

COMSE6998: Modern Serverless Cloud Applications

Lecture 7: S3 Websites, Queueing/SQS, Content Management Systems (Intro)

Dr. Donald F. Ferguson

Donald.F.Ferguson@gmail.com

Contents

- Introduction
 - Homework status, and some questions.
 - Q&A
- New technical topics:
 - Building a Web Site – Serverless
 - Messaging
 - Asynchronous Operations
 - Message Driven Processing/Queueing
 - Amazon SQS Overview
 - Content/Asset Management (Introduction)
- Putting some pieces together.

Introduction

New Technical Topics

Static Website

Web Application Basic Concepts

1. User performs an action that requires data from a database to be displayed.



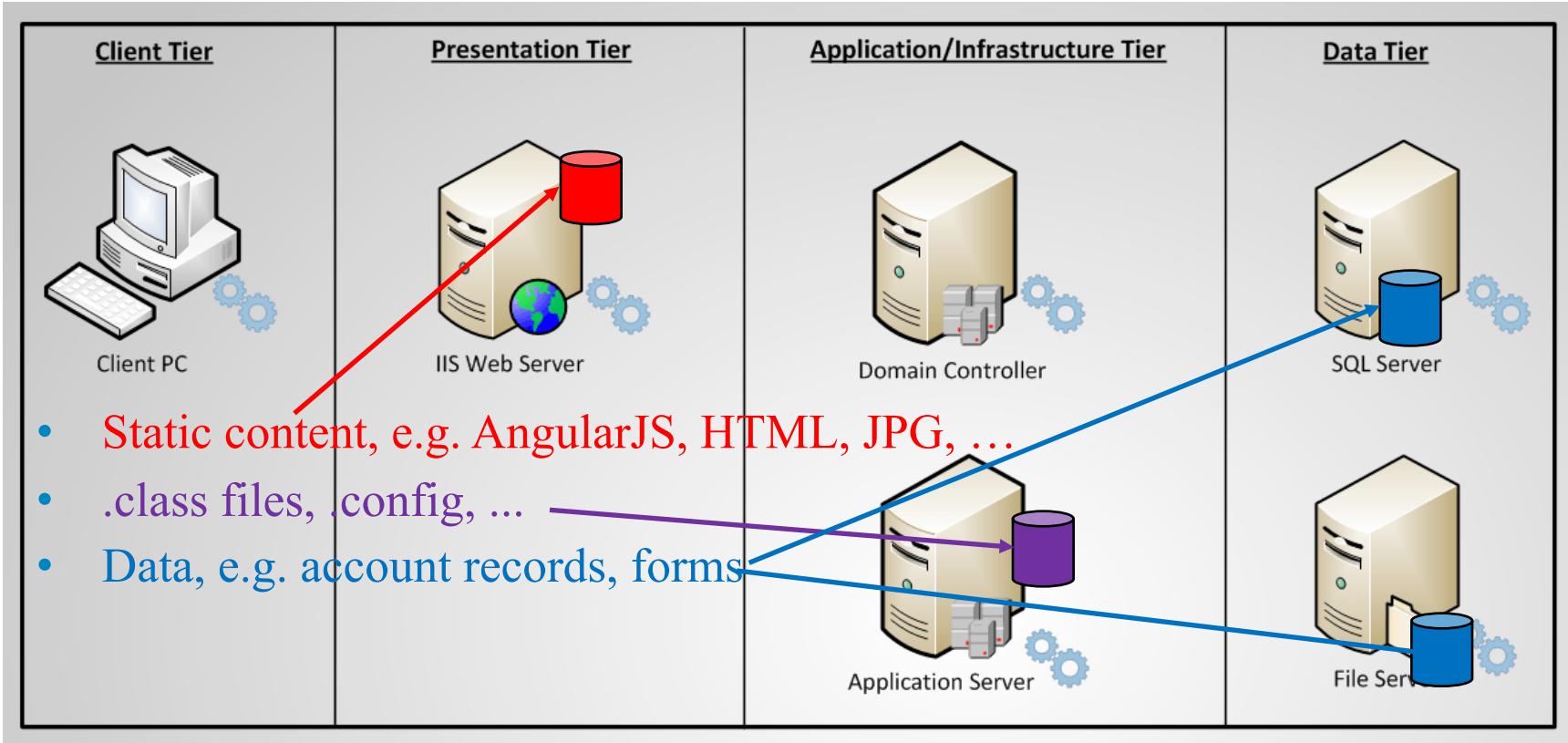
Application User

Web Client
(Presentation Tier)

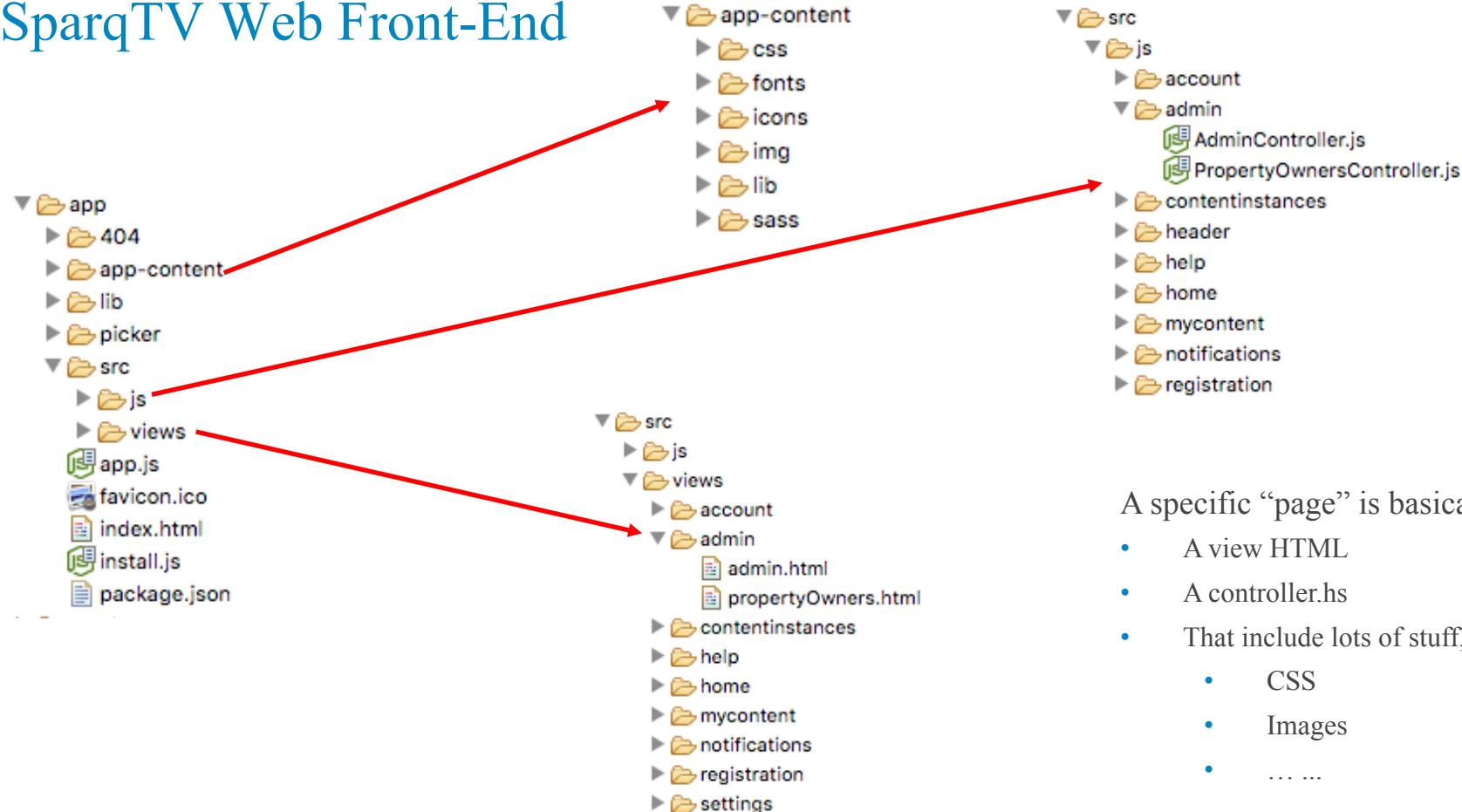
Web Server
(Application Tier)

Database
(Data Tier)

