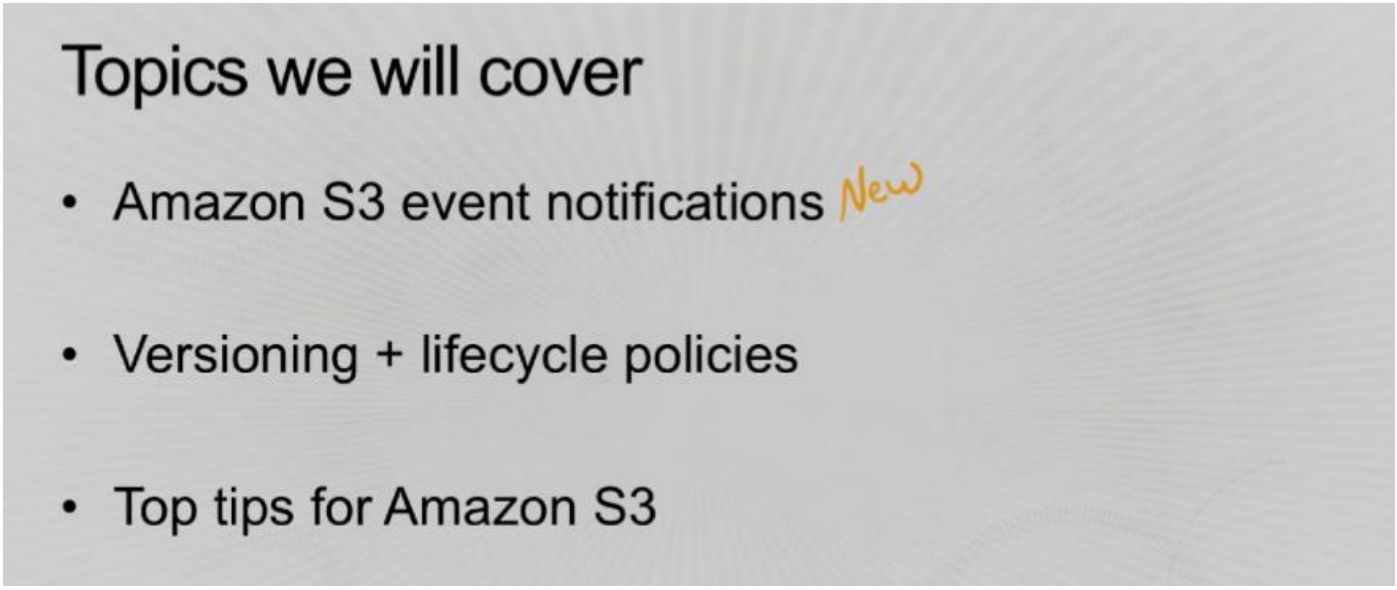
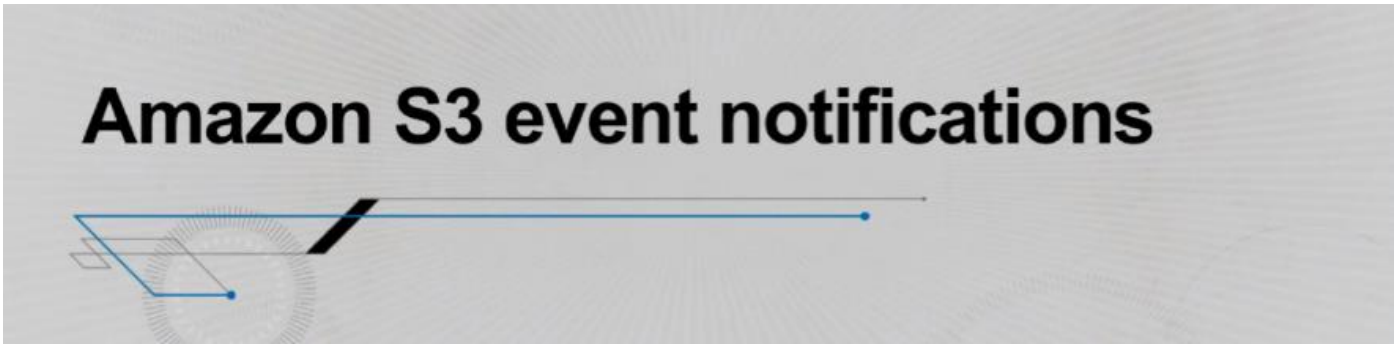
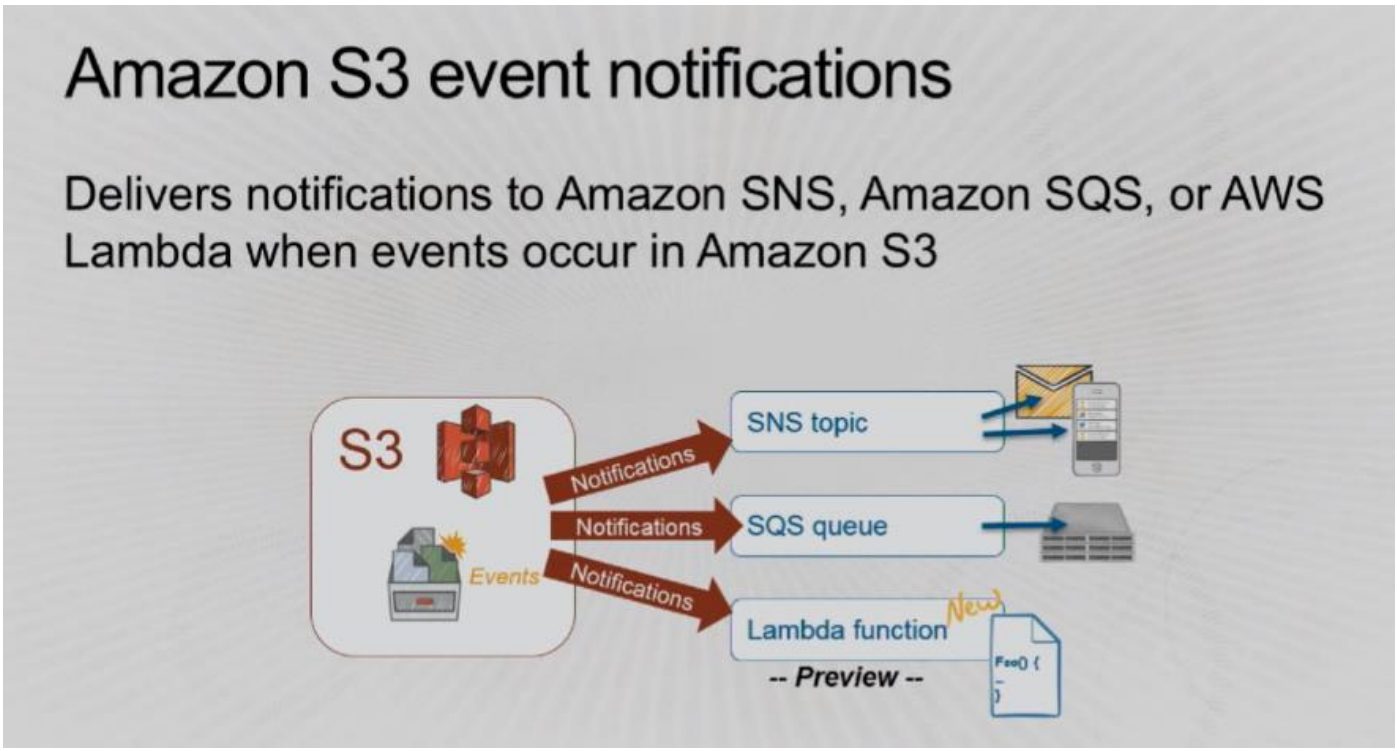
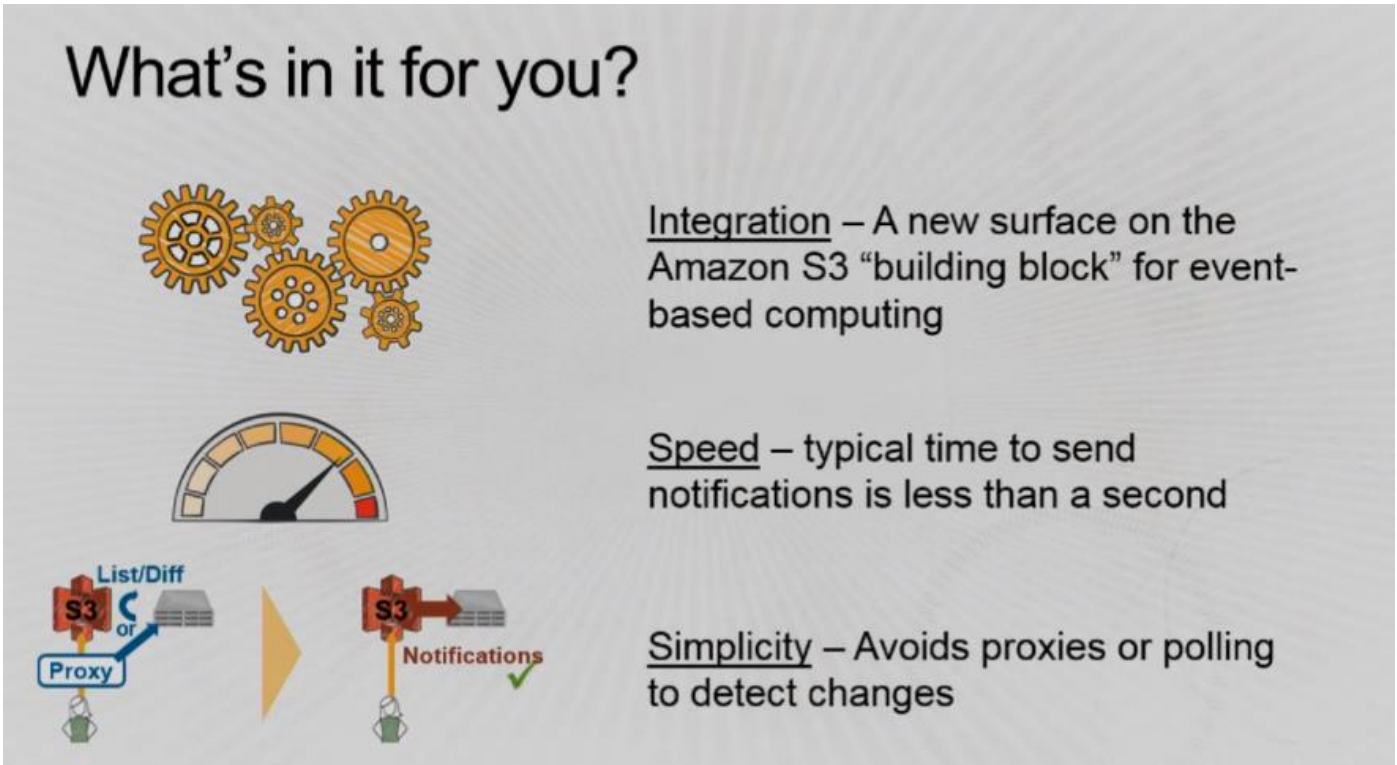
Come learn about new and existing Amazon S3 features that can help you better protect your data, save on cost, and improve usability, security, and performance. We will cover a wide variety of Amazon S3 features and go into depth on several newer features with configuration and code snippets, so you can apply the learnings on your object storage workloads.





