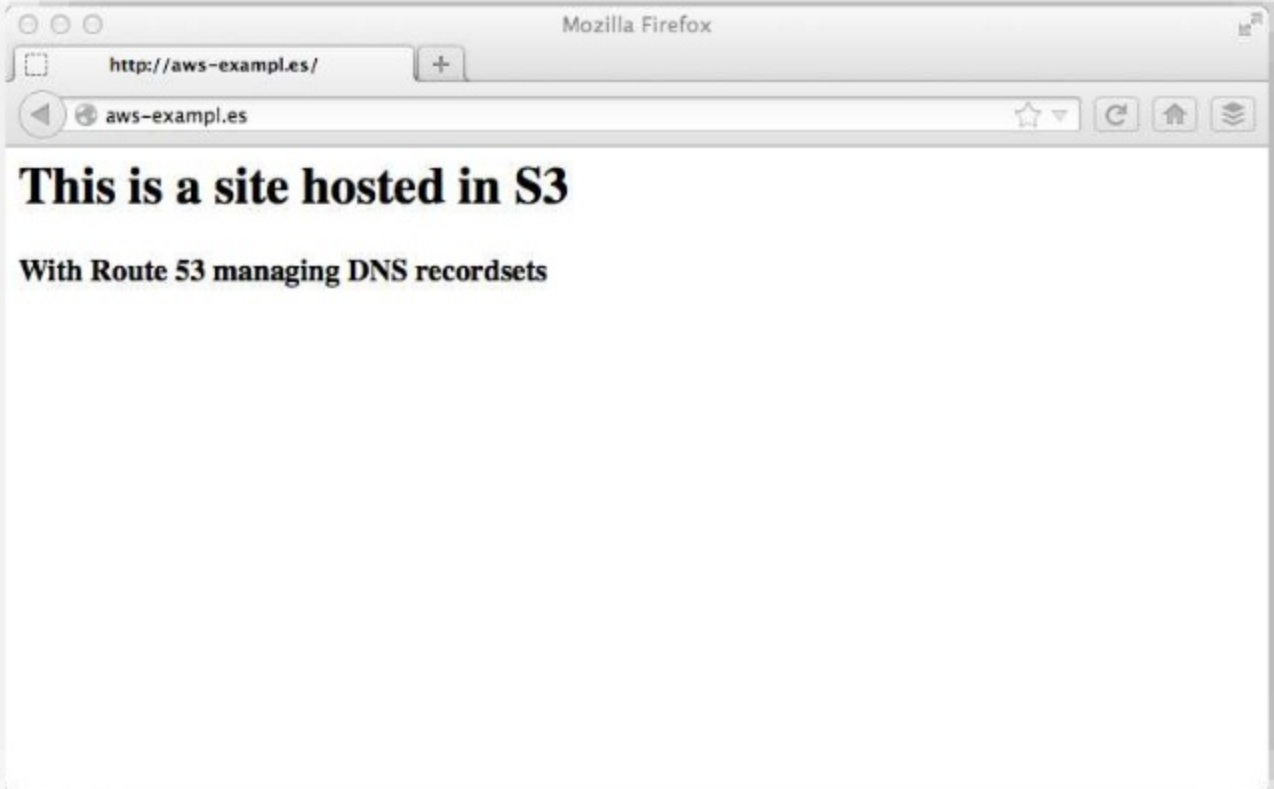
docs.aws.amazon.com/AmazonS3/latest/dev/website-hosting-custom-domain-walkthrough.html

Websites



docs.aws.amazon.com/AmazonS3/latest/dev/website-hosting-custom-domain-walkthrough.html

Websites



docs.aws.amazon.com/AmazonS3/latest/dev/website-hosting-custom-domain-walkthrough.html

Time-Limited URLS

Control how long objects can be accessed for

Signed URLs

Provide time-limited access to specific objects that expires after a set period

Access Permissions

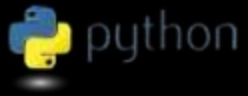
Use on objects in non-public buckets to prevent access once the signed URL has expired

```
https://ianmas-aws.testbucket.s3.amazonaws.com/testfile.txt
?Signature=JHCa39GV1fKRKkEnAWzI88lH7f8%3D
&Expires=1391425438
&AWSAccessKeyId=AKIAIRBKBJ3ZAYAXFC2Q
```

python

>>>

Python boto



1st parameter is link lifetime in seconds

```
>>> import boto
>>> conn = boto.connect_s3()
>>> conn.generate_url(300, 'GET', bucket='ianmas-aws.testbucket',
key='testfile.txt')

'https://ianmas-
aws.testbucket.s3.amazonaws.com/testfile.txt?Signature=QUWA%2BGOohFeJ
5pdEzxtdaIFIA6w%3D&Expires=1391425142&AWSAccessKeyId=AKIAIRBKBJ3ZAYAX
FC2Q'
```

Force a non-SSL link

```
>>> conn.generate_url(300, 'GET', bucket='ianmas-aws.testbucket',
key='testfile.txt', force_http=True)

'http://ianmas-
aws.testbucket.s3.amazonaws.com/testfile.txt?Signature=tALx9KeeSisDSc
0N7KlM%2BIDFZXI%3D&Expires=1391425562&AWSAccessKeyId=AKIAIRBKBJ3ZAYAX
FC2Q'
```