Physical Topology



SparqTV Web Front-End



A specific “page” is basically

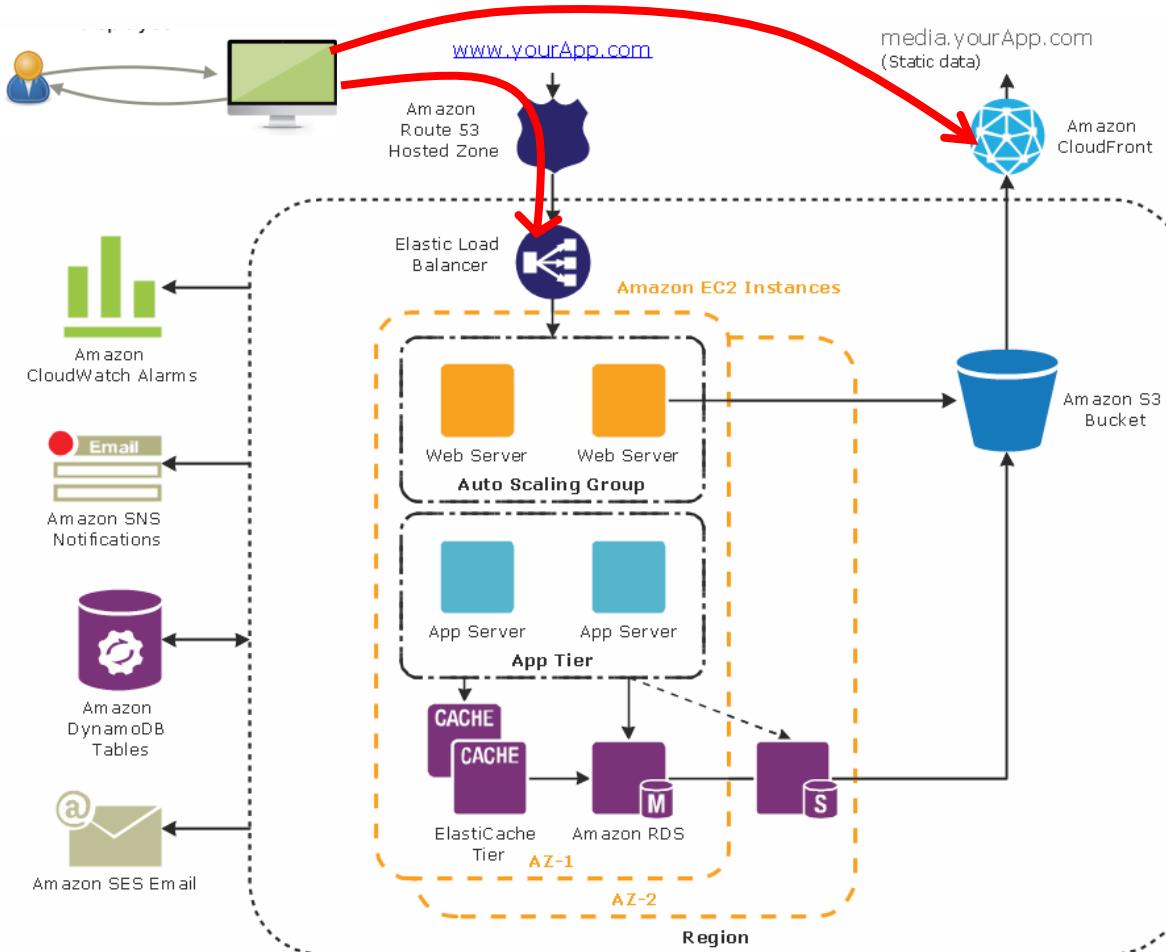
- A view HTML
- A controller.js
- That include lots of stuff,
 - CSS
 - Images
 -

Amazon S3 Website Steps

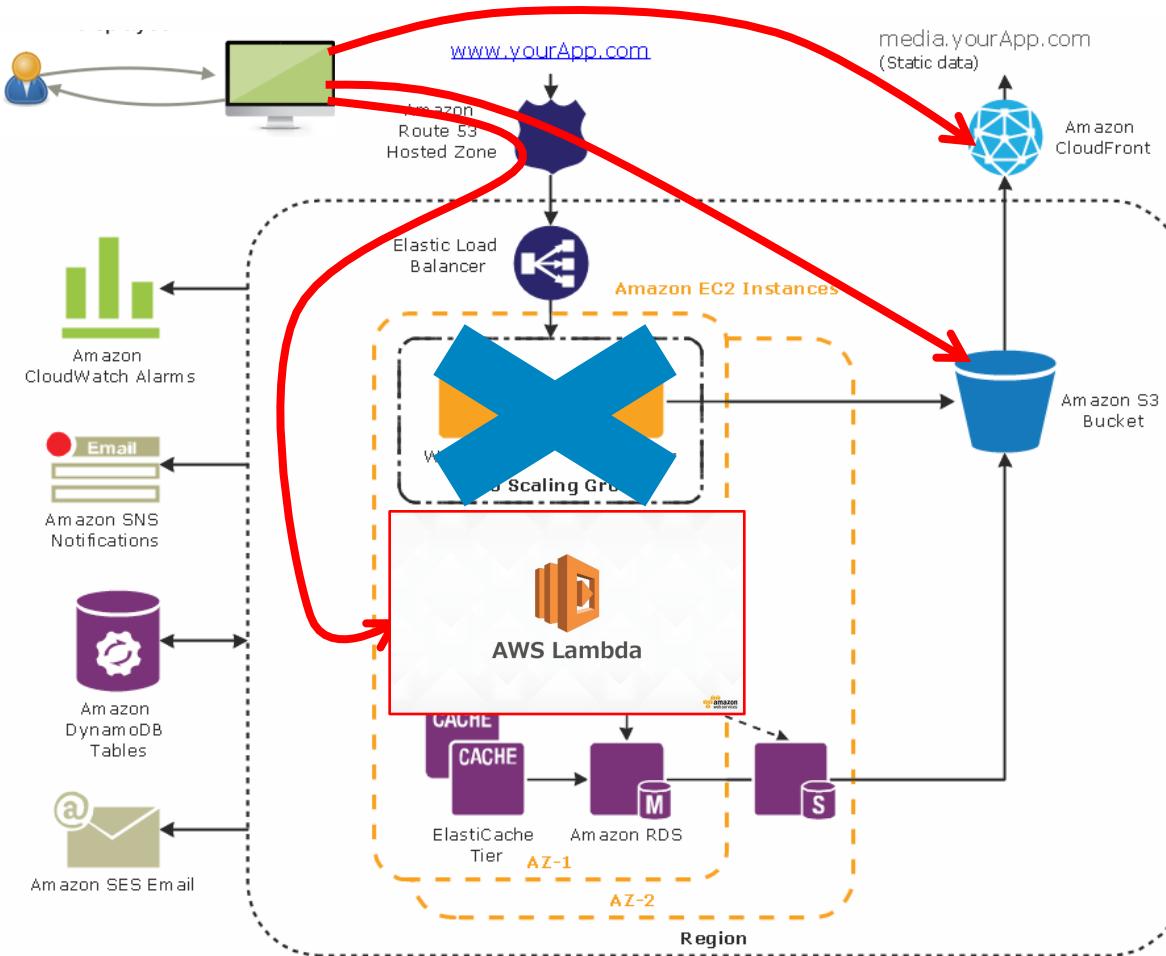
<http://docs.aws.amazon.com/gettingstarted/latest/swh/setting-up.html>

1. Setting Up: (Account, Permissions, etc).
2. Create the Buckets for Your Website
3. Configure Your Buckets
4. Deploy Your Website
- ~~5. (Optional) Register Your Domain Name~~
- ~~6. (Optional) Associate a Domain Name with Your Website~~
- ~~7. (Optional) Speed Up Your Website~~