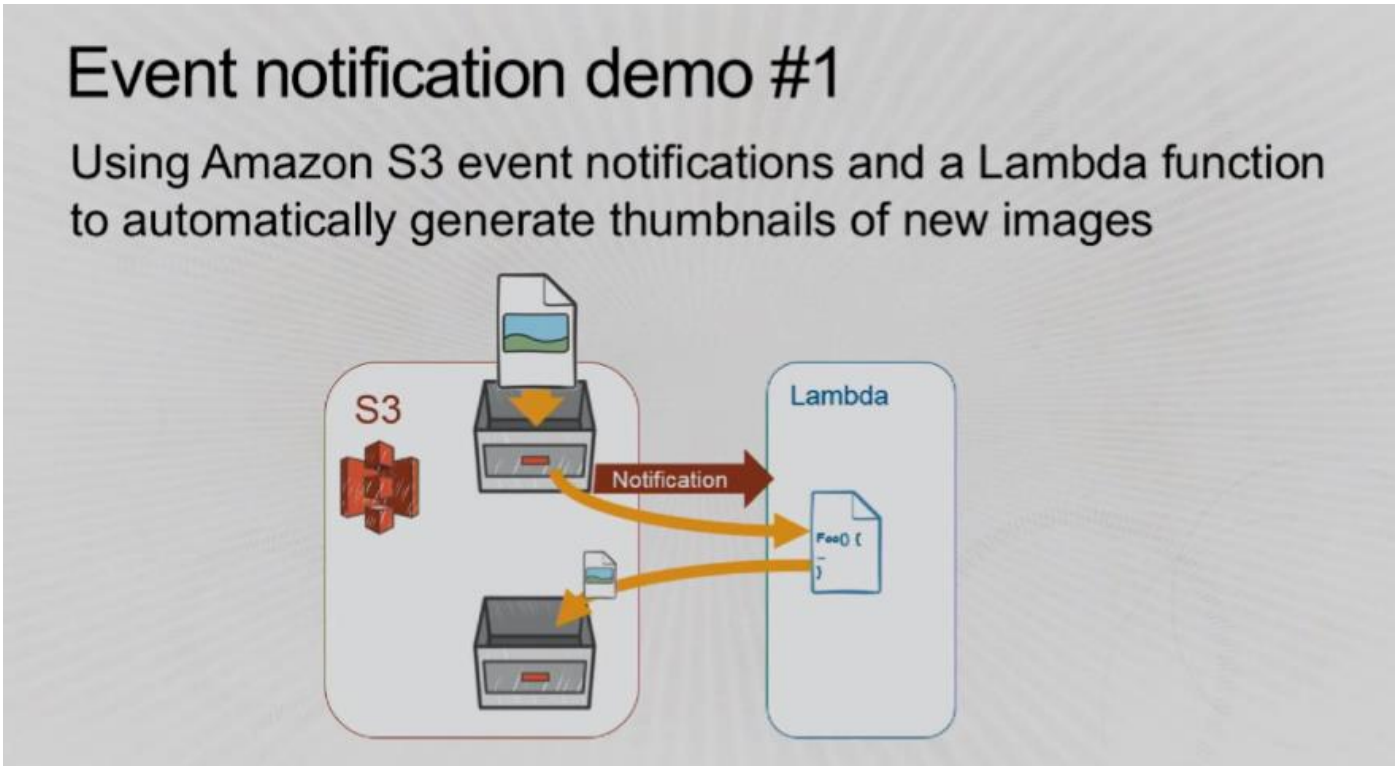


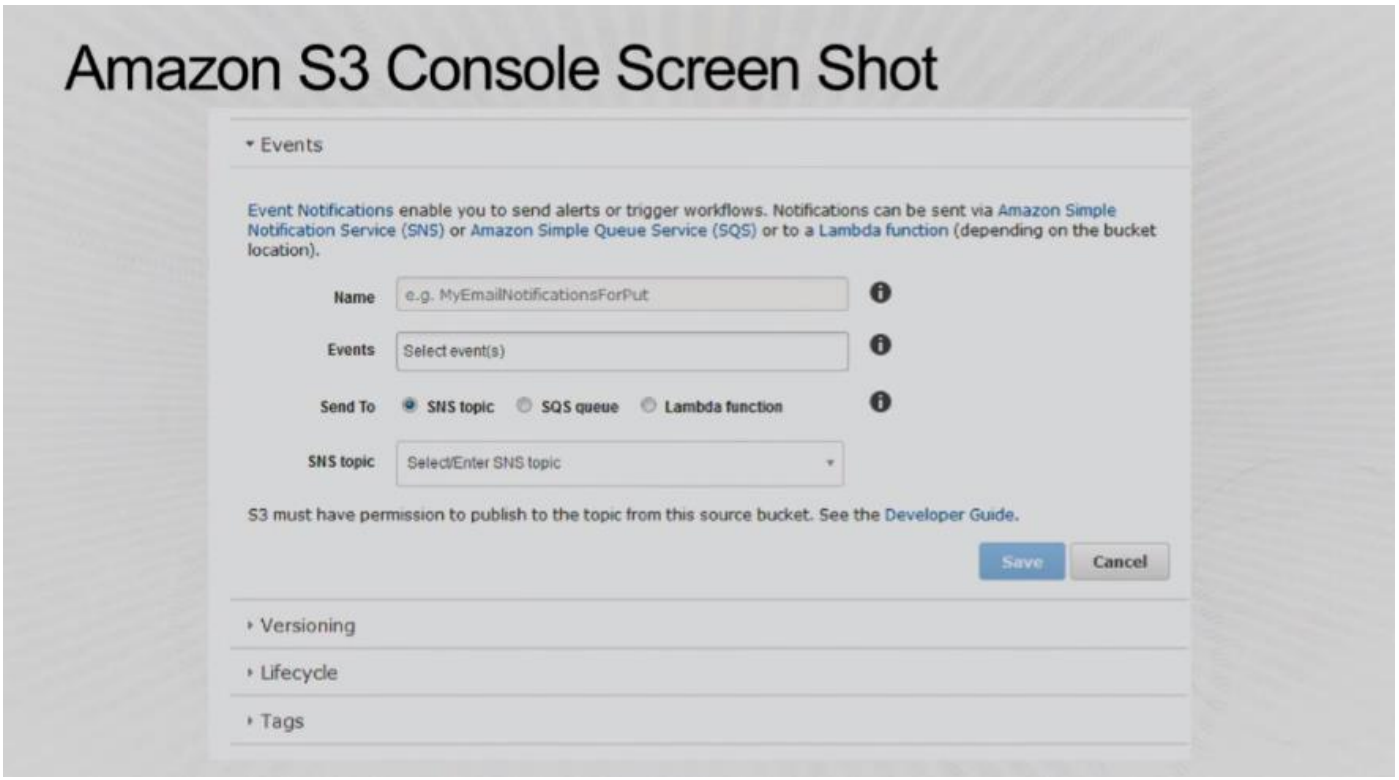
Events are things that happen on your S3 buckets, some form of actions like a new object being created with a PUT. You can send the event notification to SNS for sending alerts to email addresses or HTTP endpoints, or to SQS to trigger a workflow, you can set up your EC2 worker fleet to pull those messages off the SQS queue and work on them. Or you can send the event to Lambda to run some code, this allows you take an event in S3 and do some processing in Lambda on-demand.



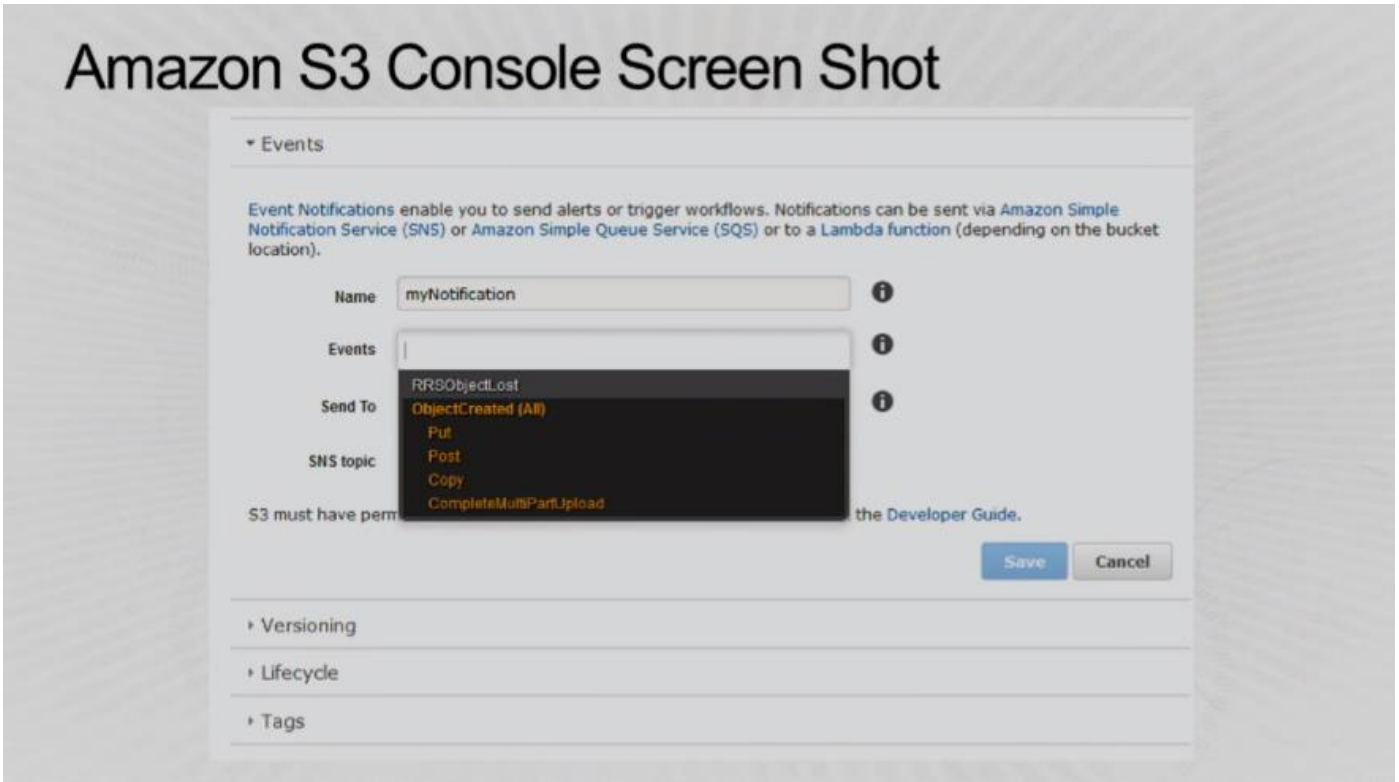
This is event-based computing

You can use notifications in any of the above ways like resizing and image file that arrives in your S3 bucket. You can process data files or log files as they arrive in S3 and forward the results to your DynamoDB table. You can also have your CloudWatch logs delivered to S3 and do something with the data as they come in.