Python boto

Error response: link expired

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>AccessDenied</Code>
  <Message>Request has expired</Message>
  <RequestId>70297390BE427DC7</RequestId>
  <Expires>2014-02-03T11:03:58Z</Expires>
  <HostId>I0rI0OWUCnBttFSpEw6Mx4u8uRHgtOSw9k2euDW37skFCU7HH0ulSkUGGaUbn2vg</HostId>
  <ServerTime>2014-02-03T11:08:22Z</ServerTime>
</Error>
```

Object versioning

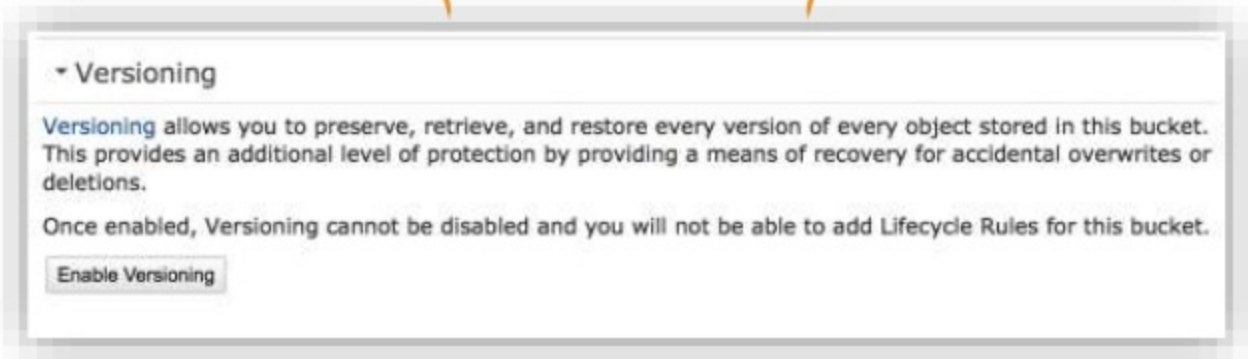
Preserving object histories

Bucket level

Automatically preserves all copies of objects

Persistent

Even deleted object history is held

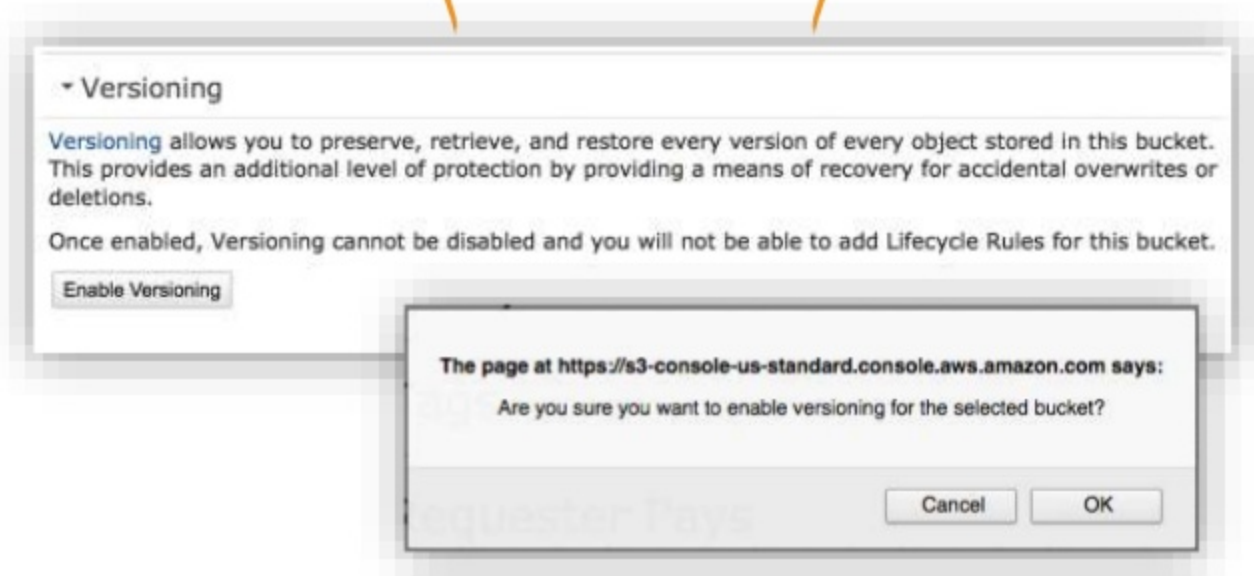


Bucket level

Automatically preserves all copies of objects

Persistent

Even deleted object history is held

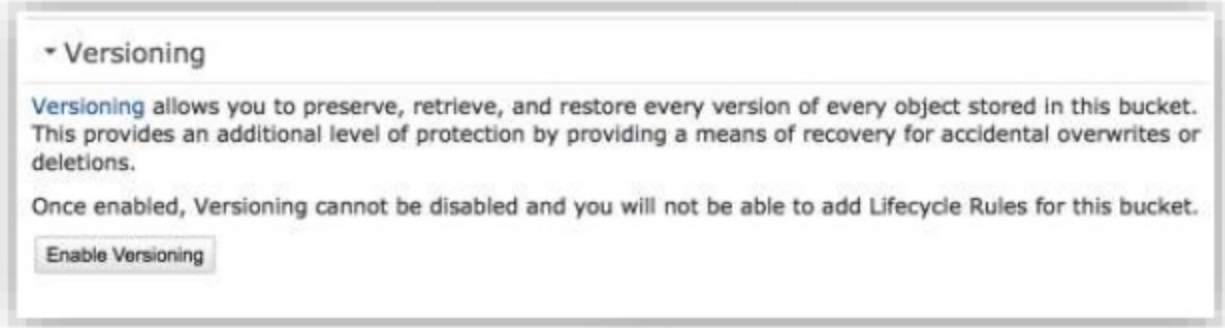


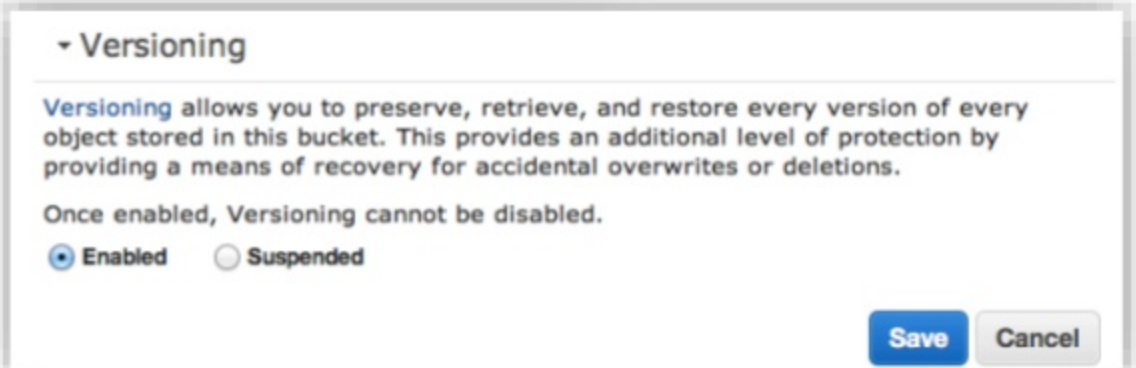
Bucket level

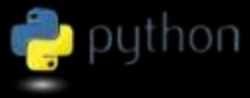
Automatically preserves all copies of objects

Persistent

Even deleted object history is held







Python boto