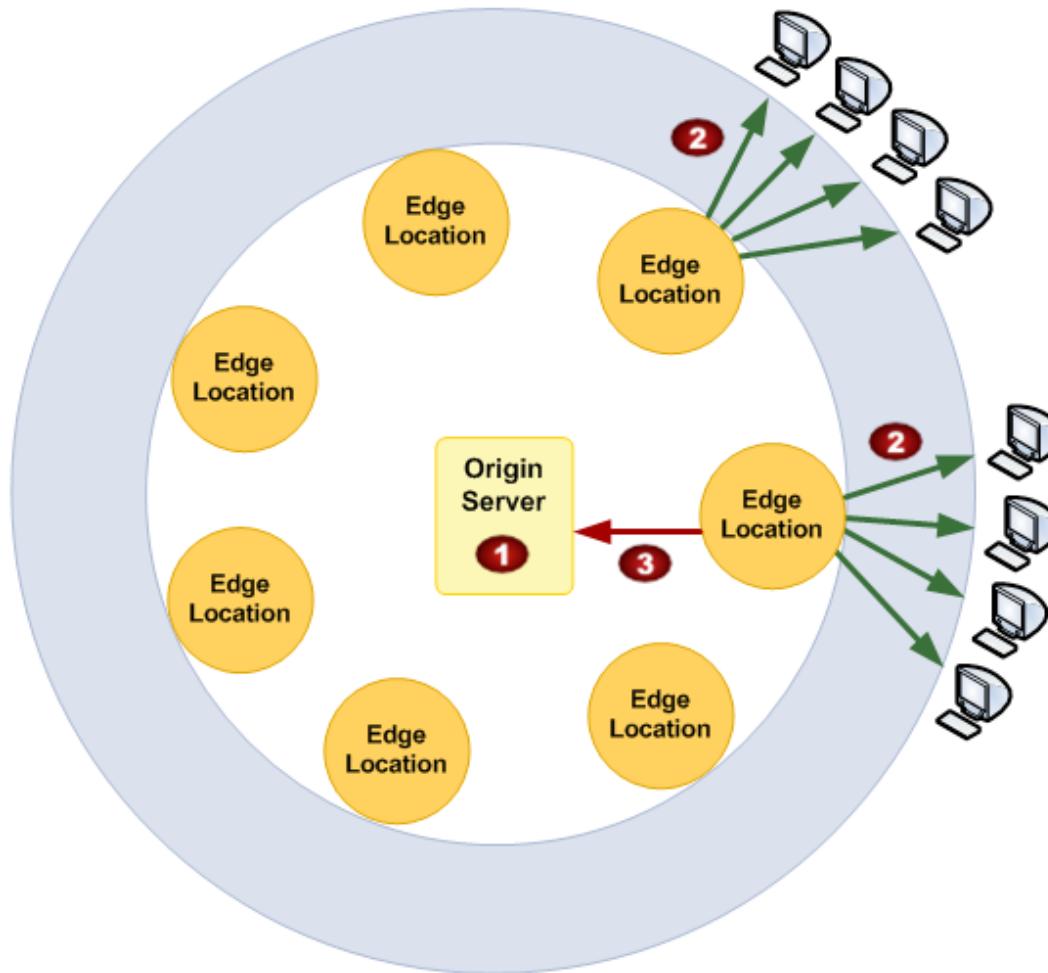
“Traditional” AWS Topology



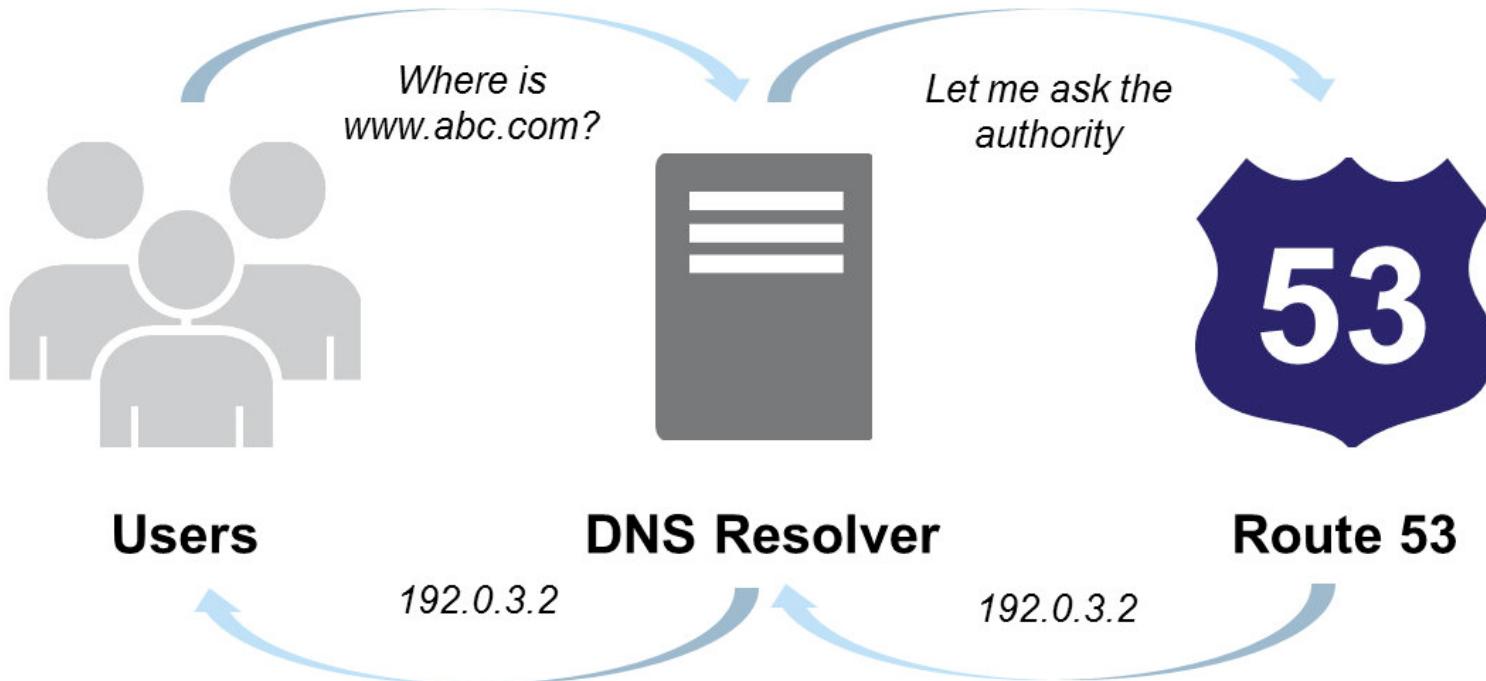
Our Model



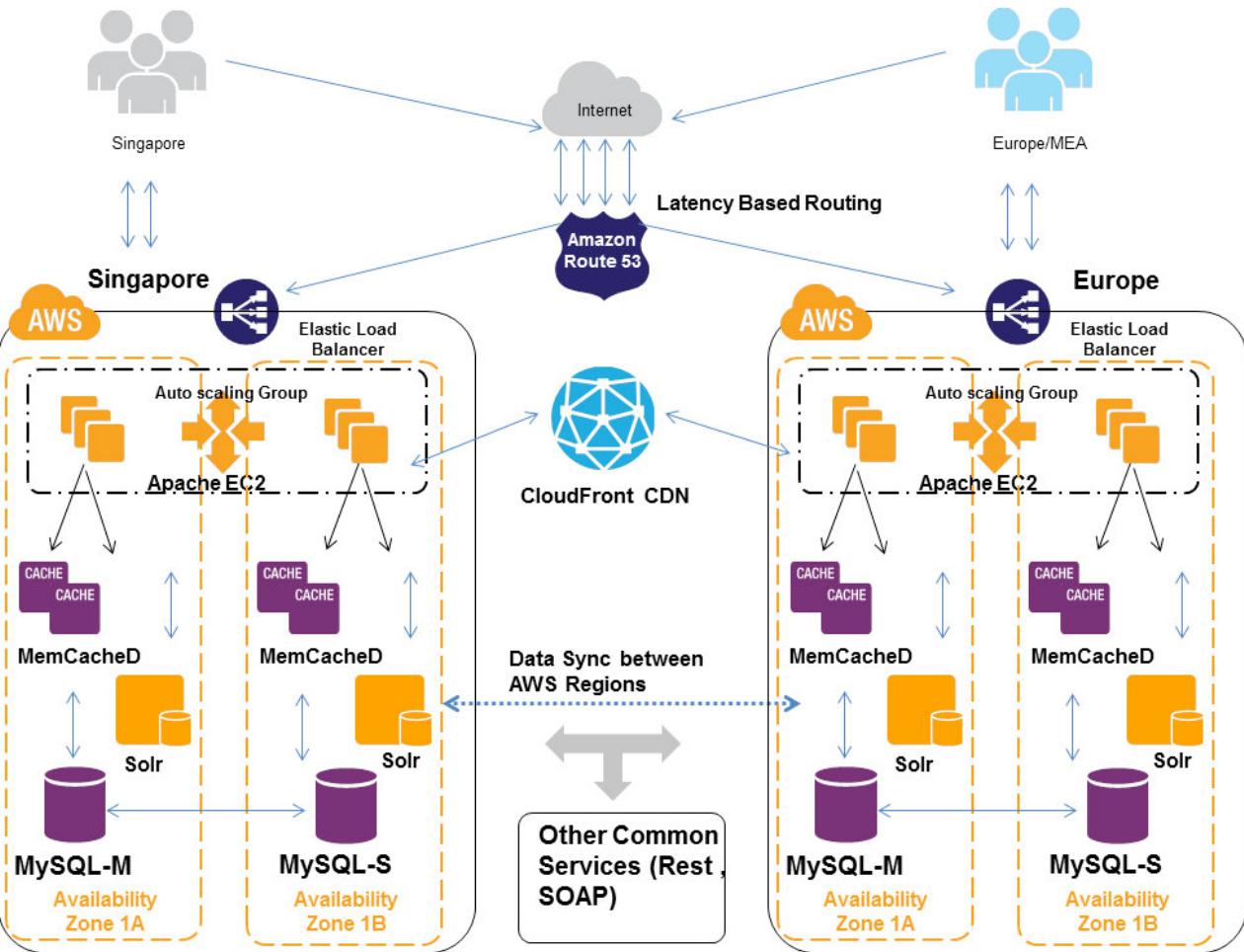
AWS CloudFront



AWS Route 53

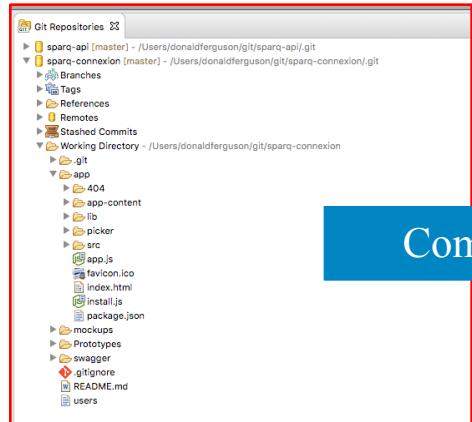


Example



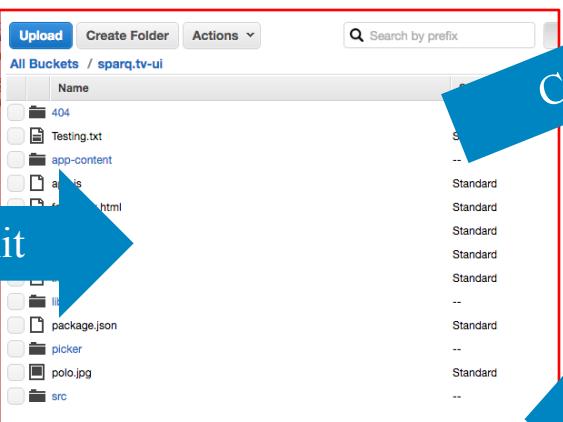
Putting it Together

Eclipse (Node)



Commit

S3



Config

CloudFront Distributions

[Create Distribution](#) [Distribution Settings](#) [Delete](#) [Enable](#) [Disable](#)

Viewing :	Any Delivery Method	Any State	
Delivery Method	ID	Domain Name	
Web	E11JPZG9QK06XL	d1qg8h4l6x6ma7.cloudfront.net	
Web	E1VOXR00UTSTW6	d2hjlc013sr2bd.cloudfront.net	
<input type="checkbox"/> Web	E3J0JTRHR300ZS	d17lrqjhtsop1j.cloudfront.net	
<input type="checkbox"/> Web	EKBL9QAD2W9VF	d2kovksorsxicly.cloudfront.net	

Route 53

Config

- Automated as part of DevOps/CI
 - For projects, you have very little change and few pages and can do manually.

Demo

The screenshot shows the AWS S3 console at <https://console.aws.amazon.com/s3/home?region=us-east-1#>. The top navigation bar includes links for AWS, Services, Edit, and a user account (donald.f.ferguson@gmail.com). Below the navigation is a toolbar with 'Create Bucket' (highlighted in blue), 'Actions' (with dropdown options 'None', 'Properties', and 'Transfers'), and a search bar.