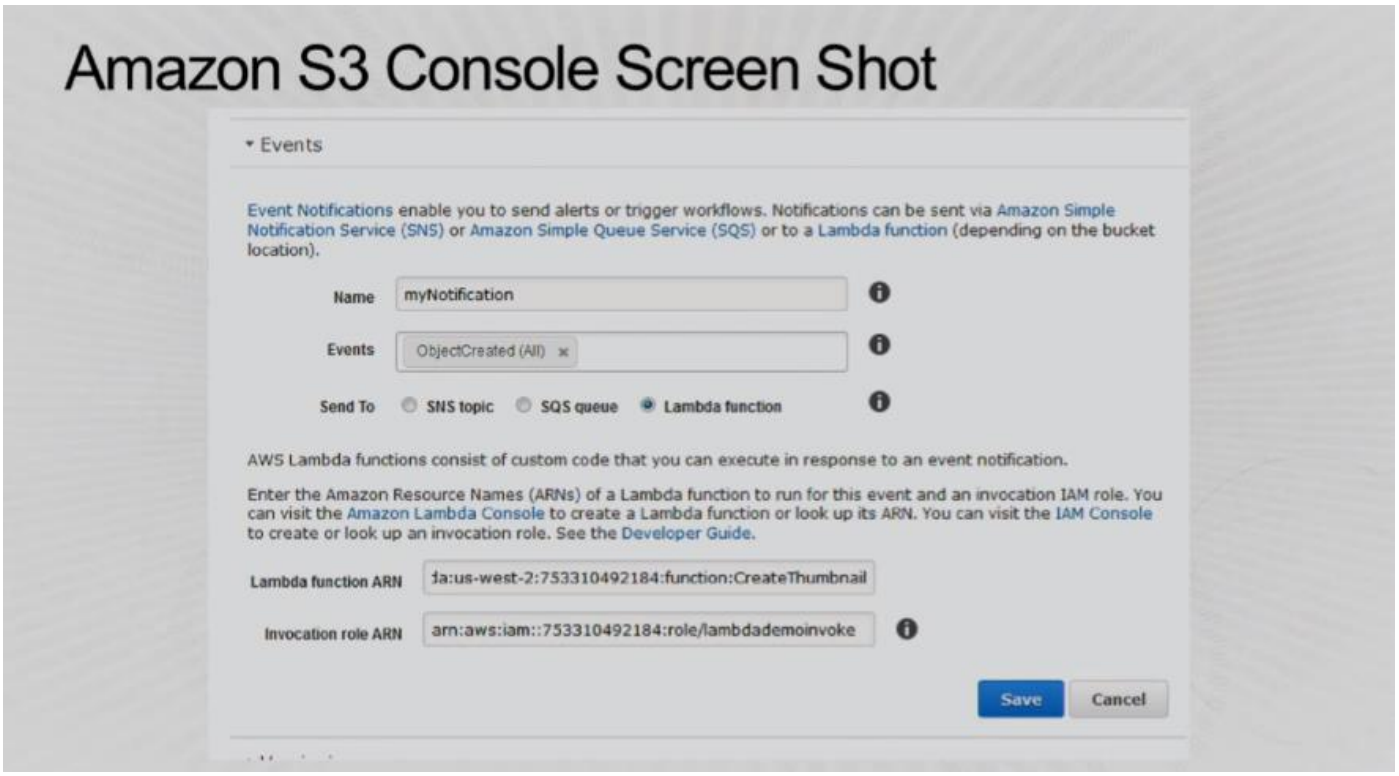


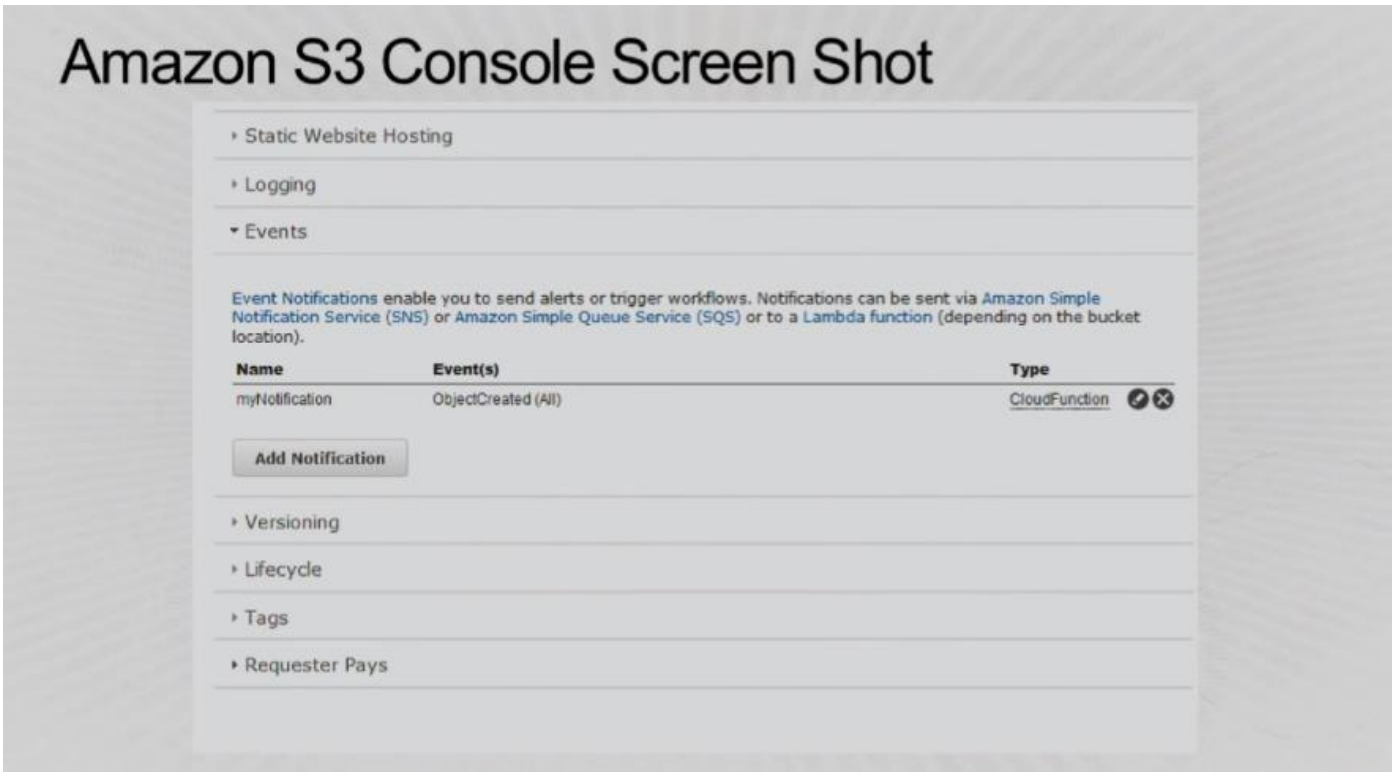


We set this up for the S3 bucket



You can choose the event that you want to create the notification for like PUTS, All, etc. then you choose where to send the event to like a Lambda and supply both the ***arn*** of the Lambda function and the invocation role, which is a way to control permission to the lambda function by giving S3 the authority to invoke that lambda.





We have now created the notification for this S3 bucket. We can create many notifications for a single bucket.

Lambda function code

```
function handler(event, context) {
// read values from the event
   var srcBucket = event.Records[0].s3.bucket.name;
   var srcKey    = event.Records[0].s3.object.key;
   var dstBucket = srcBucket + "resized";
   var dstKey    = "resized-" + srcKey;

   // sanity check: validate that source and destination are different buckets
   if (srcBucket == dstBucket) {
     return;
   }

   // make sure it's a jpg or png
   var imageType = srcKey.match(/\.([^.]*)$/)[1];
   if (imageType != "jpg" && imageType != "png") {
      return;
   }
```

Continued…

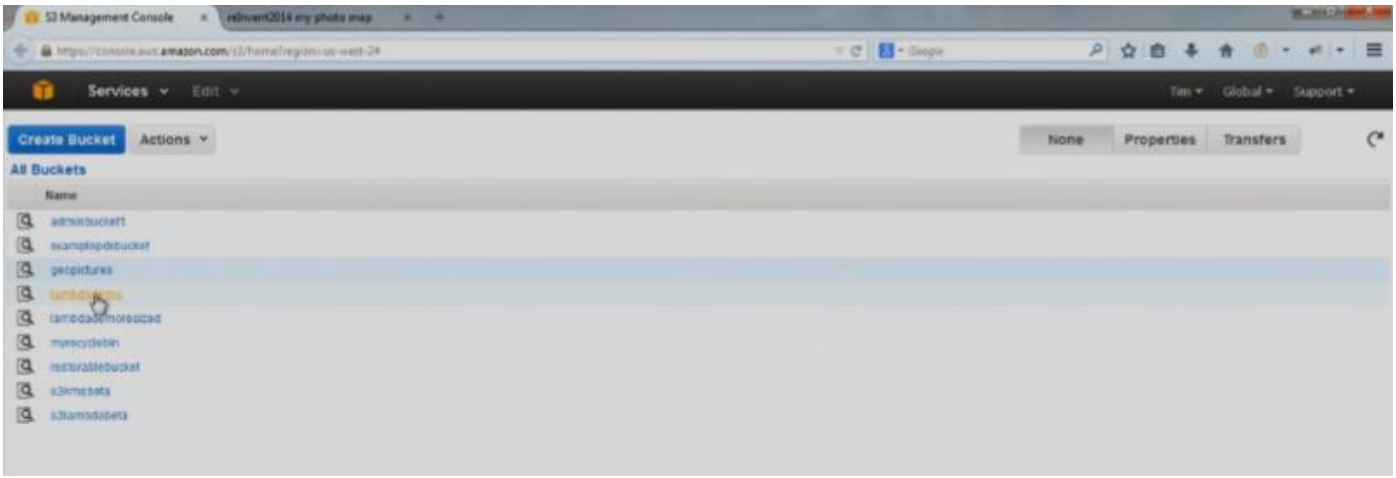
This is a NodeJS code for our lambda function