```
>>> import boto
>>> conn = boto.connect_s3()
>>> bucket = conn.get_bucket('mybucket')

>>> versions = bucket.list_versions()
>>> for version in versions:
...     print version.name + version.version_id
...

myfile.txt jU9eVv800OlP4PQx6zskMEyPIoExne57
myfile.txt xOJzMvMmGv0Bx2v4QpIypbkkH2XE2yyq
myfile.txt 8cjozv9Hmkzum8xj.8q8BZxR5CuXnzon
```

Python boto

```
>>> key = bucket.get_key('myfile.txt',
version_id='8cjozv9Hmkzum8xj.8q8BZxR5CuXnzon')
>>> key.get_contents_as_string()

'this is version 1 of my file'
```

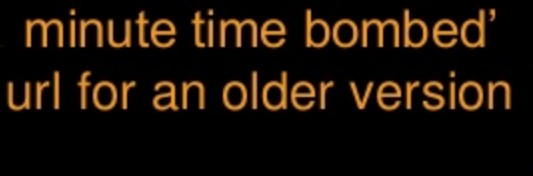
Python boto

```
>>> key = bucket.get_key('myfile.txt',
version_id='8cjozv9Hmkzum8xj.8q8BZxR5CuXnzon’)
>>> key.get_contents_as_string()

'this is version 1 of my file’

>>> key = bucket.get_key('myfile.txt',
version_id='xOJzMvMmGv0Bx2v4QpIypbkkH2XE2yyq’)
>>> key.get_contents_as_string()

'this is version 2 of my file’
```

python

Python boto

```
>>> key = bucket.get_key('myfile.txt',
version_id='8cjozv9Hmkzum8xj.8q8BZxR5CuXnzon’)
>>> key.get_contents_as_string()

'this is version 1 of my file’

>>> key = bucket.get_key('myfile.txt',
version_id='xOJzMvMmGv0Bx2v4QpIypbkkH2XE2yyq’)
>>> key.get_contents_as_string()

'this is version 2 of my file’

>>> key.generate_url(600)
'https://mybucket.s3.amazonaws.com/myfile.txt?Signature=ABCD&
Expires=1358857379&AWSAccessKeyId=AB&
versionId=xOJzMvMmGv0Bx2v4QpIypbkkH2XE2yyq'
```



Python boto

Metadata

System and user generated

System metadata

Name	Description	Editable?
Date	Object creation date	No
Content-Length	Object size in bytes	No
Content-MD5	Base64 encoded 128bit MD5 digest	No
x-amz-server-side-encryption	Server side encryption enabled for object	Yes
x-amz-version-id	Object version	No
x-amz-delete-marker	Indicates a version enabled object is deleted	No
x-amz-storage-class	Storage class for the object	Yes
x-amz-website-redirect-location	Redirects request for the object to another object or external URL	Yes

User metadata

Key-value pairs stored with objects and returned with requests

x-amz-meta-{your metadata key in here}

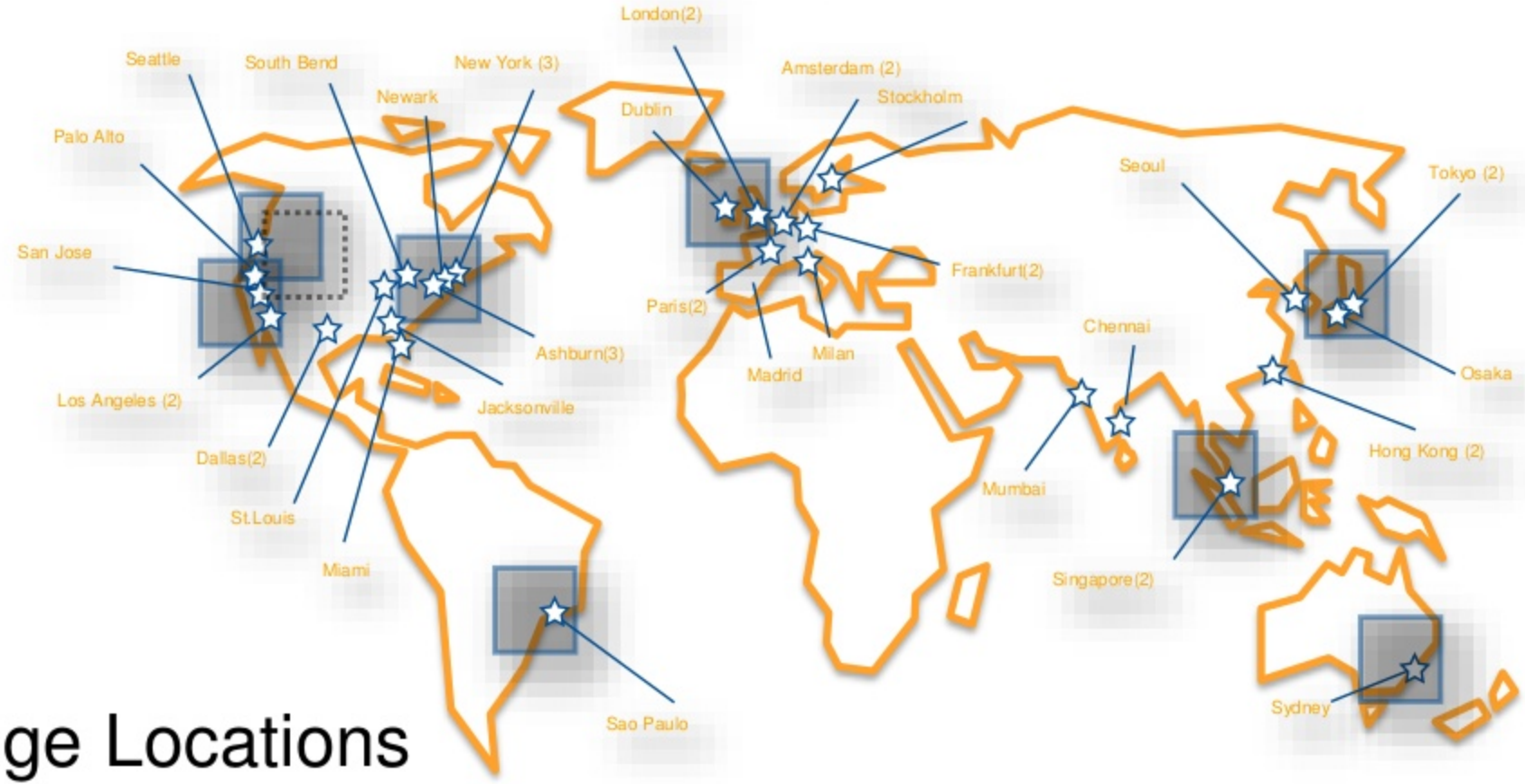
```
>>> key.set_metadata('my_tag', 'my metadata')