The main area displays a table titled 'All Buckets (9)' with a column header 'Name'. The listed buckets are:

- lambda-function-bucket-us-east-1-1457105811684
- lambda-function-bucket-us-west-2-1456939763324
- seekatv.content
- sparq-authentication-swagger
- sparq-connexion
- sparq-warehouse
- sparq.tv-ui
- sparq.tv.dev
- sparq.tv.test

On the right side of the screen, there are two promotional banners:

- S3 Console**: Click the "Feedback" button at the bottom of the page to send us your comments.
- Announcement: Load data up to 300% faster with S3 Transfer Acceleration.**: Test it here.

Demo

The screenshot shows the AWS S3 console interface. The left pane displays a list of objects in the 'sparq.tv.test' bucket:

	Name	Storage Class	Size	Last Modified
<input type="checkbox"/>	404	--	--	--
<input type="checkbox"/>	app-content	--	--	--
<input type="checkbox"/>	app.js	Standard	9.2 KB	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	favicon.ico	Standard	1.1 KB	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	index.html	Standard	5.2 KB	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	install.js	Standard	1.5 KB	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	lib	--	--	--
<input type="checkbox"/>	package.json	Standard	855 bytes	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	picker	--	--	--
<input type="checkbox"/>	src	--	--	--

The right pane shows the properties of the 'sparq.tv.test' bucket:

Bucket: sparq.tv.test

- Bucket: sparq.tv.test
- Region: US Standard
- Creation Date: Tue Sep 27 09:43:34 GMT-400 2016
- Owner: ec2-sparq

Navigation links for bucket management:

- Permissions
- Static Website Hosting
- Logging
- Events
- Versioning
- Lifecycle
- Cross-Region Replication
- Tags
- Requester Pays
- Transfer Acceleration

Demo

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with 'AWS', 'Services', 'Edit', and user information ('donald.f.ferguson@gmail.com ...'). Below the navigation is a toolbar with 'Upload', 'Create Folder', 'Actions', a search bar ('Search by prefix'), and buttons for 'None', 'Properties', and 'Transfers'. A 'Close' button is also present.

The main area displays a list of objects in the 'All Buckets / sparq.tv.test' section. The objects listed are:

	Name	Storage Class	Size	Last Modified
<input type="checkbox"/>	404	--	--	--
<input type="checkbox"/>	app-content	--	--	--
<input type="checkbox"/>	app.js	Standard	9.2 KB	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	favicon.ico	Standard	1.1 KB	Tue Sep 27 10:45:02 GM
<input checked="" type="checkbox"/>	index.html	Standard	5.2 KB	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	install.js	Standard	1.5 KB	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	lib	--	--	--
<input type="checkbox"/>	package.json	Standard	855 bytes	Tue Sep 27 10:45:02 GM
<input type="checkbox"/>	picker	--	--	--
<input type="checkbox"/>	src	--	--	--

The 'index.html' object is selected, highlighted with a blue border. On the right side, a detailed view of the 'Object: index.html' is shown. The object details are as follows:

- Bucket: sparq.tv.test
- Name: index.html
- Link: (redacted)
- Size: 5354
- Last Modified: Tue Sep 27 10:45:02 GMT-400 2016
- Owner: ec2-sparq
- ETag: bde04e86e13da55b300a7534326653b8
- Expiry Date: None
- Expiration Rule: N/A

Below the object details, there are three expandable sections: 'Details', 'Permissions', and 'Metadata'. The 'Metadata' section contains the message 'Metadata is a set of name-value pairs. Learn more.' and a key-value pair entry for 'Content-Type' with a value of 'text/html'. Buttons for 'Add more metadata' and 'Remove selected metadata' are available. At the bottom right are 'Save' and 'Cancel' buttons.

Demo

The screenshot shows the AWS S3 console interface. On the left, a list of objects in the 'sparq.tv.test' bucket is displayed:

	Name	Storage Class	Size	Last Modified
	404	--	--	--
	app-content	--	--	--
	app.js	Standard	9.2 KB	Tue Sep 27 10:45:02 GM
	favicon.ico	Standard	1.1 KB	Tue Sep 27 10:45:02 GM
<input checked="" type="checkbox"/>	index.html	Standard	5.2 KB	Tue Sep 27 10:45:02 GM
	install.js	Standard	1.5 KB	Tue Sep 27 10:45:02 GM
	lib	--	--	--
	package.json	Standard	855 bytes	Tue Sep 27 10:45:02 GM
	picker	--	--	--
	src	--	--	--

A red arrow points from the text "ETag will become important in future lectures." to the ETag value in the object details panel.

Object: index.html

Bucket: sparq.tv.test
Name: index.html
Link: [\[REDACTED\]](#)
Size: 5354
Last Modified: Tue Sep 27 10:45:02 GMT-400 2016
ETag: bde04e86e13da55b300a7534326653b8
Expires Date: None
Expires On Rule: N/A

> Details
> Permissions
▼ Metadata

Metadata is a set of name-value pairs. [Learn more.](#)

Key: Content-Type Value: text/html

Add more metadata Remove selected metadata

Save Cancel

ETag will become important in future lectures.

Project Implications

- Your web site is relatively static and relatively simple. Something like ...

- /content

- CSS files
 - Images
 - etc.

- /app

- /controllers

- Controller1.js
 - Controller2.js

- ...

- /views

- View1.html
 - View2.html

- ...

You have the following options:

- Just do it manually through the AWS console.
- Use one of the various git commit/push tools.
- Use s3 sync command line.

(<http://docs.aws.amazon.com/cli/latest/userguide/using-s3-commands.html>)

The sync command has the following form. Possible source-target combinations are:

- Local file system to Amazon S3
- Amazon S3 to local file system
- Amazon S3 to Amazon S3

```
$ aws s3 sync <source> <target> [--options]
```

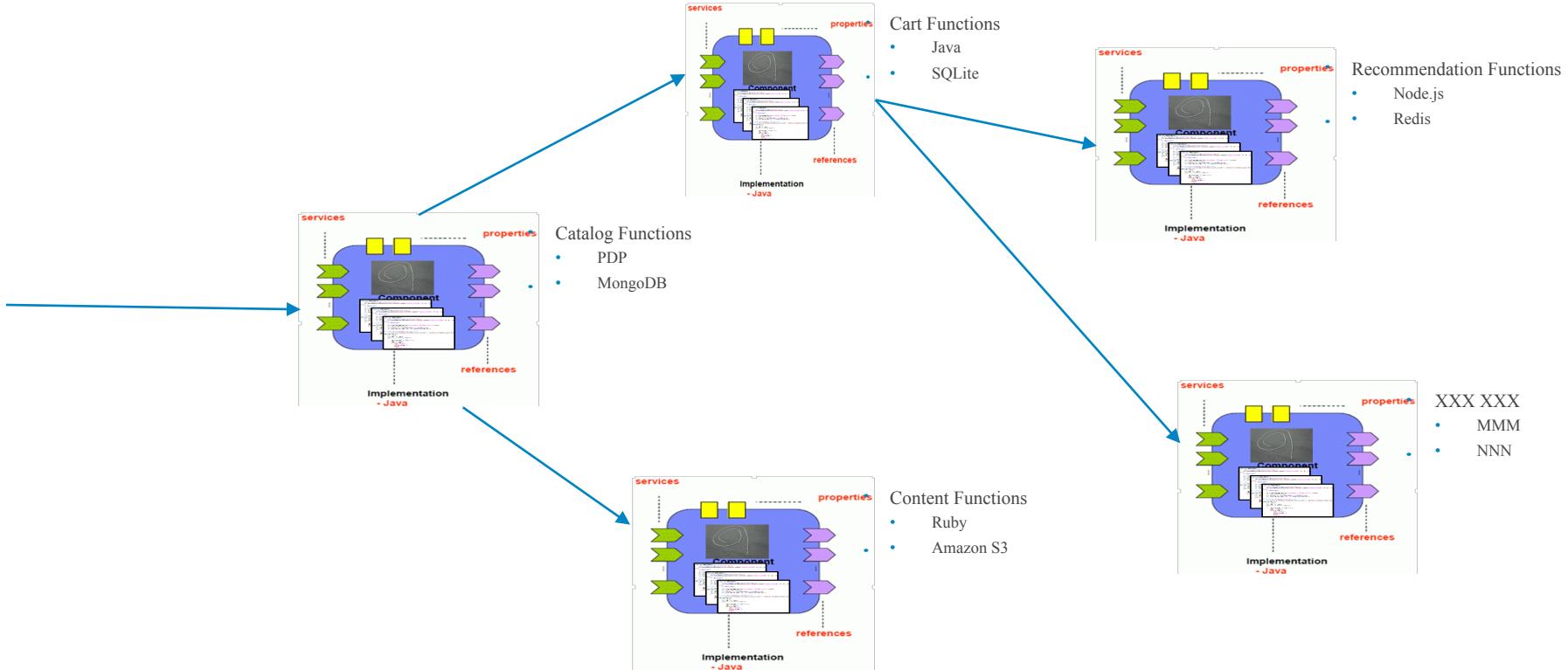
The following example synchronizes the contents of an Amazon S3 folder named *path* in *my-bucket* with the current working directory. s3 sync updates any files that have a different size or modified time than files with the same name at the destination. The output displays specific operations performed during the sync. Notice that the operation recursively synchronizes the subdirectory *MySubdirectory* and its contents with *s3://my-bucket/path/MySubdirectory*.

```
$ aws s3 sync . s3://my-bucket/path
upload: MySubdirectory\MyFile3.txt to s3://my-bucket/path/MySubdirectory/MyFile3.txt
upload: MyFile2.txt to s3://my-bucket/path/MyFile2.txt
upload: MyFile1.txt to s3://my-bucket/path/MyFile1.txt
```