Lambda function code – continued

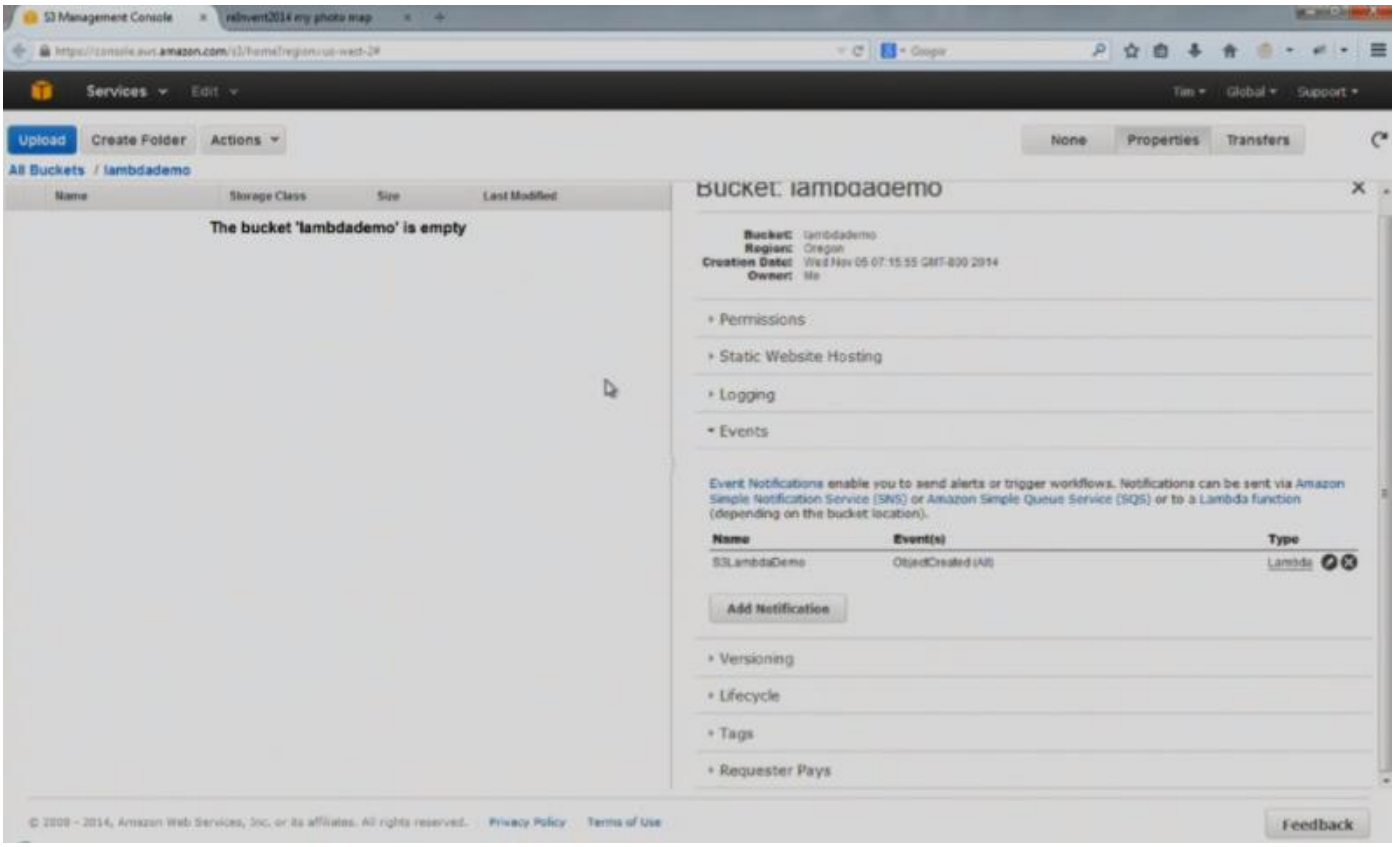
```
   // download image from S3 into buffer
   function download(next) {
      s3.getObject({Bucket: srcBucket, Key: srcKey}, next);
    },
   // generate the thumbnail
   function tranform(response, next) {
      gm(response.Body).size(function(err, size) {
         this.resize(width, height).toBuffer(imageType, function(err, buffer) {
            if (err) { next(err);}
            else {next(null, response.ContentType, buffer);}
         });
      });
   },
   // Put into S3
   function upload(data, next) {
     s3.putObject({Bucket: dstBucket, Key: dstKey, Body: data, ContentType: contentType
        }, next);
```

We are using the ImageMagick library that is available in Lambda to create the thumbnails