>>> key.get_metadata('my_tag')
'my metadata'
```

Python boto

CloudFront

Content delivery from global edge locations

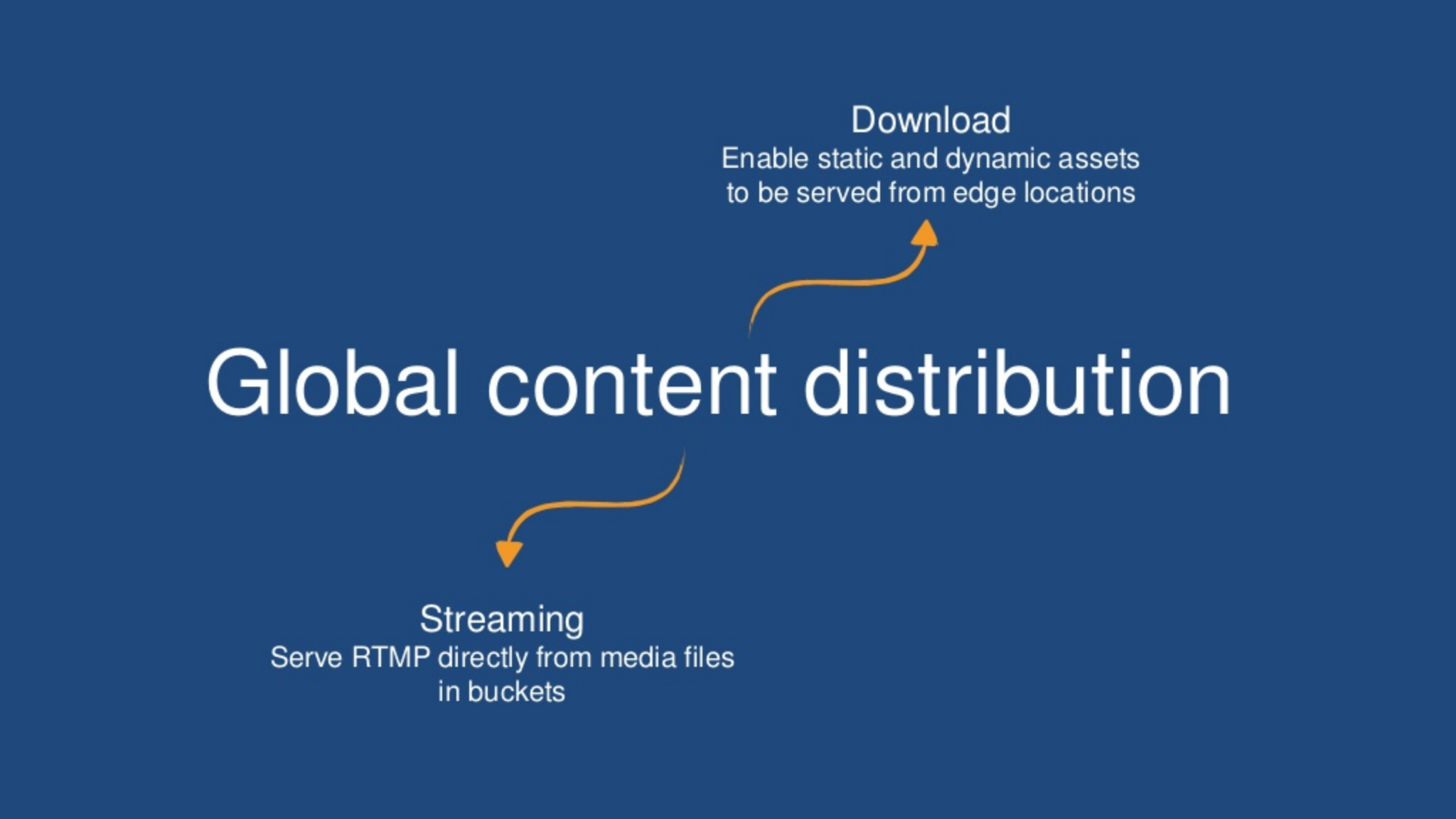
CloudFront Edge Locations



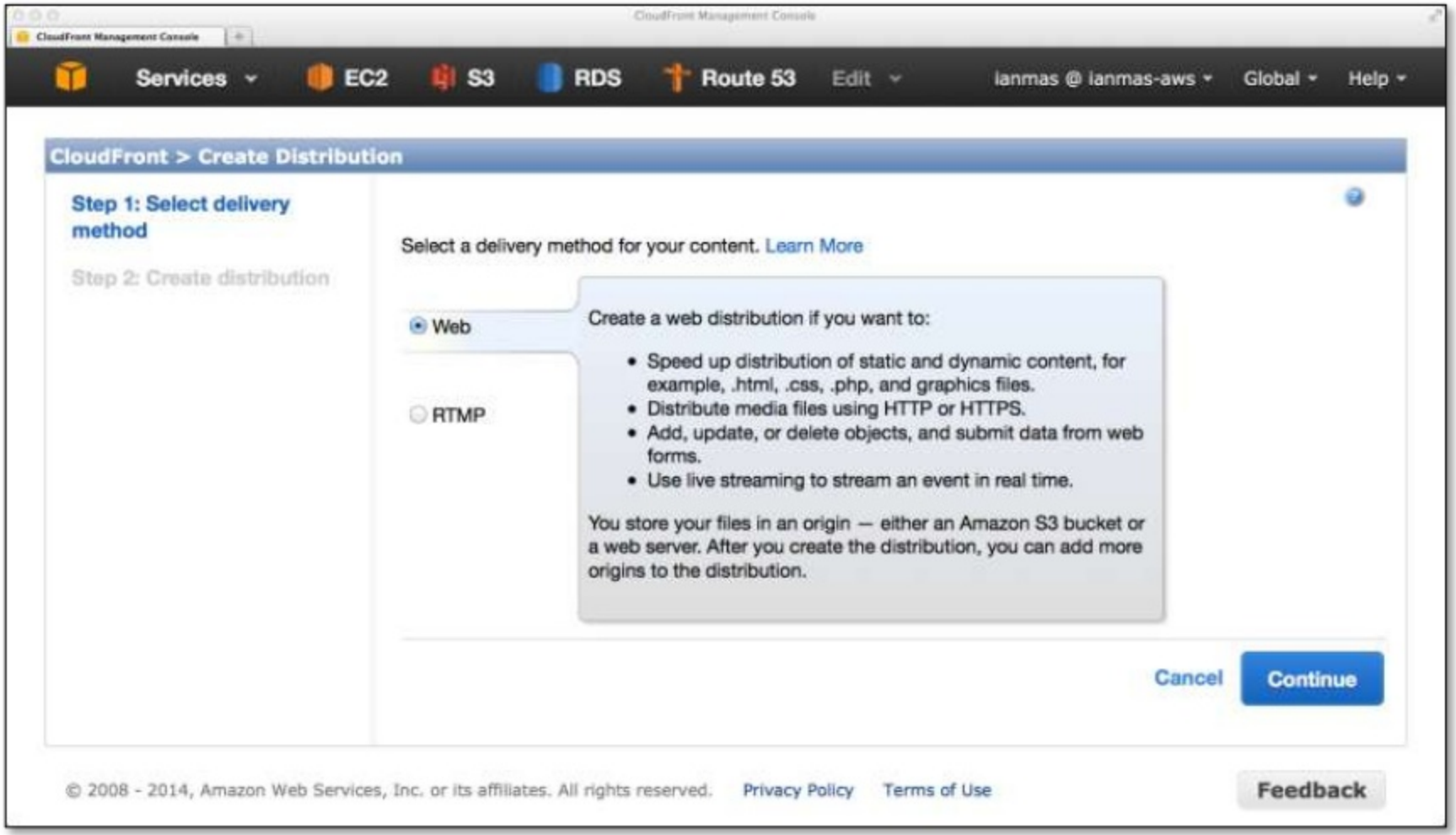
☆ Edge Locations

To deliver content to end users with lower latency

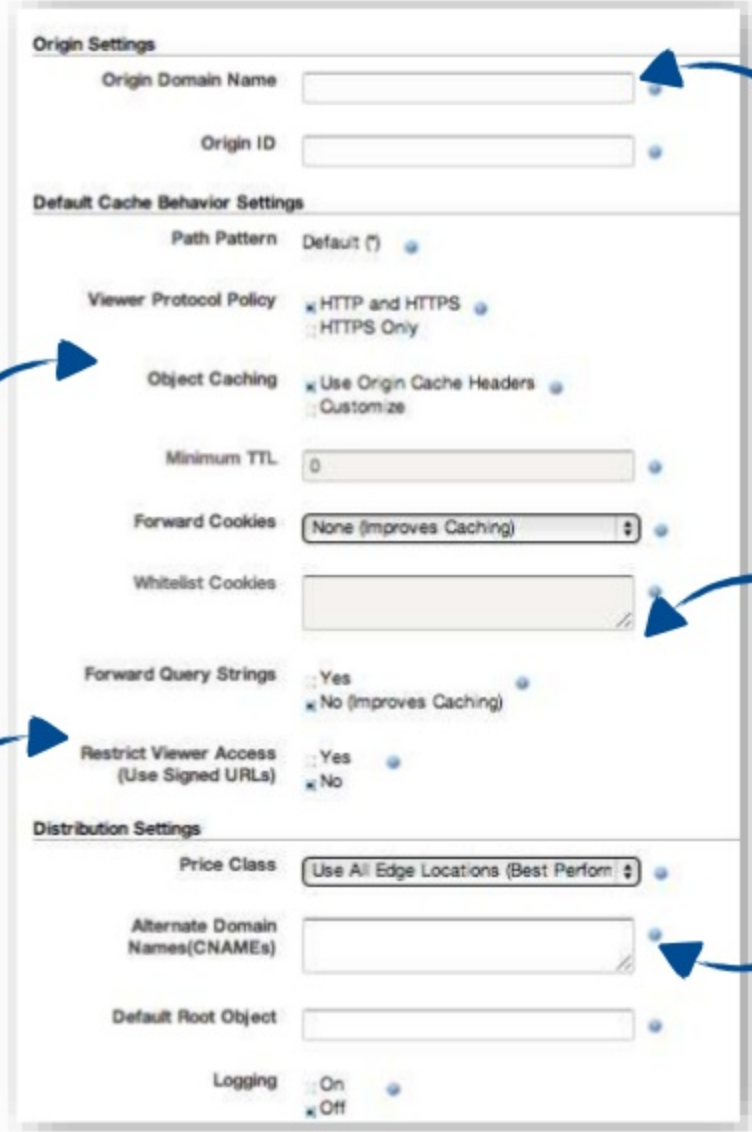
A global network of edge locations Supports global DNS infrastructure (Route53) and Cloud Front CDN


Download
Enable static and dynamic assets
to be served from edge locations
Global content distribution
Streaming
Serve RTMP directly from media files
in buckets

Console


CloudFront Management Console
Services
EC2
S3
RDS
Route 53
Edit
ianmas @ ianmas-aws
Global
Help