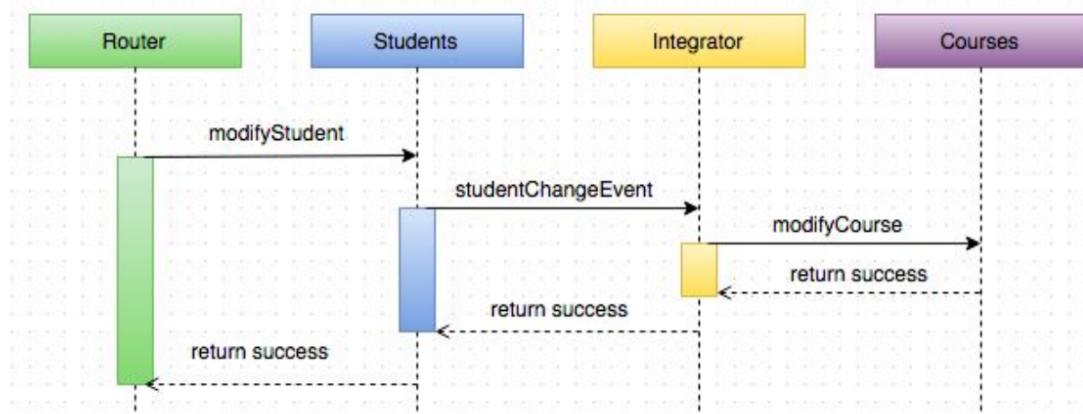
Asynchronous Operations and Message Queues

Asynchronous Operations

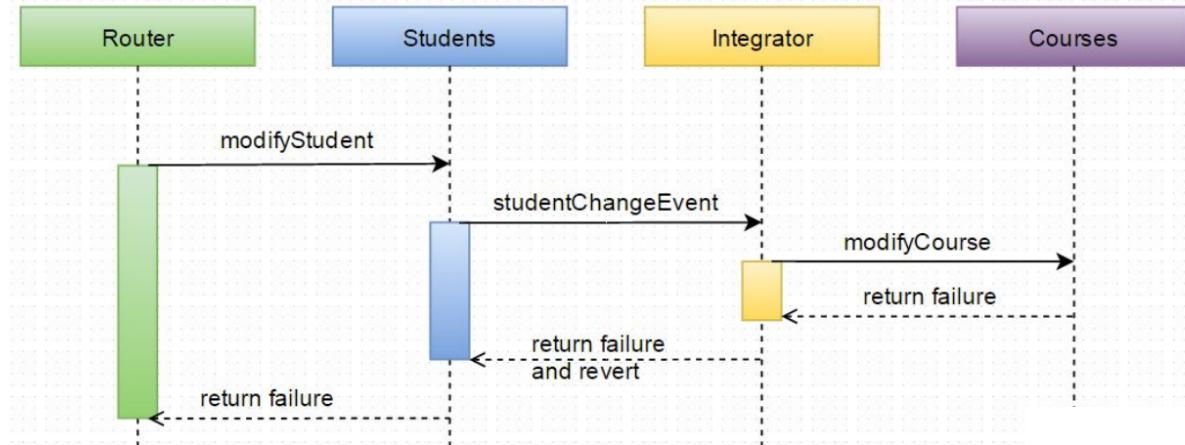
REST Implementations Call REST Services



Synchronous Integration



However, if that request fails when updating a course in Courses, the sequence looks as follows:



Synchronous and Asynchronous

- HTTP operations are inherently *synchronous*
 - Connect
 - Verb
 - Response
 - Disconnect
- The implementation of the operation may “take a while,”
 - A calls B before returning. B calls C and D before returning to A. ...
 - The implementation may be computationally or data intensive, e.g. DB query, image transcoding, ...
 - Something in the implementation may operate at “people speed,” e.g.
 - Creating an Account may require.
 - Manual approval by “Dave.”
- Holding open connections can cause problems
 - A server typically has an upper limit on open connections. Holding a connection for long requests will cause some client calls to “get a busy signal.”
 - A connection is more likely to break the longer it is open. A call graph of “long” connections is inherently fragile. One breaks and the whole thing breaks.

Asynchronous Operation

Request:

This is going to “take a while.”

```
POST /blogs HTTP/1.1  
<xml>  
  blogdata  
</xml>
```

Response:

```
HTTP/1.1 202 Accepted  
Location: /queue/12345
```

So, I am going to

- Tell you that your request looks OK.
- Acknowledge that I am working on it.
- Give you a URL on which you can poll with GET for the answer

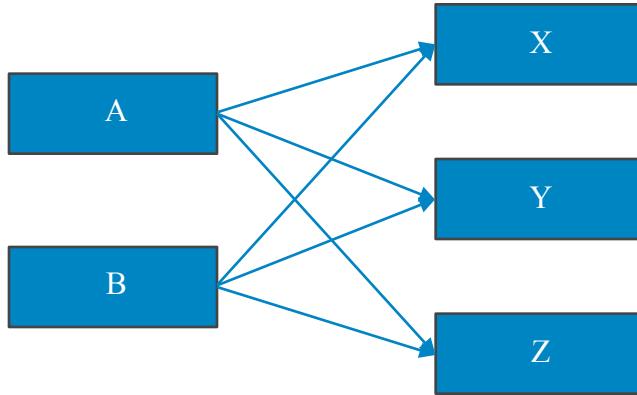
Basic Implementation Approach

- Define a *collection* /QueuedResponses
 - A client can call .../QueuedResponses/21 to get a specific response.
 - You already know how to do this for .../customer, .../address
 - The data format in the table is {id, status, JSONString}, where
 - JSONString is the response you would have received for a synchronous request.
 - Status is one of “in-progress” or “complete.”
- A simple implementation would be writing a *façade*
 - Accept request
 - Create new table entry with status = “in progress”
 - Return 202 and URL
 - Call the *actual* implementation
 - Update the database table entry with the JSON result
- Most application platforms have middleware approaches to support registering callbacks, threads, etc. The implementation would typically
 - Invoke some long running action, e.g. DB query, workflow process and register a callback
 - The callback implementation updates the entry in the response table.

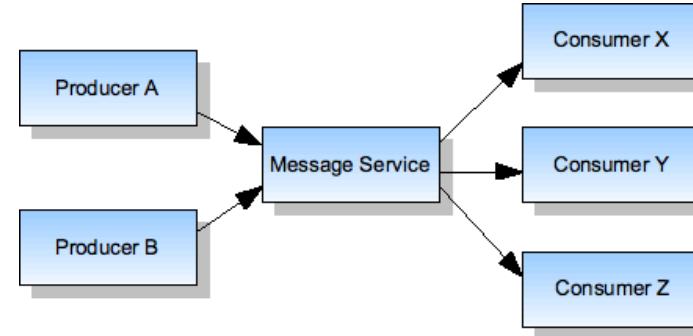
Message Driven Processing

Message Queue Service Pattern

Anti-Pattern



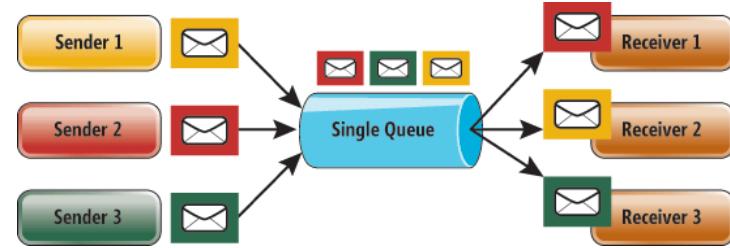
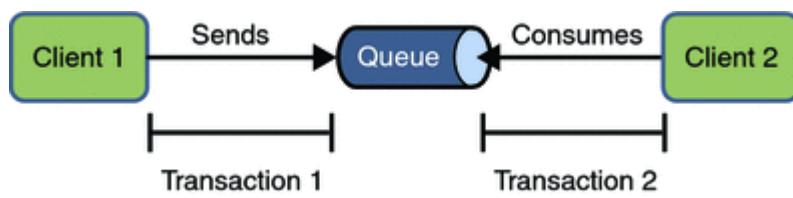
Best Practice Pattern



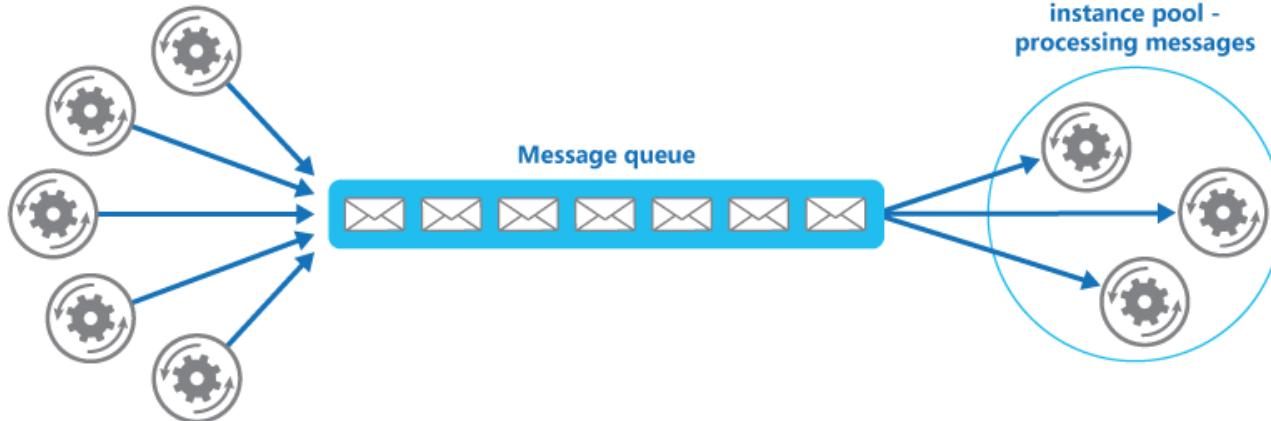
Point-to-Point communication between modules becomes fragile at scale

- Adding modules P and Q requires finding and configuring all senders.
- I want to send M to any one of X1, X2 and X3, but only one.
- I want to make sure that someone processes M but do not want to hold the transaction until I get a response (I may not even need the response).
- My destinations are not “up” at the same times I am.