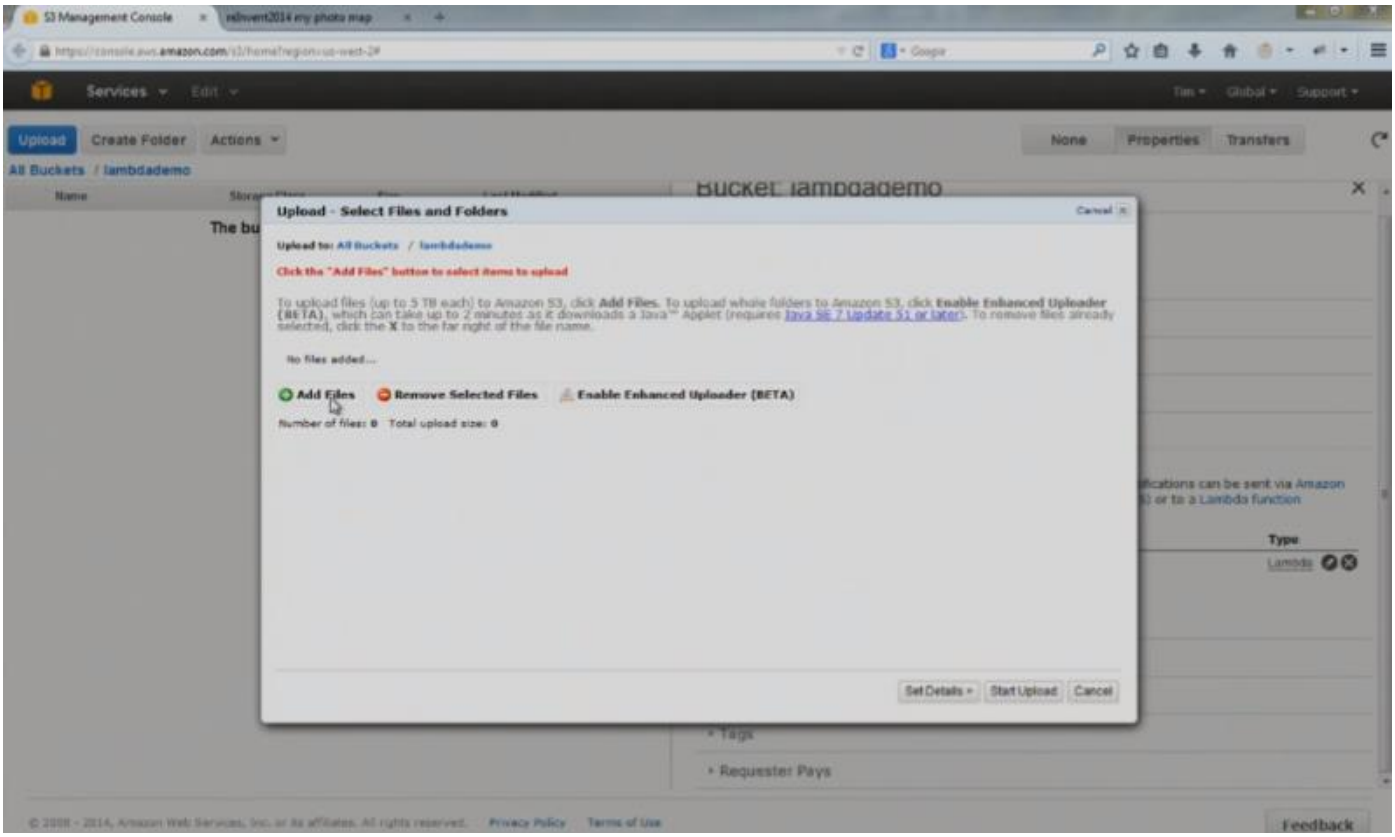
Demo

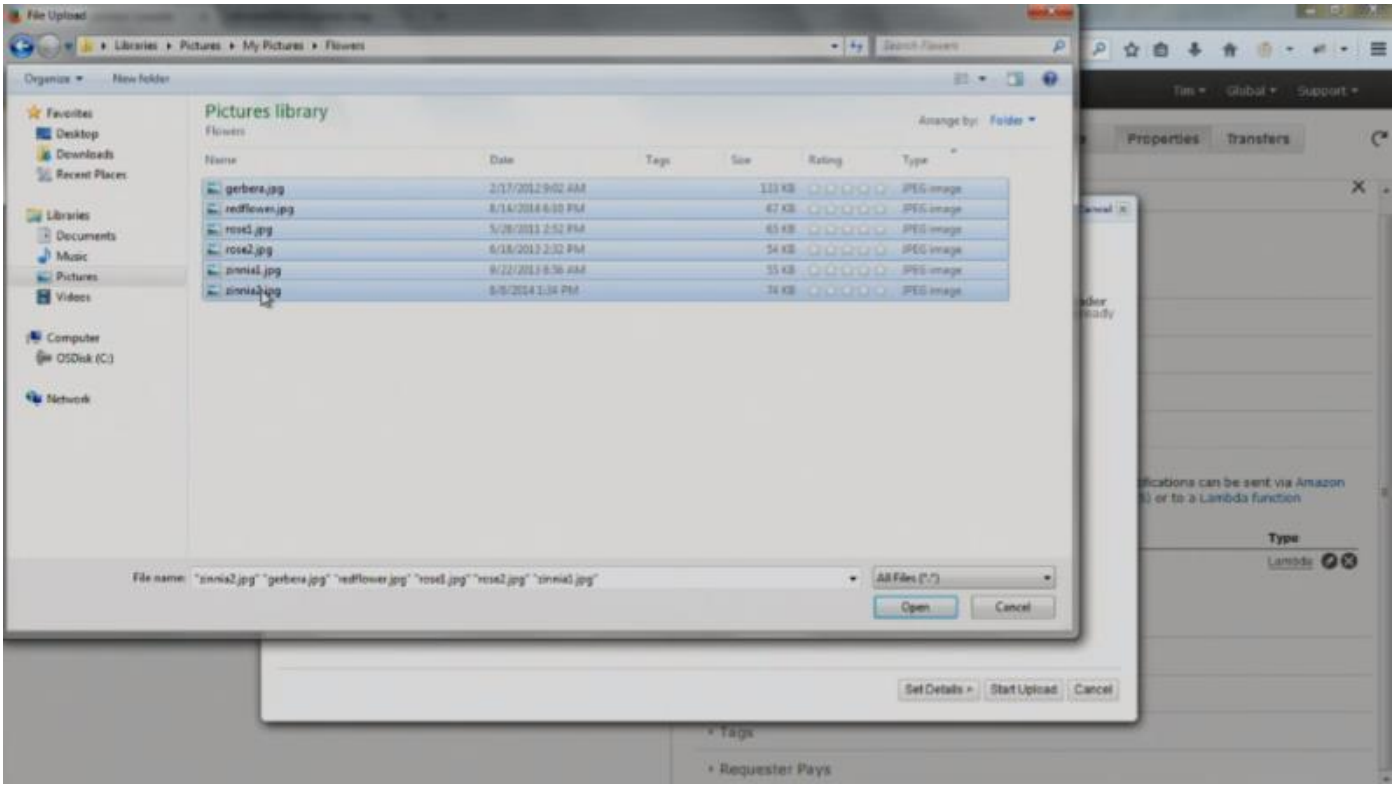


This is the source S3 bucket

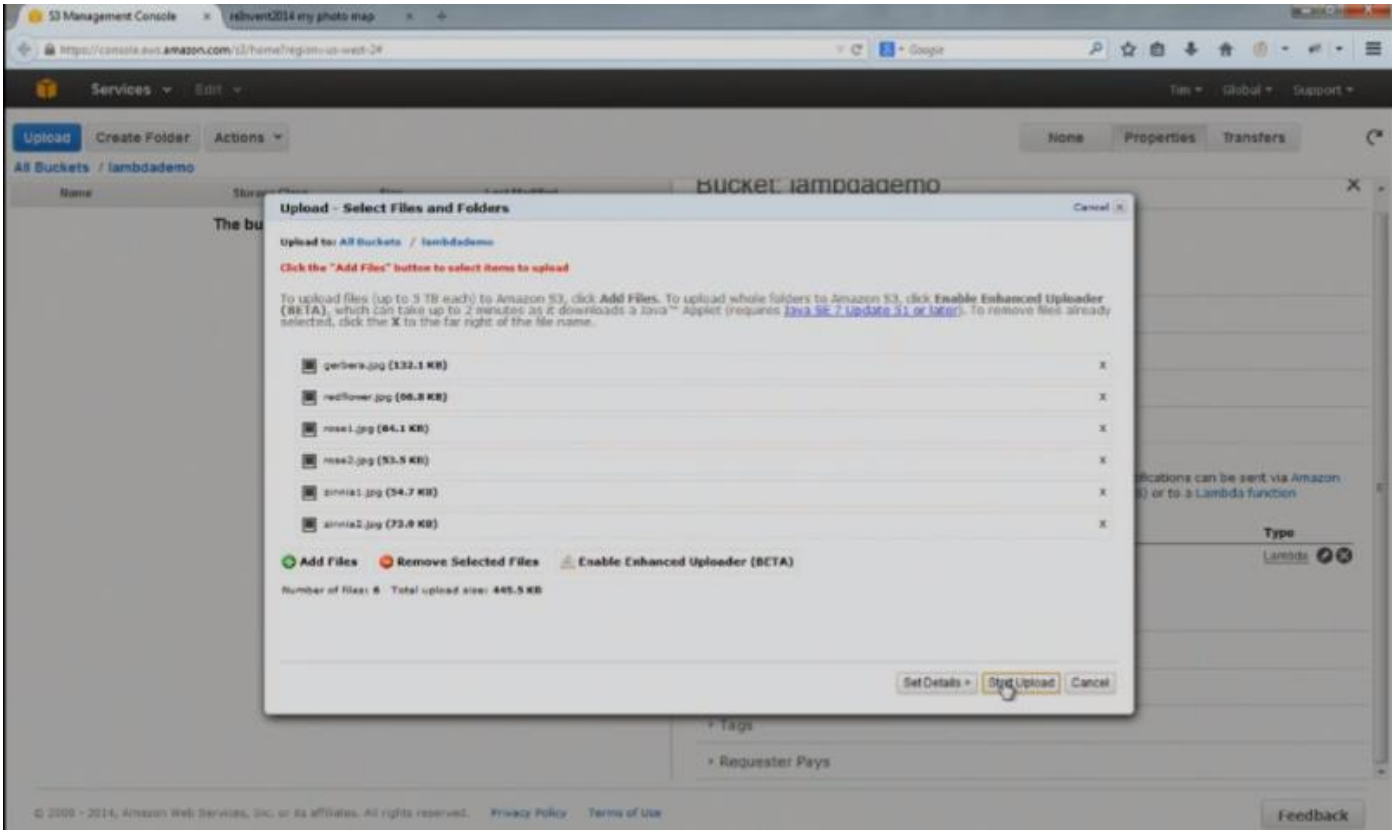


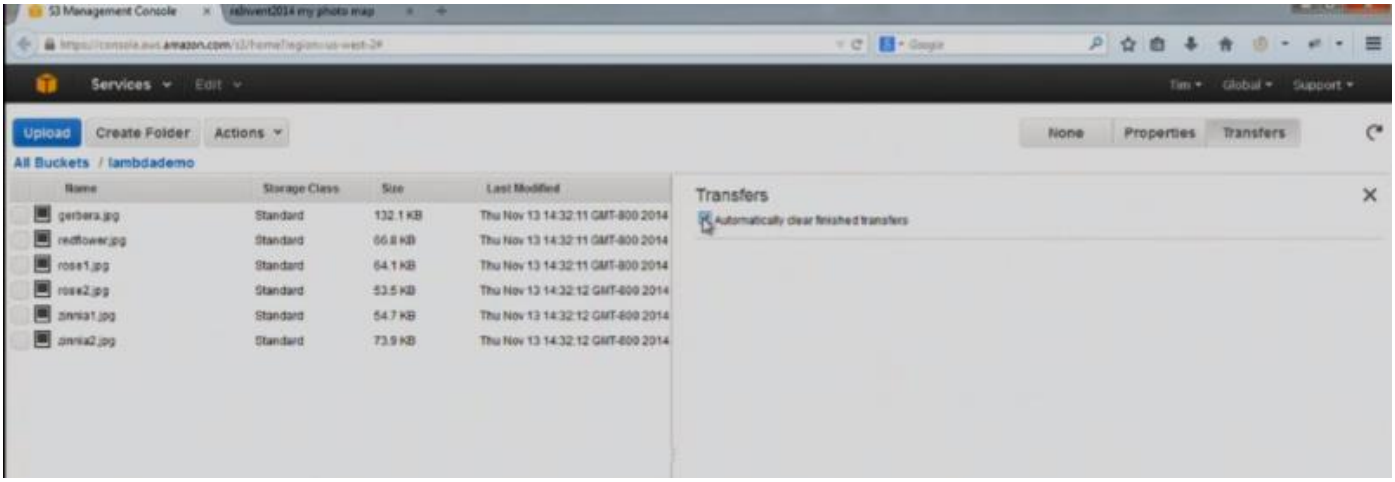
We have set up an ObjectCreated(All) notification for this S3 bucket to be sent to the S3LambdaDemo lambda function