CloudFront > Create Distribution
Step 1: Select delivery method
Step 2: Create distribution
Select a delivery method for your content. Learn More
Web
RTMP
Create a web distribution if you want to:
Speed up distribution of static and dynamic content, for example, .html, .css, .php, and graphics files.
Distribute media files using HTTP or HTTPS.
Add, update, or delete objects, and submit data from web forms.
Use live streaming to stream an event in real time.
You store your files in an origin — either an Amazon S3 bucket or a web server. After you create the distribution, you can add more origins to the distribution.
Cancel
Continue
© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Feedback

Console



S3 Bucket

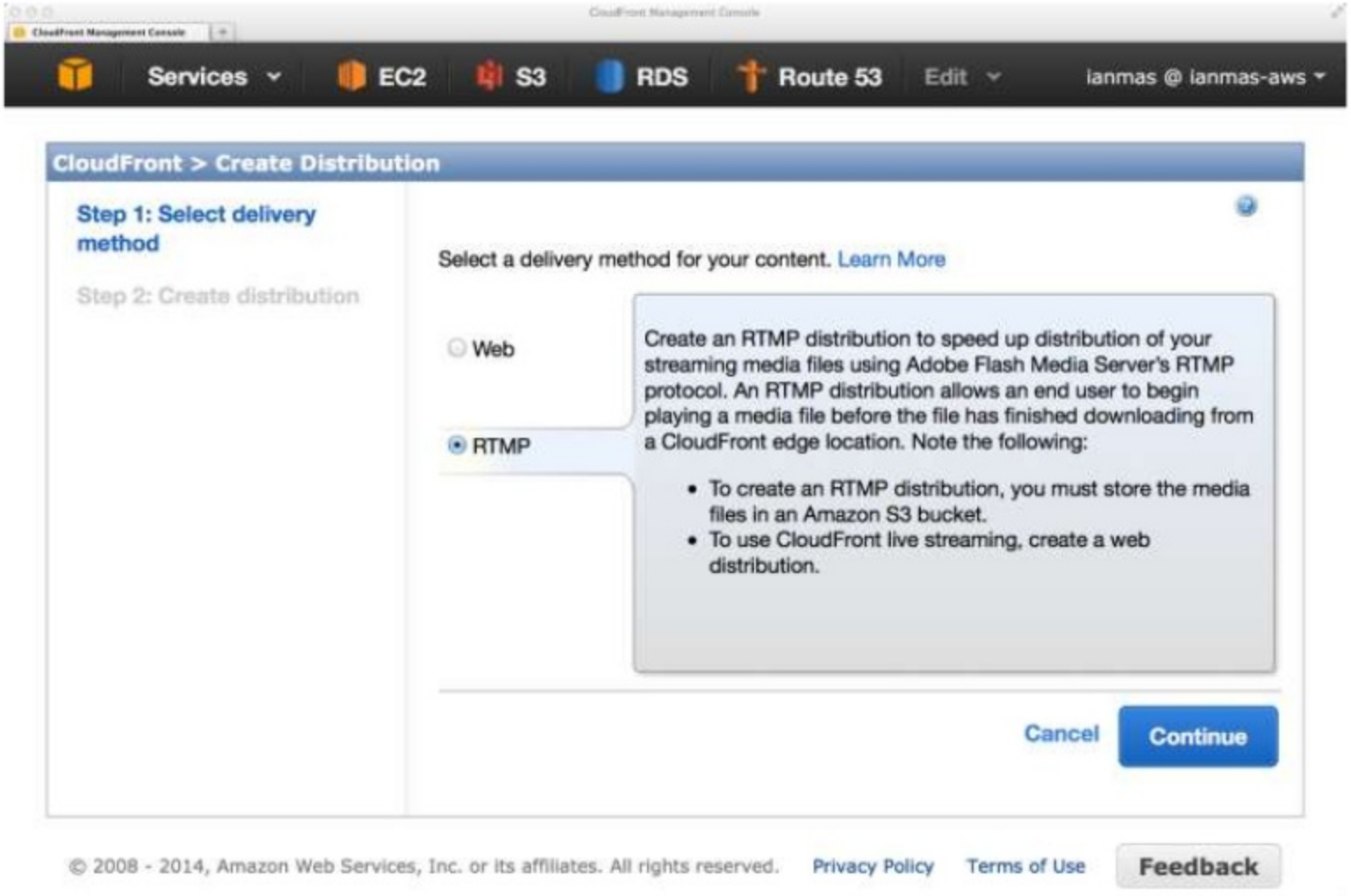
Cache control

Forwarding

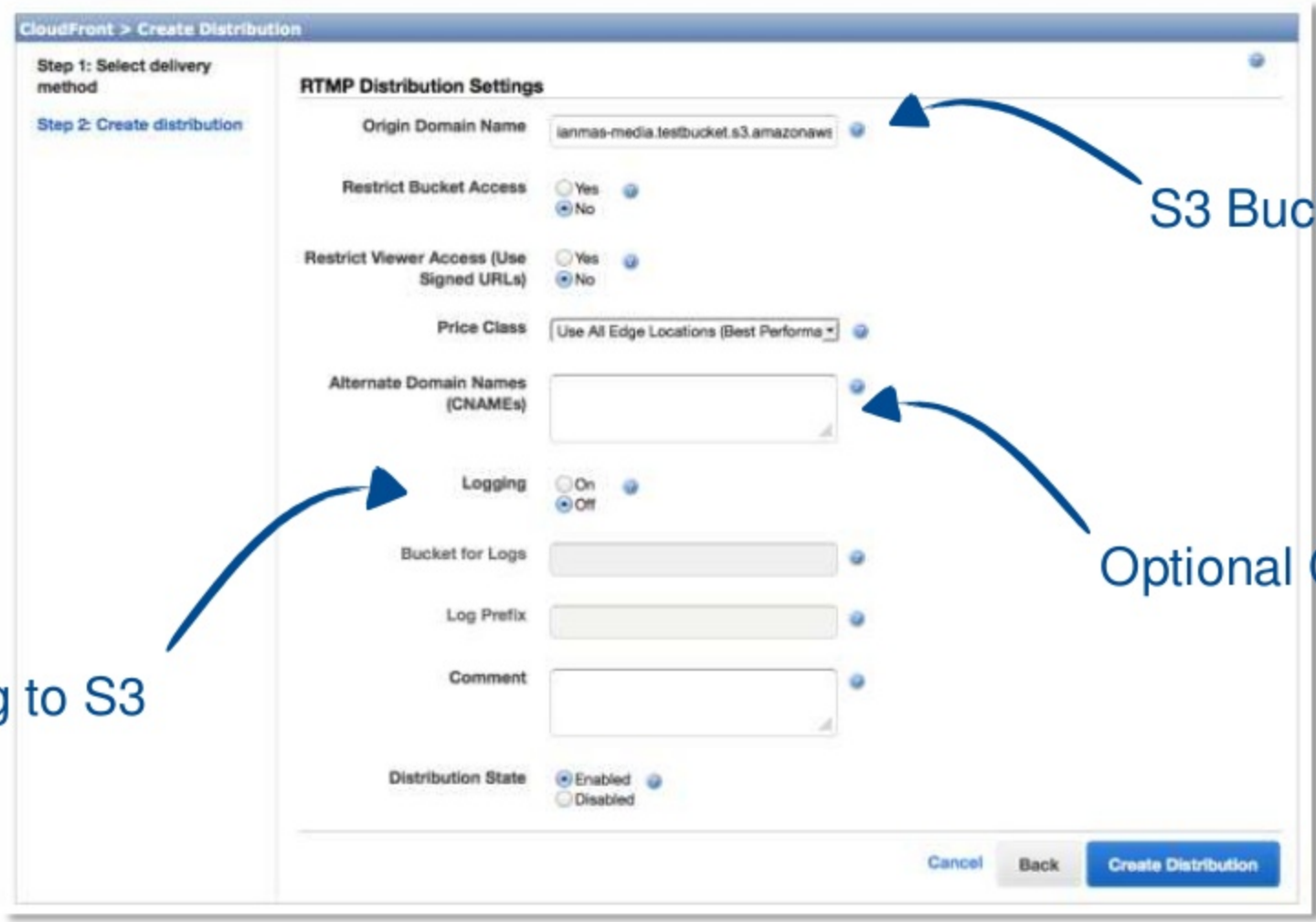
Edge access controls

Optional CNAME

Console


Services
EC2
S3
RDS
Route 53
Edit
ianmas @ ianmas-aws
CloudFront > Create Distribution
Step 1: Select delivery method
Step 2: Create distribution
Select a delivery method for your content. Learn More
Web
RTMP
Create an RTMP distribution to speed up distribution of your streaming media files using Adobe Flash Media Server's RTMP protocol. An RTMP distribution allows an end user to begin playing a media file before the file has finished downloading from a CloudFront edge location. Note the following:
To create an RTMP distribution, you must store the media files in an Amazon S3 bucket.
To use CloudFront live streaming, create a web distribution.
Cancel
Continue
© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Feedback

Console


CloudFront > Create Distribution
Step 1: Select delivery method
Step 2: Create distribution
RTMP Distribution Settings
Origin Domain Name
ianmas-media.testbucket.s3.amazonaws
Restrict Bucket Access
Yes
No
Restrict Viewer Access (Use Signed URLs)
Yes
No
Price Class
Use All Edge Locations (Best Performa
Alternate Domain Names (CNAMEs)
Logging
On
Off
Bucket for Logs
Log Prefix
Comment
Distribution State
Enabled
Disabled
Cancel
Back
Create Distribution
S3 Bucket
Optional CNAME
Logging to S3

Example

Download distribution