Message Exchange Patterns



Application instances - generating messages



Why Would I Use a Queue

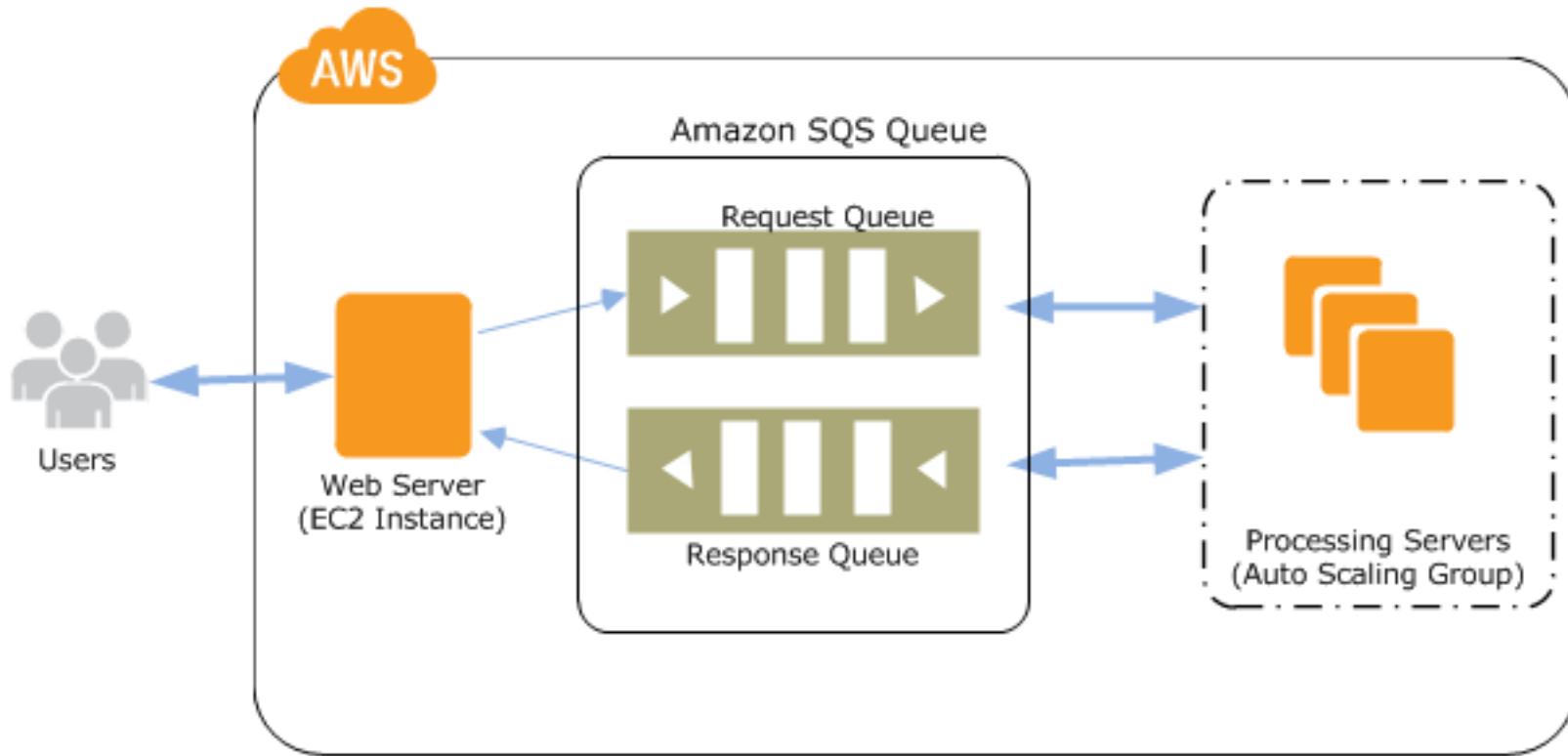
<http://www.javacodegeeks.com/2013/06/working-with-amazon-simple-queue-service-using-java.html> is one example)

- Flow control
 - My server may be able to process 100 reqs/second
 - There are thousands of clients
 - The aggregate rate could be 10 req/s or 300 req/s
 - Without a queue, clients would get connect failures under load
- The request is going to take a long time
 - Image reformatting and tagging
 - I cannot have my code “wait” for a response
 - And I do not want to write a lot of “Is it done yet?” calls
- I do not know (or care) which one implements the logic
 - I have 5 – 10 image processors for GIFs
 - I have 7 – 10 processors for .wav files
 - I do not want to know which ones are active, when and processing what.
 - I just want to put the “thing in a queue” knowing that someone will eventually pick it up.
- Think “email” for API calls

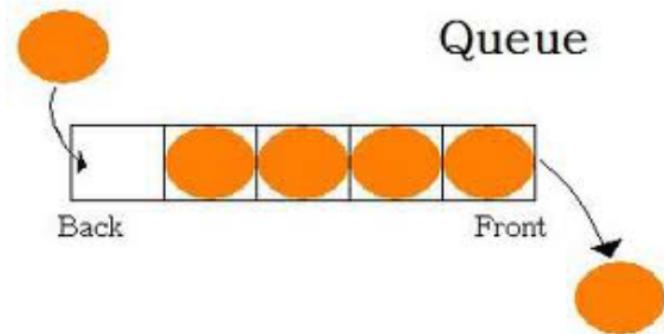
Many use cases
and patterns for
Message driven processing.

Amazon SQS

Amazon SQS – Example



Queue?



Queue Operations

- Create Q
- Delete Q
- Send Message
- Receive Message
- Delete Message

FIFO Semantics

- ~~Strict, limited scalability.~~
- Best effort, highly scalable.
- ~~Transactional~~

What is SQS?

- Amazon **Simple Queue Service** is a **fast, reliable, scalable and fully managed** queue service
- Amazon SQS enables **asynchronous message-based communication** between **distributed components** of an application

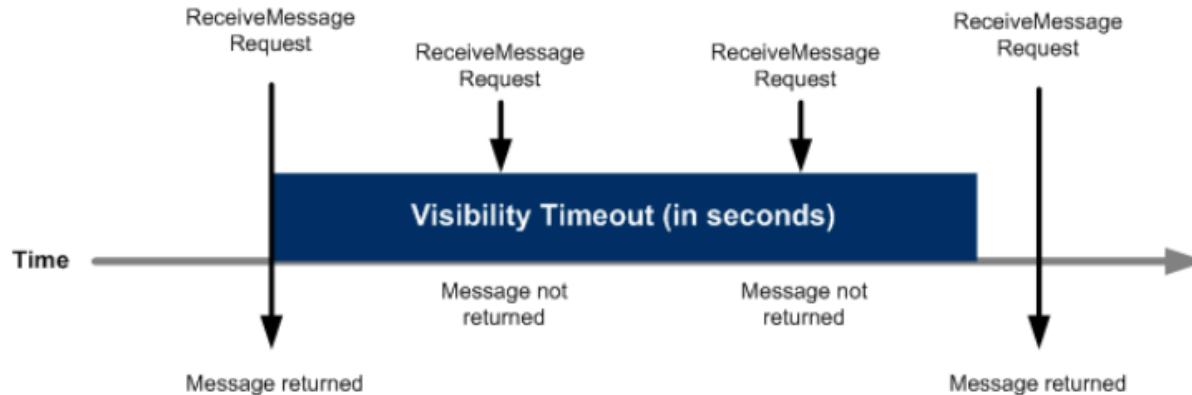
SQS – What it can do?

- Amazon AWS SQS page:
 - SQS makes it simple and cost-effective to *decouple the components* of a cloud application
 - You can use SQS to transmit any volume of data, at any level of throughput, *without losing messages* or requiring other services to be always available
 - With SQS, you can offload the administrative burden of *operating and scaling* a highly available messaging cluster, while paying a low price for only what you use.

SQS – Major Features

- Redundant Infrastructure
- Multiple writers and readers
- Configurable settings per queue
- Variable message size (up to 256 KB)
- Access Control
- Delay Queues