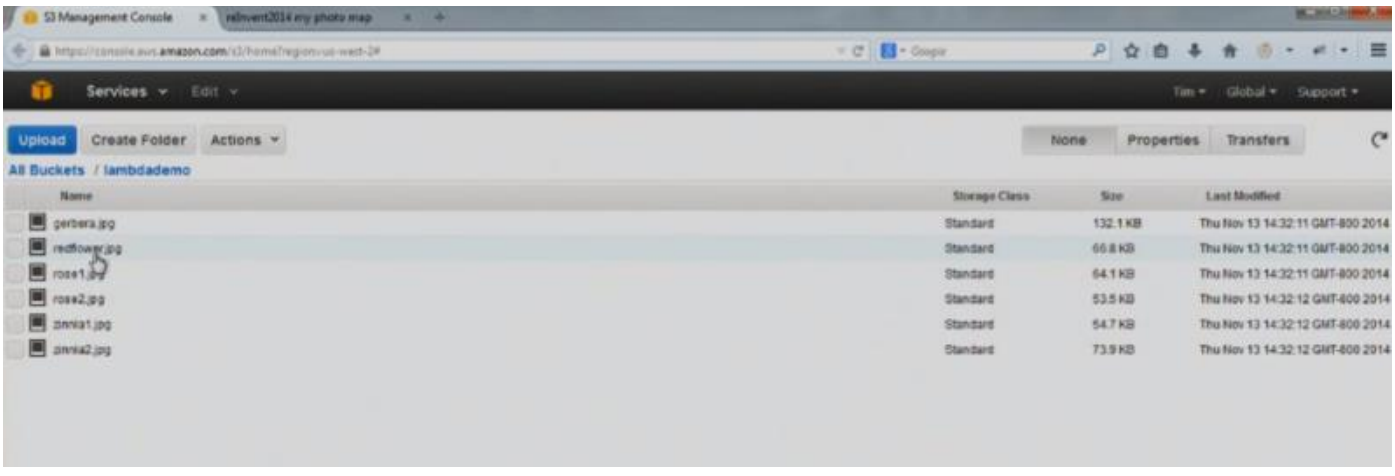


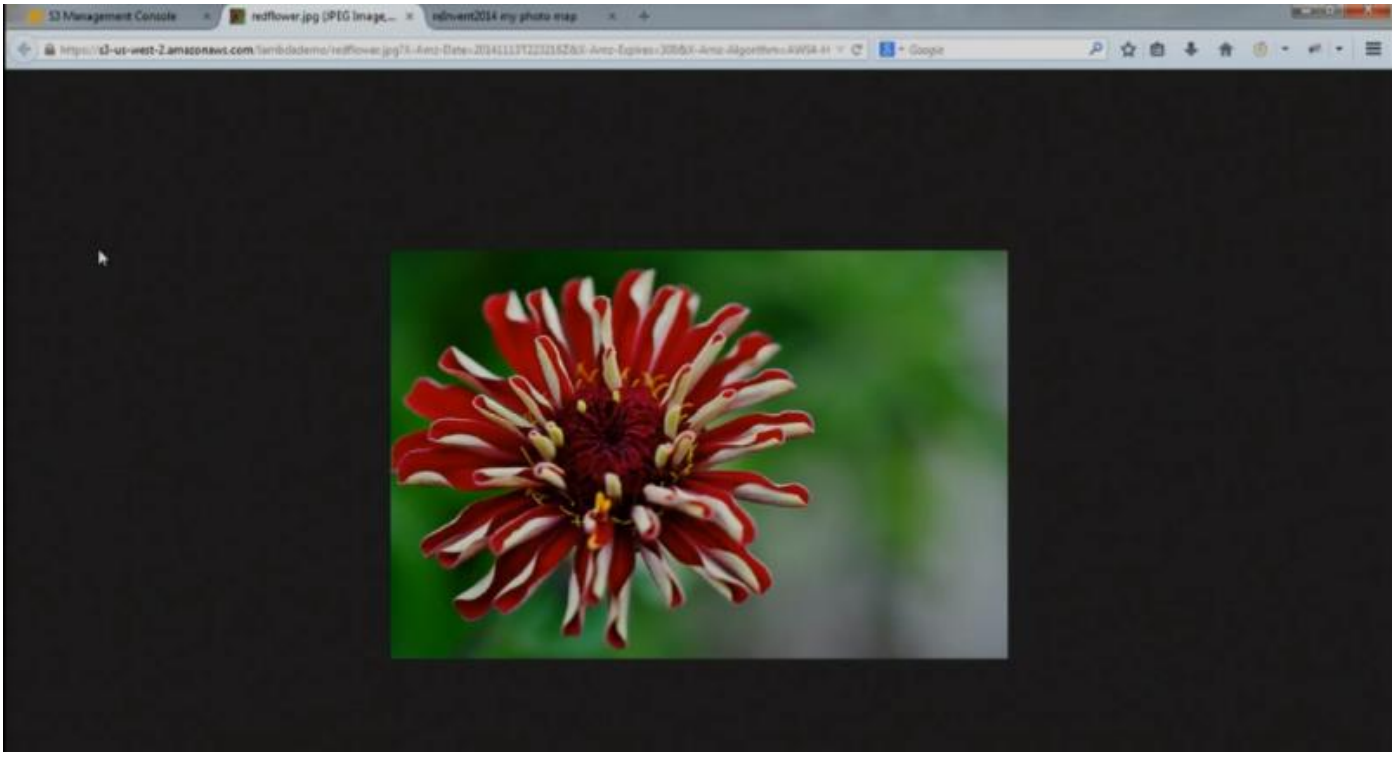


We can now upload some files into the source S3 bucket as above

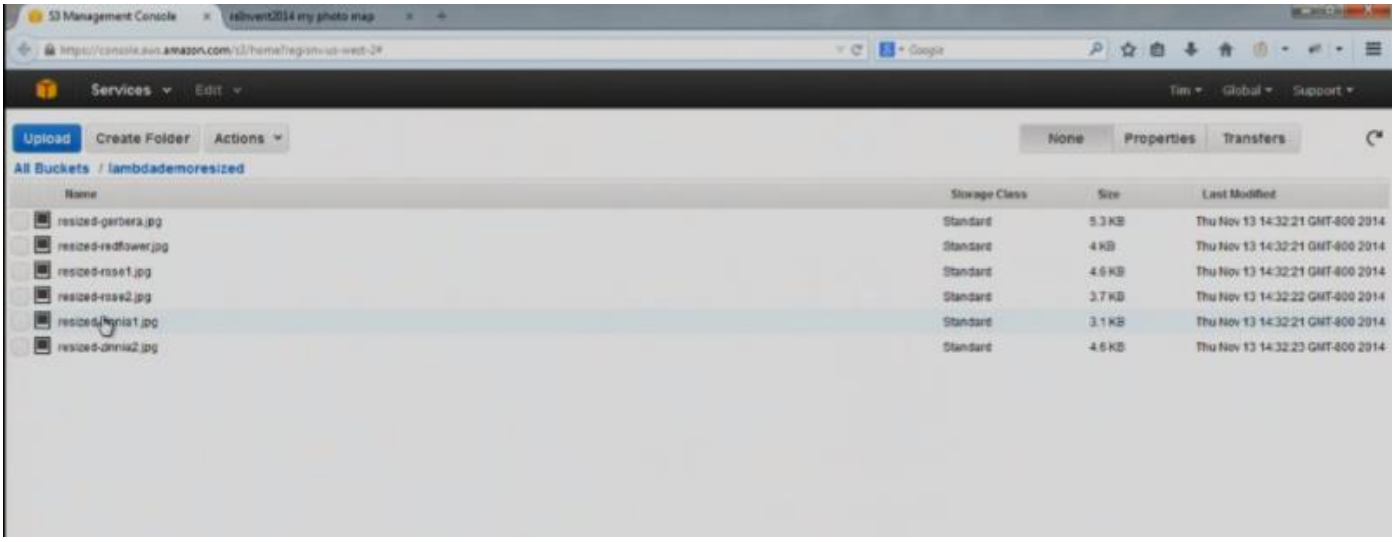

S3 Management Console
reInvent2014 my photo map
Services
Edit
Upload
Create Folder
Actions
None
Properties
Transfers
All Buckets / lambdademo
Upload - Select Files and Folders
Upload to: All Buckets / lambdademo
Click the "Add Files" button to select items to upload
To upload files (up to 5 TB each) to Amazon S3, click Add Files. To upload whole folders to Amazon S3, click Enable Enhanced Uploader (BETA), which can take up to 2 minutes as it downloads a Java™ Applet (requires Java SE 7 Update 51 or later). To remove files already selected, click the X to the far right of the file name.
gerbera.jpg (132.1 KB)
redflower.jpg (66.8 KB)
rose1.jpg (64.1 KB)
rose2.jpg (53.5 KB)
zinnia1.jpg (54.7 KB)
zinnia2.jpg (73.0 KB)
Add Files
Remove Selected Files
Enable Enhanced Uploader (BETA)
Number of files: 6 Total upload size: 445.5 KB
Set Details
Start Upload
Cancel
Tags
Requester Pays
Feedback


S3 Management Console
reInvent2014 my photo map
Services
Edit
Upload
Create Folder
Actions
None
Properties
Transfers
All Buckets / lambdademo
Name
Storage Class
Size
Last Modified
gerbera.jpg Standard 132.1 KB Thu Nov 13 14:32:11 GMT-800 2014
redflower.jpg Standard 66.8 KB Thu Nov 13 14:32:11 GMT-800 2014
rose1.jpg Standard 64.1 KB Thu Nov 13 14:32:11 GMT-800 2014
rose2.jpg Standard 53.5 KB Thu Nov 13 14:32:12 GMT-800 2014
zinnia1.jpg Standard 54.7 KB Thu Nov 13 14:32:12 GMT-800 2014
zinnia2.jpg Standard 73.9 KB Thu Nov 13 14:32:12 GMT-800 2014
Transfers
Automatically clear finished transfers

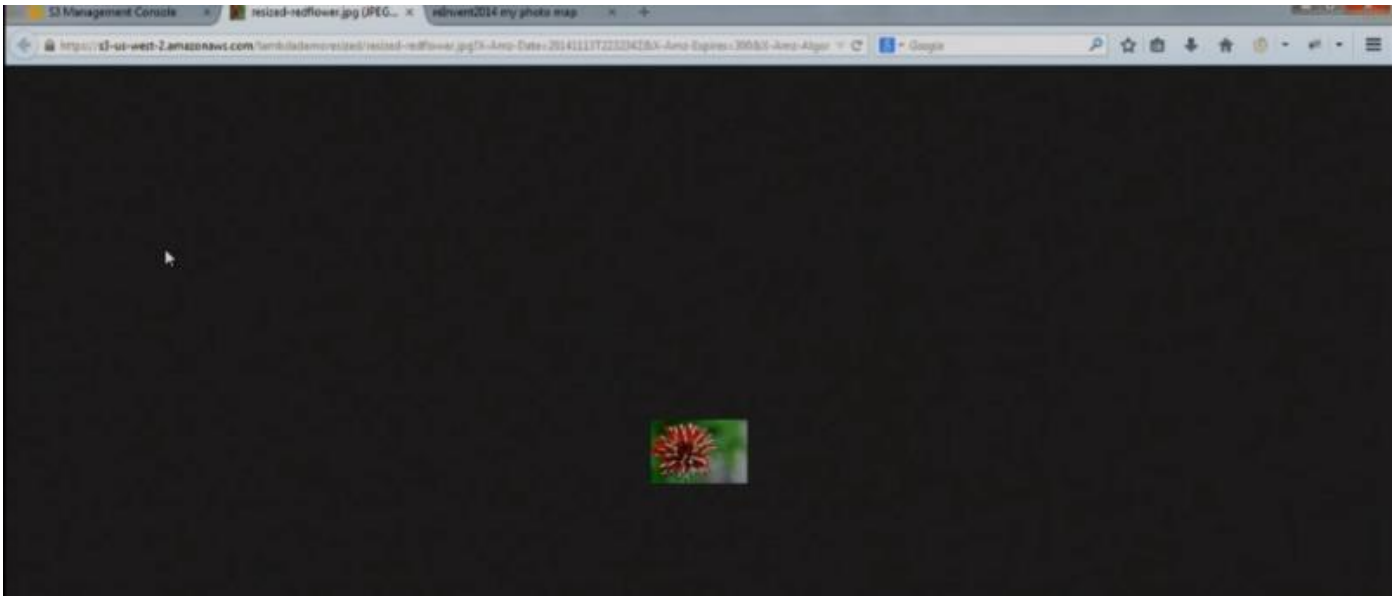

S3 Management Console
reInvent2014 my photo map
Services
Edit
Upload
Create Folder
Actions
None
Properties
Transfers
All Buckets / lambdademo
Name
Storage Class
Size
Last Modified
gerbera.jpg Standard 132.1 KB Thu Nov 13 14:32:11 GMT-800 2014
redflower.jpg Standard 66.8 KB Thu Nov 13 14:32:11 GMT-800 2014
rose1.jpg Standard 64.1 KB Thu Nov 13 14:32:11 GMT-800 2014
rose2.jpg Standard 53.5 KB Thu Nov 13 14:32:12 GMT-800 2014
zinnia1.jpg Standard 54.7 KB Thu Nov 13 14:32:12 GMT-800 2014
zinnia2.jpg Standard 73.9 KB Thu Nov 13 14:32:12 GMT-800 2014



This is an example of one of the uploaded pictures that the Lambda will create a thumbnail for and save into the destination S3 bucket.



We can now see the created thumbnails images in our resized bucket, these were each created by the Lambda function using the S3 PutObject event notification as a trigger.



This is the equivalent thumbnail

Event notification details

- No additional charge from Amazon S3; you pay only for associated use of Amazon SNS, Amazon SQS, Lambda
- Configured at the bucket level and supports different destinations for different events
- Today supports PUT, POST, COPY, MultiPartComplete, and RRSObjectLost events; more to come

Event notification details, continued

- Highly reliable – designed for nine ‘9’s with at least once delivery
- The destination topic, queue, or Lambda function must be in the same AWS region as the S3 bucket

Sample notification JSON


```
{
   "Records":[
      {
         "eventVersion":"2.0",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"2014-11-06T18:02:44.551Z",
         "eventName":"ObjectCreated:Put",
         "userIdentity":{
            "principalId":"A1AM2HK7YZABCD"
         },
         "requestParameters":{
            "sourceIPAddress":"10.240.9.17"
         },
         "responseElements":{
            "x-amz-request-id":"0DAFE1024D7E1234",
            "x-amz-id-
2":"YVbY/Q7NSt1Rao68w1zlVfJr4H8x3aaEzVgPbzN8J0WE19mkPa7pqwertyuiop+m"
         },
```


Continued…

The content of the SNS notification that was sent to trigger the lambda function are in JSON format as above,