```
<html>
<script type='text/javascript'
src='http://d2ew7gdzogp20x.cloudfront.net/jwplayer/jwplayer.js'></script>

<body>
<div id='player'></div>
<script type='text/javascript'>
  jwplayer('player').setup({
    file: "rtmp://s1eat02wfxn38u.cloudfront.net/cfx/st/montage-medium.mp4",
    width: "480",
    height: "270",
  });
</script>
</body>
</html>
```

Streaming distribution

Summary

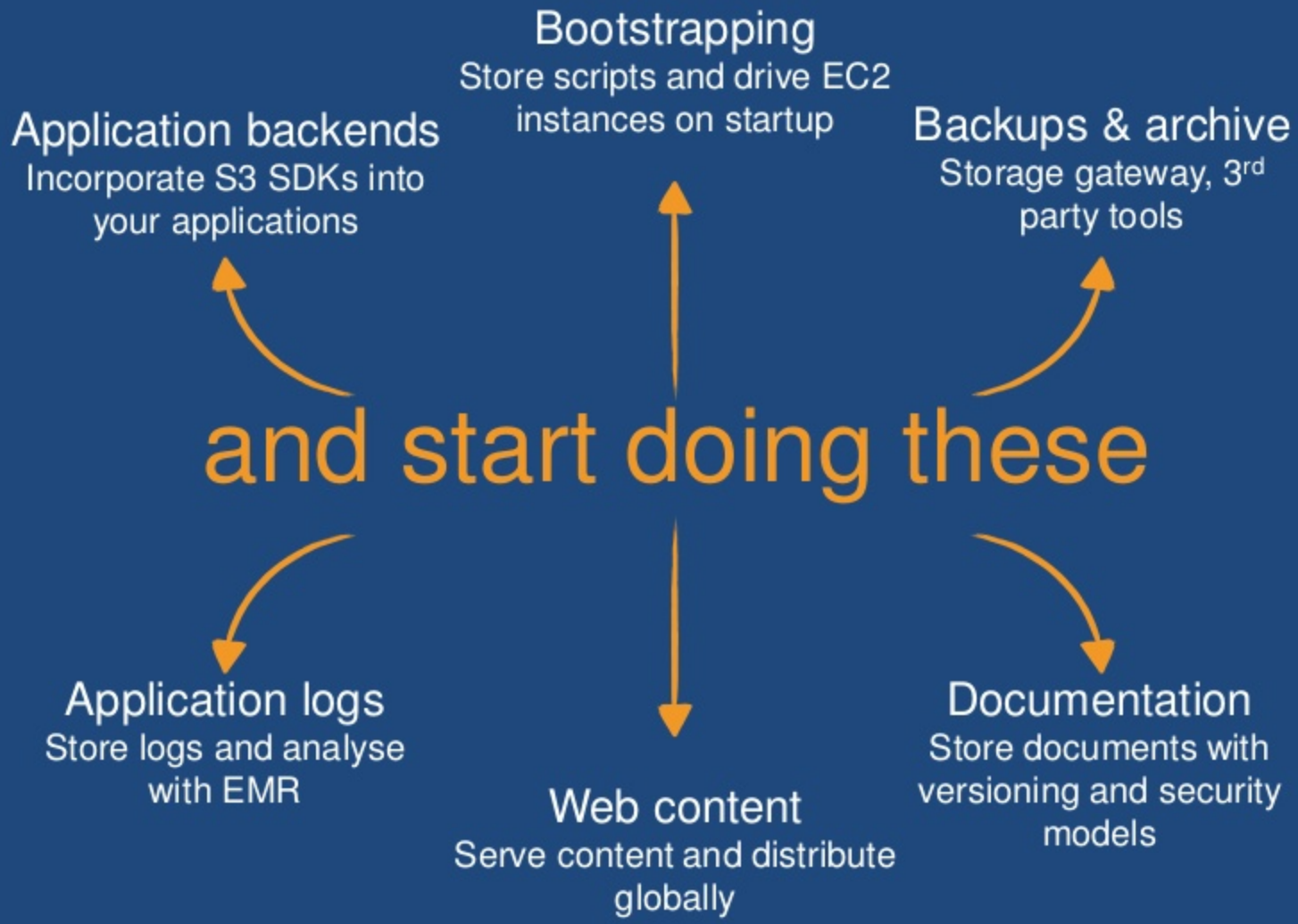
Stop doing these:

Capacity planning

Management of storage

Worrying about backing up the backup

Fixing broken hardware


Bootstrapping
Store scripts and drive EC2 instances on startup
Application backends
Incorporate S3 SDKs into your applications
Backups & archive
Storage gateway, 3rd party tools
and start doing these
Application logs
Store logs and analyse with EMR
Web content
Serve content and distribute globally
Documentation
Store documents with versioning and security models

aws.amazon.com

Code samples from this Masterclass: github.com/ianmas-aws/s3-masterclass

Follow us for more events & webinars

Ian Massingham – Technical Evangelist

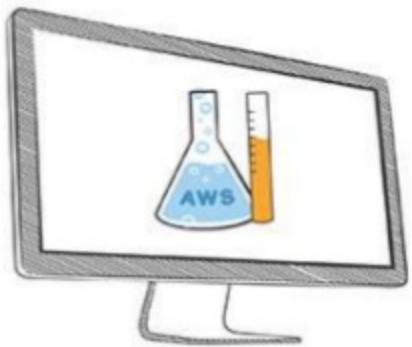
@IanMmmm

@AWS_UKI for local AWS events & news

@AWScloud for Global AWS News and Announcements

AWS Training & Certification

Self-Paced Labs



Try products, gain new skills, and get hands-on practice working with AWS technologies

aws.amazon.com/training/self-paced-labs

Training

Skill up and gain confidence to design, develop, deploy and manage your applications on AWS

aws.amazon.com/training

Certification

Demonstrate your skills, knowledge, and expertise with the AWS platform

aws.amazon.com/certification

amazon web services

We typically see customers start by trying our services

Get started now at : aws.amazon.com/getting-started

amazon web services

Design your application for the AWS Cloud

AWS Reference Architectures

The flexibility of AWS allows you to design your application architectures the way you like. AWS Reference Architecture Datasheets provide you with the architectural guidance you need in order to build an application that takes full advantage of the AWS cloud. Each datasheet includes a visual representation of the architecture and basic description of how each service is used.

Web Application Hosting
Build highly-scalable and reliable web or mobile-web applications (PDF)

Content and Media Serving
Build highly reliable systems that serve massive amounts of content and media (PDF)

Batch Processing
Build auto-scalable batch processing systems like video processing pipelines (PDF)

Fault tolerance and High Availability
Build systems that quickly failover to new instances in an event of failure (PDF)

Large Scale Processing and Huge Data sets
Build high-performance computing systems that involve Big Data (PDF)

Ad Serving
Build highly-scalable online ad serving solutions (PDF)

Disaster Recovery for Local Applications
Build cost-effective Disaster Recovery solutions for on-premises applications (PDF)

File Synchronization
Build simple file synchronization service (PDF)

More details on the AWS Architecture Center at : aws.amazon.com/architecture

amazon web services