SQS – Message Visibility Timeout



SQS – Message Visibility Timeout

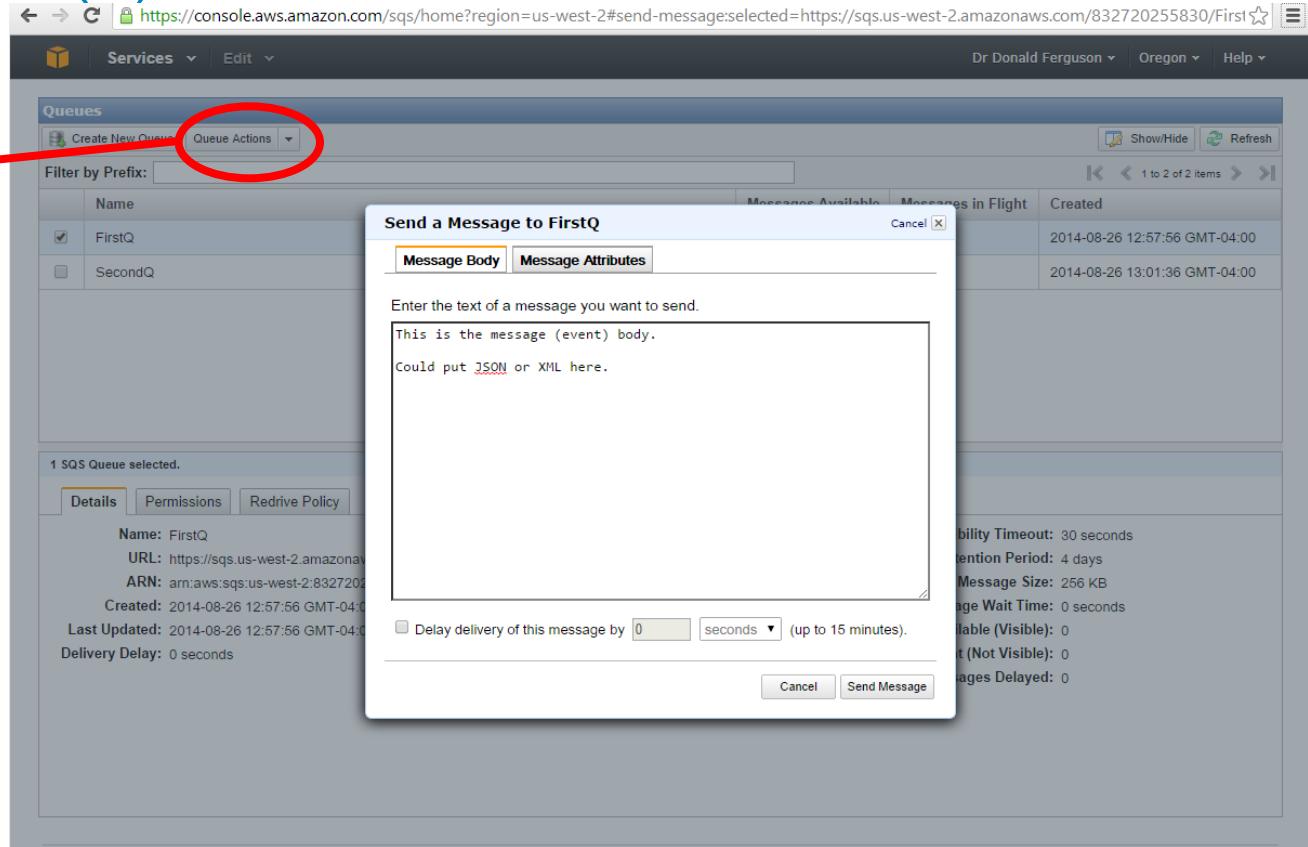
- The visibility timeout clock starts ticking once Amazon SQS returns the message
- During that time, the component processes and deletes the message
- But what happens if the component fails before deleting the message?
- If your system doesn't call *DeleteMessage* for that message before the visibility timeout expires, the message again becomes visible to the *ReceiveMessage* calls placed by the components in your system and it will be received again
- If a message should only be received once, your system should delete it within the duration of the visibility timeout
- Each queue starts with a default setting of 30 seconds for the visibility timeout

SQS Message Lifecycle

- Messages that are stored in Amazon SQS have a lifecycle that is easy to manage but ensures that all messages are processed.
 1. A system that needs to send a message will find an Amazon SQS queue, and use *SendMessage* to add a new message to it
 2. A different system that processes messages needs more messages to process, so it calls *ReceiveMessage*, and this message is returned
 3. Once a message has been returned by *ReceiveMessage*, it will **not** be returned by any other *ReceiveMessage* until the **visibility timeout** has passed. This keeps multiple computers from processing the same message at once.
 4. If the system that processes messages successfully finishes working with this message, it calls *DeleteMessage*, which removes the message from the queue so no one else will ever process it. If this system fails to process the message, then it will be read by another *ReceiveMessage* call as soon as the **visibility timeout** passes

Amazon SQS (I)

Selected
“Send Msg”



Amazon SQS (I)

The screenshot shows the AWS SQS console interface. On the left, the 'Queues' list shows two queues: 'FirstQ' (selected) and 'SecondQ'. The 'Details' tab for 'FirstQ' is open, displaying its configuration: Name: FirstQ, URL: https://sqs.us-west-2.amazonaws.com/832720255830/FirstQ, ARN: arn:aws:sqs:us-west-2:832720255830:FirstQ, Created: 2014-08-26 12:57:56 GMT-04:00, Last Updated: 2014-08-26 12:57:56 GMT-04:00, and Delivery Delay: 0 seconds. In the center, a modal dialog titled 'Send a Message to FirstQ' is open, showing the 'Message Attributes' tab. It contains two attributes: 'Mood' (String type, value 'Cranky') and 'ProfessorIQ' (Number type, value '11'). At the bottom of the dialog are 'Cancel' and 'Send Message' buttons. The background shows the 'Messages in Queue' table with two items listed.

Name	Type	Values
Mood	String	Cranky
ProfessorIQ	Number	11

SQS API

(AWS.Request) `addPermission(params = {}, callback)`

Adds a permission to a queue for a specific **principal**.

(AWS.Request) `changeMessageVisibility(params = {}, callback)`

Changes the visibility timeout of a specified message in a queue to a new value.

(AWS.Request) `changeMessageVisibilityBatch(params = {}, callback)`

Changes the visibility timeout of multiple messages.

(AWS.Request) `createQueue(params = {}, callback)`

Creates a new queue, or returns the URL of an existing one.

(AWS.Request) `deleteMessage(params = {}, callback)`

Deletes the specified message from the specified queue.

(AWS.Request) `deleteMessageBatch(params = {}, callback)`

Deletes up to ten messages from the specified queue.

(AWS.Request) `deleteQueue(params = {}, callback)`

Deletes the queue specified by the **queue URL**, regardless of whether the queue is empty.

(AWS.Request) `getQueueAttributes(params = {}, callback)`

(AWS.Request) `getQueueUrl(params = {}, callback)`

Returns the URL of an existing queue.

(AWS.Request) `listDeadLetterSourceQueues(params = {}, callback)`

Returns a list of your queues that have the `RedrivePolicy` queue attribute configured with a dead letter queue.

For more information about using dead letter queues, see [Using Amazon SQS Dead Letter Queues](#).

(AWS.Request) `listQueues(params = {}, callback)`

Returns a list of your queues.

(AWS.Request) `purgeQueue(params = {}, callback)`

Deletes the messages in a queue specified by the **queue URL**.

When you use the `PurgeQueue` API, the deleted messages in the queue cannot be retrieved.

When you purge a queue, the message deletion process takes up to 60 seconds.

(AWS.Request) `receiveMessage(params = {}, callback)`

Retrieves one or more messages, with a maximum limit of 10 messages, from the specified queue.

(AWS.Request) `removePermission(params = {}, callback)`

Revokes any permissions in the queue policy that matches the specified `Label` parameter.

(AWS.Request) `sendMessage(params = {}, callback)`

Delivers a message to the specified queue.

(AWS.Request) `sendMessageBatch(params = {}, callback)`

Delivers up to ten messages to the specified queue.

(AWS.Request) `setQueueAttributes(params = {}, callback)`

Sets the value of one or more queue attributes.

Content Management Introduction

Complex Website Catalog

The screenshot shows the Amazon.com homepage with a search query of "web service architecture ferguson" entered in the search bar. The results page features a prominent advertisement for the Fire HDX 8.9 tablet. Below the ad, there's a section for "Amazon Fashion" featuring "MUST-HAVE DENIM & MORE". The right side of the page displays several promotional banners for the Kindle, APC power strips, and a Veterans Day sale.

www.amazon.com

Donald's Amazon.com Today's Deals Gift Cards Sell Help

Shop by Department Search All web service architecture ferguson Go

9 DAYS LEFT Sponsored By Hershey's Halloween Shop >Shop now

Hello, Donald Your Account Your Prime Cart Wish List

Instant Video Digital Music Cloud Drive Fire HDX Appstore for Android Digital Games & Software Audible Audiobooks

fire HDX 8.9 STILL LIGHTER THAN AIR

950,000 more pixels | \$120 less than iPad Air 2 >Learn more

What's in the Bag? Denim from Levi's Halloween Shop

Amazon Fashion

MUST-HAVE DENIM & MORE

Levi's latest casual styles for men and women. >Shop Levi's

Related to Items You've Viewed

You viewed Customers who viewed this also viewed

ALL-NEW kindle Only \$79 >Shop now

APC by Schneider Electric Learn more

Average Amazon.com customer review ★★★★★ (17)

★★★★★ Great for keeping your network up! "This is a really good idea...to keep your network up and running."

Veterans Day Is November 11 >Shop now

Query Results

Donald's Amazon.com Today's Deals Gift Cards Sell Help

Shop by Department Search All web service architecture ferguson Go

9 DAYS LEFT Sponsored By Hershey's Halloween Shop >Shop now

Hello, Donald Your Account Your Prime Cart Wish List

7 results for "web service architecture ferguson"

Show results for Books > Internet & Web Culture Web Development & Design Web Services Web Development & Design Programming

Refine by Amazon Prime Prime Eligible Avg. Customer Review ★★★★☆ & Up ★★★☆☆ & Up ★★☆☆☆ & Up International Shipping AmazonGlobal Eligible Condition New Used

GET BUSINESS-SIZED SPENDING POWER

Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and... by Sanjiva Weerawarana, Francisco Curbera, Frank Leymann and Tony Storey (Apr 1, 2005)
\$64.99 \$36.64 Paperback Prime
Only 1 left in stock - order soon.
More Buying Choices - Paperback
\$9.97 new (23 offers)
\$0.49 used (40 offers)

Service-Oriented Computing - ICSOC 2005: Third International Conference, Amsterdam, The Netherlands, December... by Boualem Benatallah, Fabio Casati and Paolo Traverso (Jan 9, 2006)
\$140.00 \$120.30 Paperback Prime
Get it by Friday, Oct 24
More Buying Choices - Paperback
\$108.17 new (17 offers)
\$110.00 used (4 offers)

Advances in Computer and Information Sciences and Engineering by Tarek Sobh (Aug 27, 2008)
\$209.00 \$163.63 Hardcover Prime
Only 1 left in stock - order soon.
More Buying Choices - Hardcover
\$85.76 new (25 offers)
\$89.00 used (18 offers)

Java EE 5
See newer edition of this book
\$49.38 new (1 offer)
\$52.36 used (2 offers)

Excerpt
"... Storey, D.F. Ferguson, *Web Services Platform Architecture*, Prentice ..."
[Read More](#)

Books: See all 7 items

Excerpt
"... Ferguson, *Web Services Platform Architecture*, Prentice Hall, 2005 z ..."
[Read More](#)

Books: See all 7 items

{author, "Ferguson"}
{includes, {paper, {author, "Ferguson"}}}
{references, {book, {author, "Ferguson"}}}

Data and Rendering

- Two different *renderings*
- Of the same underlying document(s)

This screenshot shows a rendered version of a book page from Amazon. The page displays the book cover for "Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More" by Sanjiva Weerawarana. It includes a summary, reviews, and purchase options. A purple box highlights the top portion of the page, showing the book cover and title.