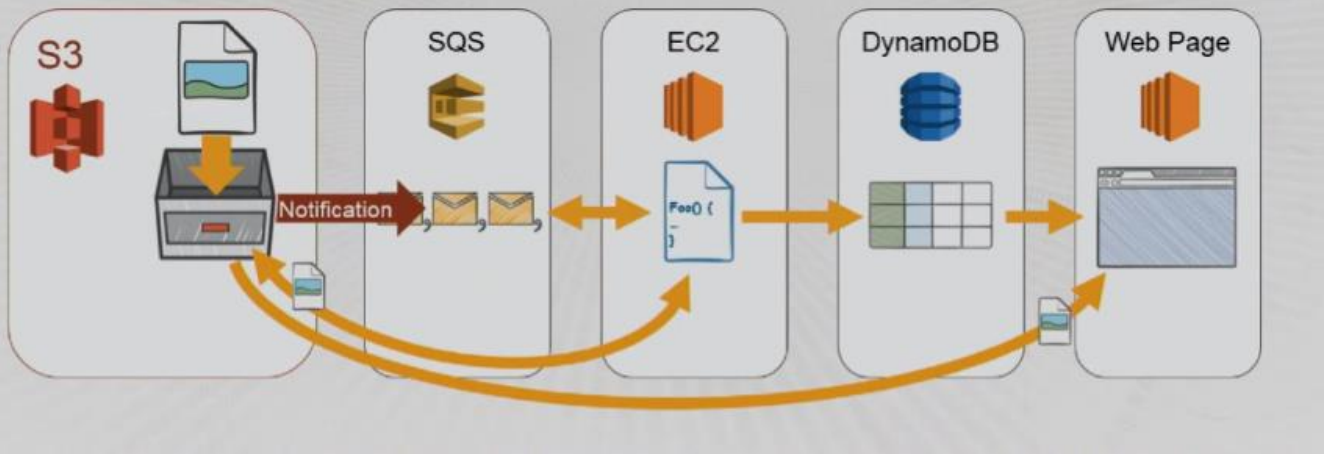
Sample notification JSON - continued


```
        "s3":{
           "s3SchemaVersion":"1.0",
           "configurationId":"lambdaeventdemo",
           "bucket":{
              "name":"lambdademo",
              "ownerIdentity":{
                 "principalId":"A1AM2HK7YZABCD"
              },
              "arn":"arn:aws:s3:::lambdademo"
           },
           "object":{
              "key":"Geranium.jpg",
              "size":102010,
              "eTag":"e03cf0062f8e99a32dd4937a1b2c31b2"
           }
        }
     }
  ]
}
```


Some of the information pertains to the S3 bucket and the object. There is an issue with S3's eventual consistency model if you already have an image in the S3 folder by the same name that you want to replace. The notification can be faster than the eventual object replacement and your lambda might end up creating the thumbnail for the old image that was already in the S3 bucket. A way to mitigate this will be to compare the eTag that is in the Notification message with the name of the object in the S3 bucket or you can use versioning.

Event notification demo #2

Using Amazon S3 event notifications with Amazon SQS and Java code to create thumbnails and record data in Amazon DynamoDB



We are going to use SQS as the message bus that gets the notifications as messages being stored on the queue. The we have a Java application running in EC2 that will be pulling those messages off the SQS queue, then read the message to be able to go pull the object out of S3, it will then look at the data in the S3 object and put some of the data into DynamoDB and also creating the image thumbnail and putting it back in another S3 bucket. We are also going to have a web page that displays the information about things that are getting stored in DynamoDB.

Main loop Java code

```
while (true) {
     // Long poll for the messages
     List<Message> messages = sqs.receiveMessage(receiveMessageRequest).getMessages();

     for (Message message : messages) {

          // Process the message
          processMessage(message.getBody());

          // Delete the message after we are done.
          sqs.deleteMessage(myQueueUrl, message.getReceiptHandle());
     }
}
```

Authentication and access control not shown for brevity, but published best practices should be followed.

This is the main loop of the Java code, it starts by polling the SQS queue for messages

processMessage Java code

```
private static void processMessage(String messageBody)    {

     // Parse the bucket and key from the event notification
     S3EventNotificationRecord item = S3EventNotificationItem.parseJson(messageBody)
          .getRecords().get(0);
     String bucketName = item.getS3().getBucket().getName();
     String objectName = item.getS3().getObject().getObjectNameOnly();
     String thumbnailFileName = thumbnailPrefix + objectName;

     if (objectName.startsWith(thumbnailPrefix)) {
          return; // Skip thumbnails to avoid recursion
     }

     // Download the file from S3
     s3.getObject(new GetObjectRequest(bucketName, objectName), new File(tempFileName));
```

Continued…

Authentication and access control not shown for brevity, but published best practices should be followed.

It parses the notification message to pull out some of the details we want to store in the DynamoDB table for display.

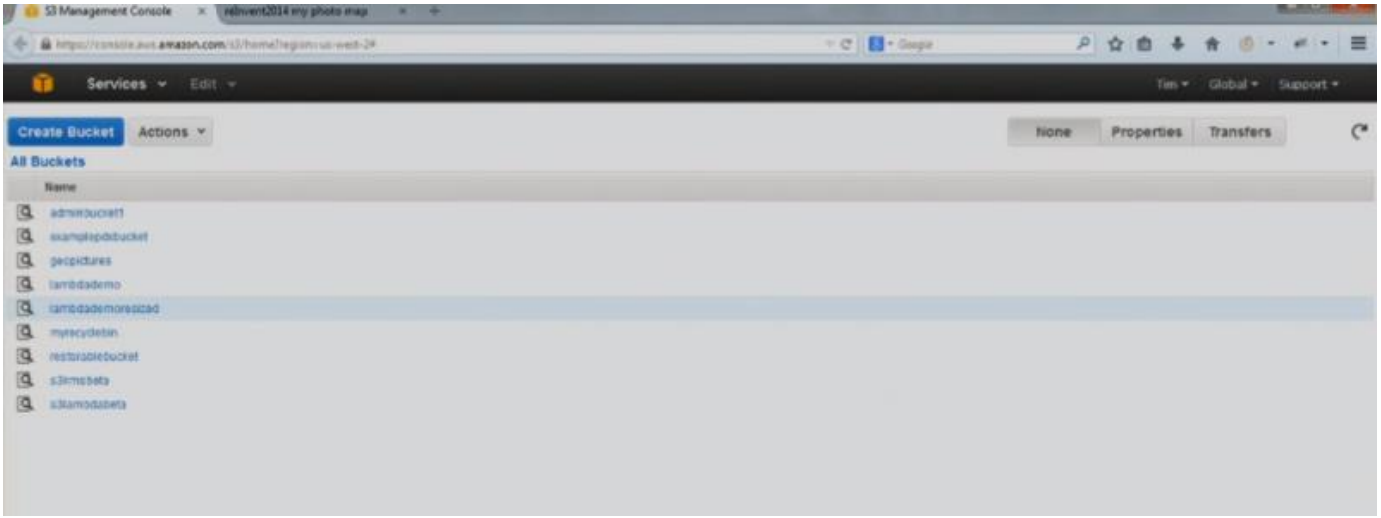
processMessage Java code - Continued

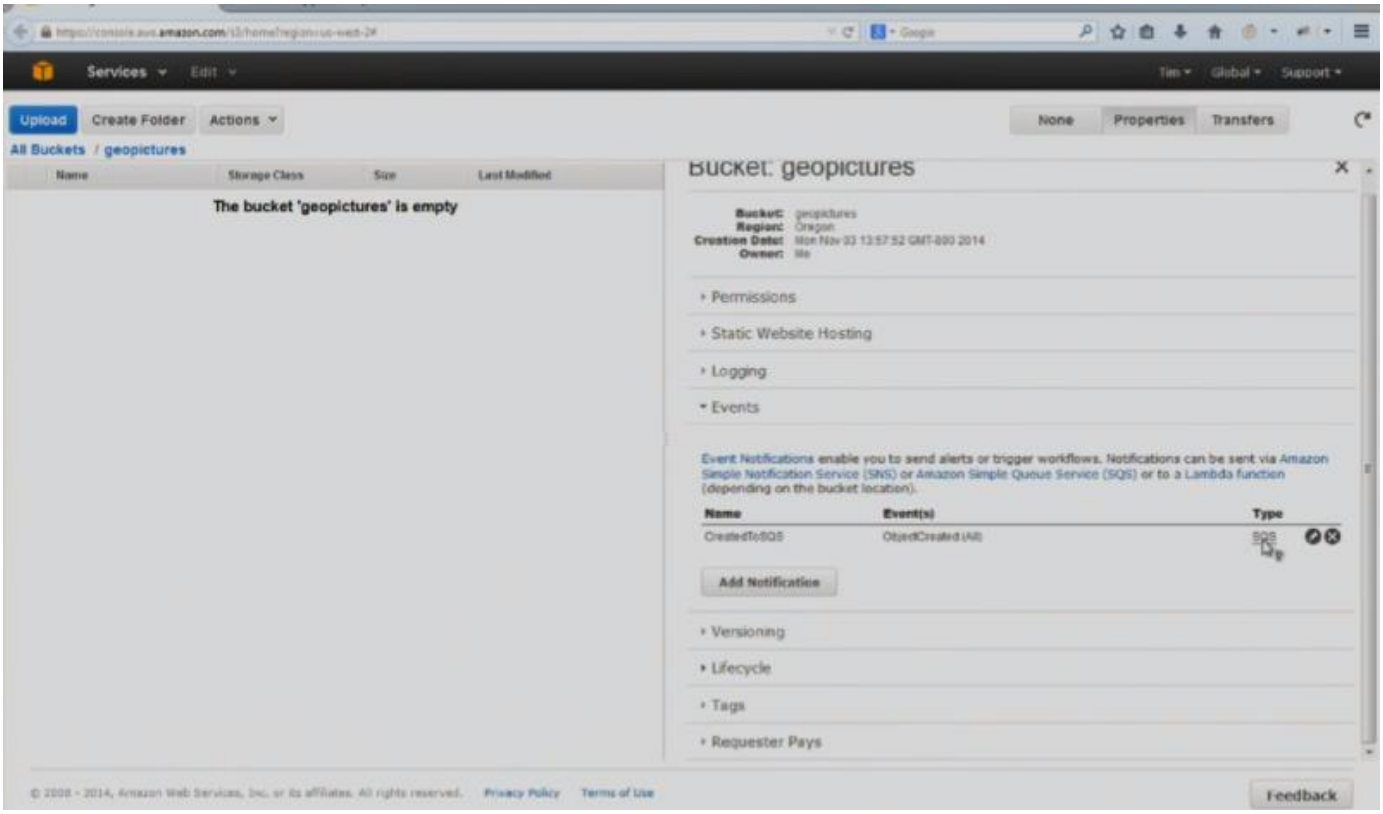
```
    // Read the GPS info from the file
    double[] gps = readGpsFromEXIF(tempFileName);

    // Put the photo info into DynamoDB for public website display
    putNewPhotoInDynamo (bucketName, objectName, gps, "http://" +
        sourceBucketName + ".s3.amazonaws.com/" + thumbnailFileName);

    // Generate a thumbnail and put it into S3
    generateThumbnail(tempFileName, thumbnailFileName);
    s3.putObject(sourceBucketName, thumbnailFileName, new File(thumbnailFileName));

    // Notify our web page to add it to the map
    sqs.sendMessage(sqs.createQueue(realTimeQueueName).getQueueUrl(), key);
}
```

Authentication and access control not shown for brevity, but published best practices should be followed.