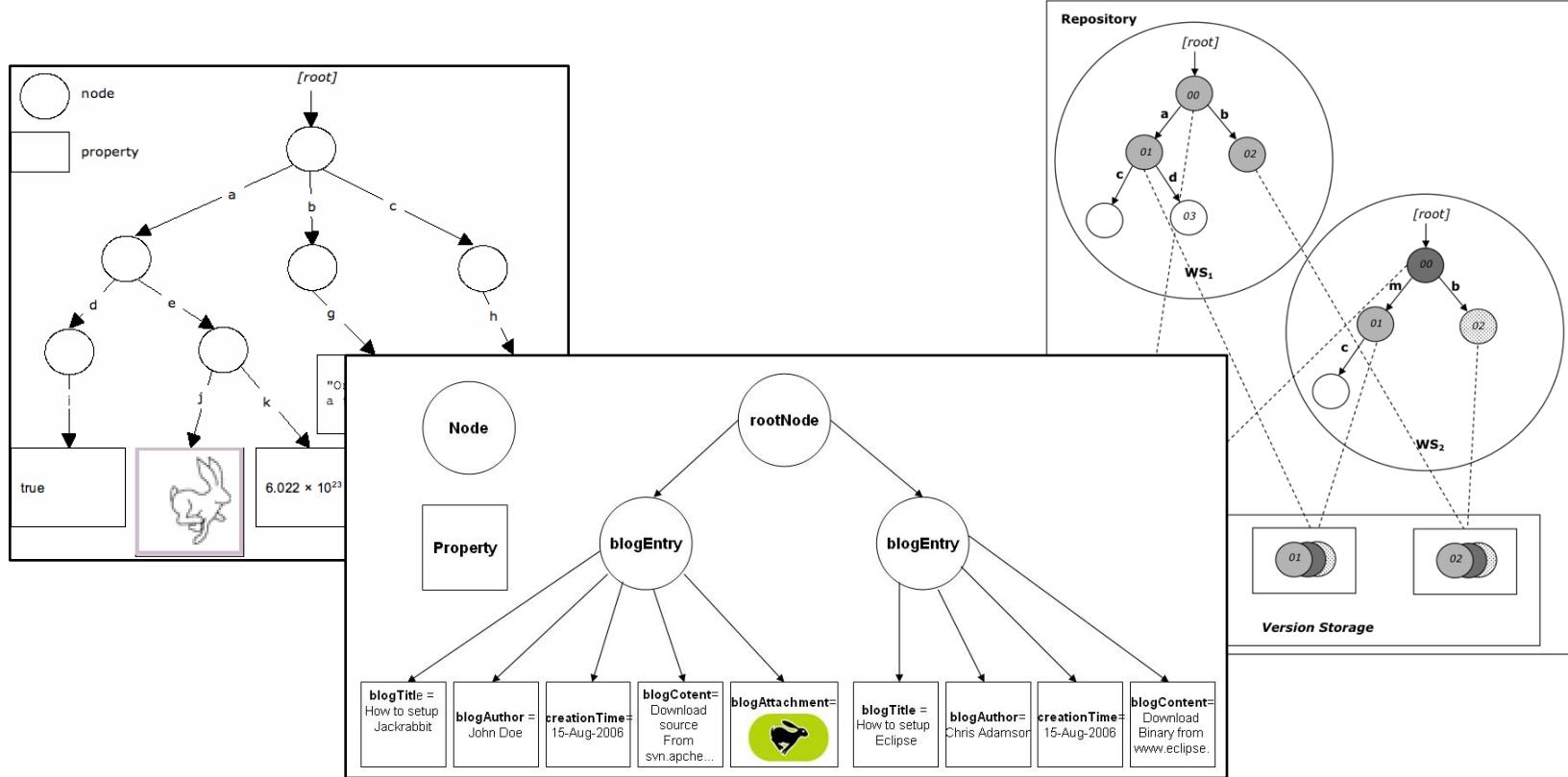
This screenshot shows the raw XML document being rendered. The XML code is identical to the one shown in the previous screenshot, but it is presented as a plain text document with syntax highlighting for XML tags. The rendered content (book cover, title, and reviews) is visible at the top of the page.

Content Management System

Some Terminology

- A content management system (CMS) is a computer application that
 - Allows publishing, editing and modifying content, organizing, deleting as well as maintenance from a central interface.
 - Such systems of content management provide procedures to manage workflow in a collaborative environment.
- A *web* content management system is a bundled or stand-alone application
 - To create, deploy, manage and store content on Web pages.
 - Content includes text and embedded graphics, photos, video, audio, ...
 - Content includes *code* that displays content in a specific way
 - A Web CMS may catalog and index content, select or assemble content at runtime, or deliver content to specific visitors in a requested way, such as other languages.

CMS Data Model



Sparq Home Page

Home ...

HOME NOTIFICATIONS SEARCH MY ACCOUNT



LOVERS & OTHERS
More at
loversandothers.tv

Lovers & Others +



Hat Trick Bitches Webseries +



Female Friendly +



Nobodies Part 1 +



Nobodies Part 2 +



Polo +

Theater People

A comedy web series about the world of independent theater and the people who live it. Set against the Minneapolis/St. Paul theater scene, "Theater People" brings together the talents of dozens of actors, designers and technicians to cast a humorous eye on the utterly unique experience that is a life in the theater.

[SEASONS](#)[EPISODES](#)

Theather people Season 1

Theater People

Theather people Season 1

[Add a new serie](#)

Theather people Season 1



Day Jobs



Theater People Season 1 - Episode 01



Oh My God, I Should So Be Dead Right Now



Theater People Season 1 - Episode 02



Blindfolds and Cigarettes



Theater People Season 1 - Episode 03



Episode : Theater People Season 1 - Episode 01



Name

Theater People Season 1 - Episode 01

Title

Day Jobs

Short description

A former Theater Person discovers that you can take the actor out of the theater, but you can't take the theater o

Long description

A former Theater Person discovers that you can take the actor out of the theater, but you can't take the theater o

Image

http://

Video

http://

5001.i

Sequence

1

[SAVE](#)[RESET](#)

VIDEO INFORMATION

Name
Video Name

ID
5162534386001

Short Description
Description of a video

Tags
property=test, series=season 1

Ref ID
...

Related Link URL
...

Related Link Text
...

Long Description
Long Description

Advertising
Ad Supported

Ad Keys
...

Folder
...

Manual Playlists
...

Sharing
...

VIDEO FILES

Source File Name
big_buck_bunny_720p_stereo.avi

Replace Source File **Retranscode**

Dimensions	Format	Upload Date
480 x 270 @ 513 kbps	MP4-H264	Sun, Oct 9, 2016 1:13 PM
480 x 270 @ 514 kbps	HLS	Sun, Oct 9, 2016 1:13 PM
640 x 360 @ 794 kbps	HLS	Sun, Oct 9, 2016 1:13 PM
640 x 360 @ 995 kbps	HLS	Sun, Oct 9, 2016 1:13 PM
640 x 360 @ 996 kbps	MP4-H264	Sun, Oct 9, 2016 1:13 PM
960 x 540 @ 1290 kbps	HLS	Sun, Oct 9, 2016 1:13 PM
960 x 540 @ 1820 kbps	MP4-H264	Sun, Oct 9, 2016 1:13 PM
960 x 540 @ 1820 kbps	HLS	Sun, Oct 9, 2016 1:13 PM
1280 x 720 @ 2119 kbps	HLS	Sun, Oct 9, 2016 1:13 PM

IMAGES

For best results, use JPG or PNG format with a minimum width of 640px for video stills and 160px for thumbnails. Aspect ratios should match the video, generally 16:9 or 4:3. [Read More](#)

Video Still 

Thumbnail 

CUSTOM FIELDS

There are no custom fields set for this video.

AVAILABILITY

Scheduling
Always Available

TEXT TRACKS

There are no text tracks associated with this video.

Content Managed

- Metadata
- Renditions
- Poster image
- Thumbnail image
- Text/subtitle tracks
- Availability (time, geos, ...)

Putting the Pieces Together

4th Project (Part...) Building on 3rd Project

- Define and provide a Lambda/REST interface to a “content catalog” in DynamoDB.
 - Model: Content Instance
 - Property (e.g. HBO, CBS) is a set of one or more Franchises.
 - A Franchise (e.g. Game of Thrones, Bing Bang Theory) is a set of one or more Series.
 - A Series (e.g. Season 1, Christmas Specials) is a set of one or more Episodes.
 - An Episode is an instance of a “TV show.”
 - Data for each type is just (ID, name).

- Content Catalog
 - Create a Lambda function to handle requests.
 - Parse JSON manifest.
 - Extract metadata.
 - Create content instance in DynamoDB.
 - Create thumbnail and video links.
 - Create manifest file.
 - Send manifest to SQS queue.
- Generate thumbnails
 - Create Lambda function to process thumbnails.
 - Extract thumbnail URL from manifest.
 - Load thumbnail from S3.
 - Create thumbnail image.
- Extract video
 - Create Lambda function to process video.
 - Extract video URL from manifest.
 - Load video from S3.
 - Create video file.
- Log processing
 - Create CloudWatch logs.
 - Log processing steps.
- Use AWS CloudWatch Metrics to monitor performance.