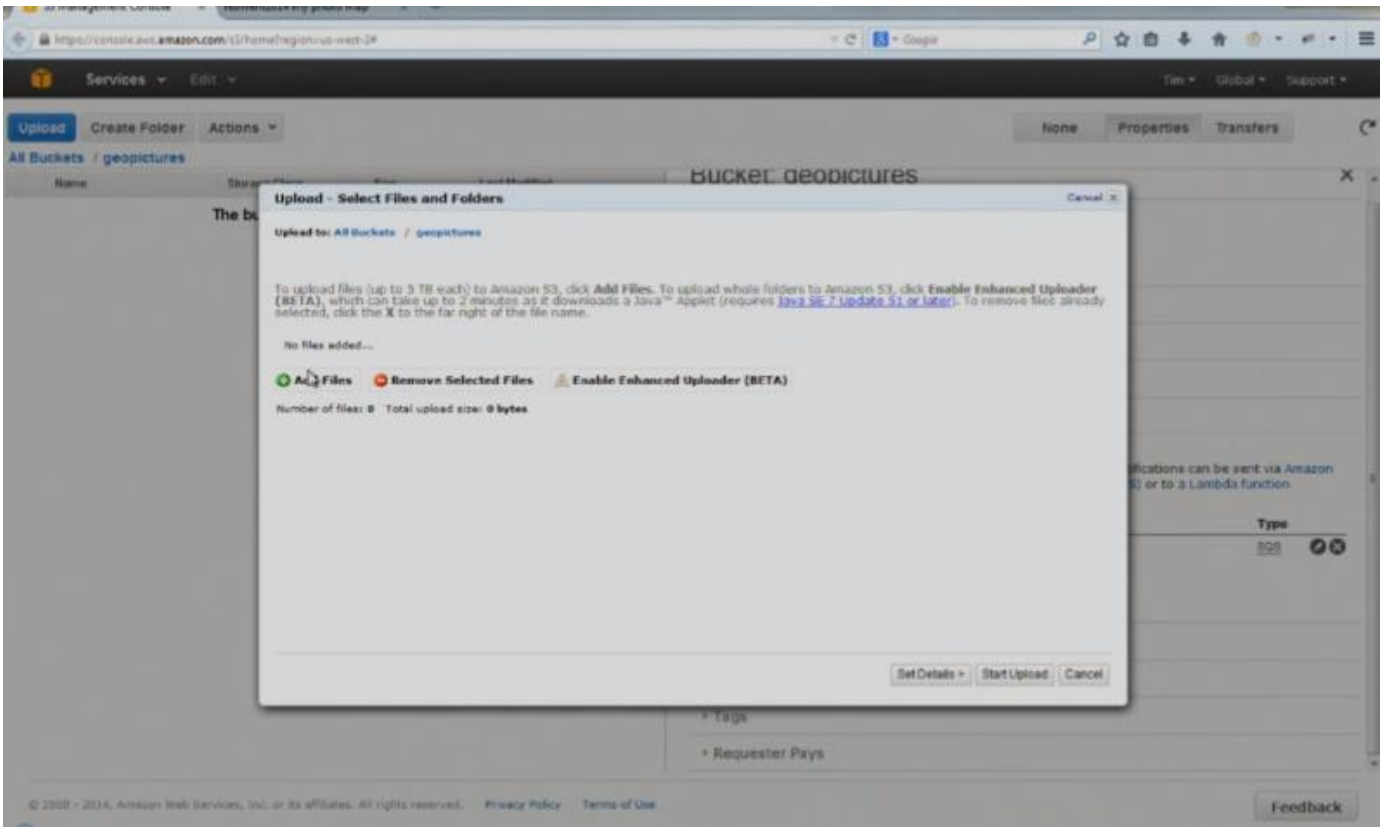
This 2nd SQS queue is used to trigger the web page that displays the results to the user so that the page knows there is a new object to display. Note that we are storing the thumbnail image in the same bucket as the original image

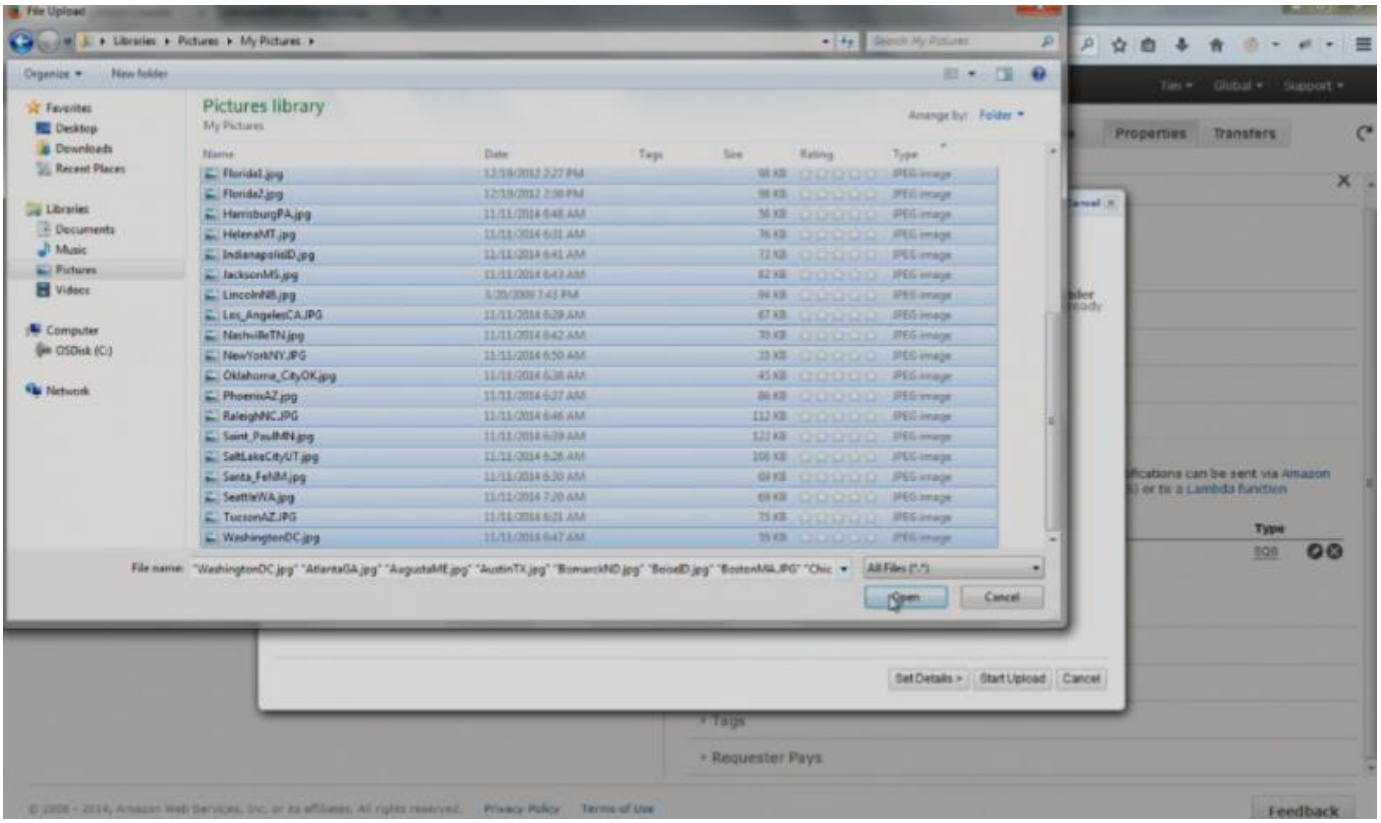
Demo

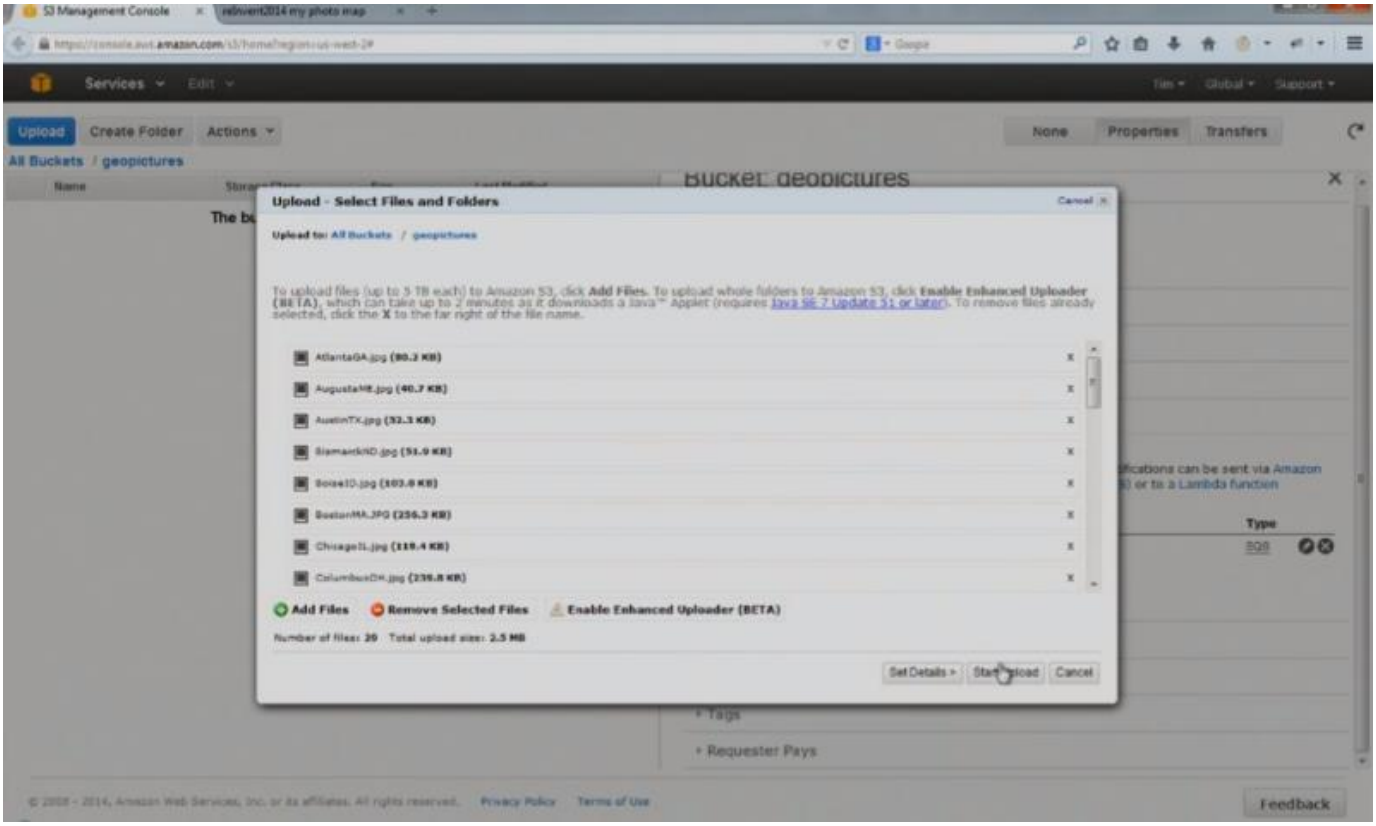




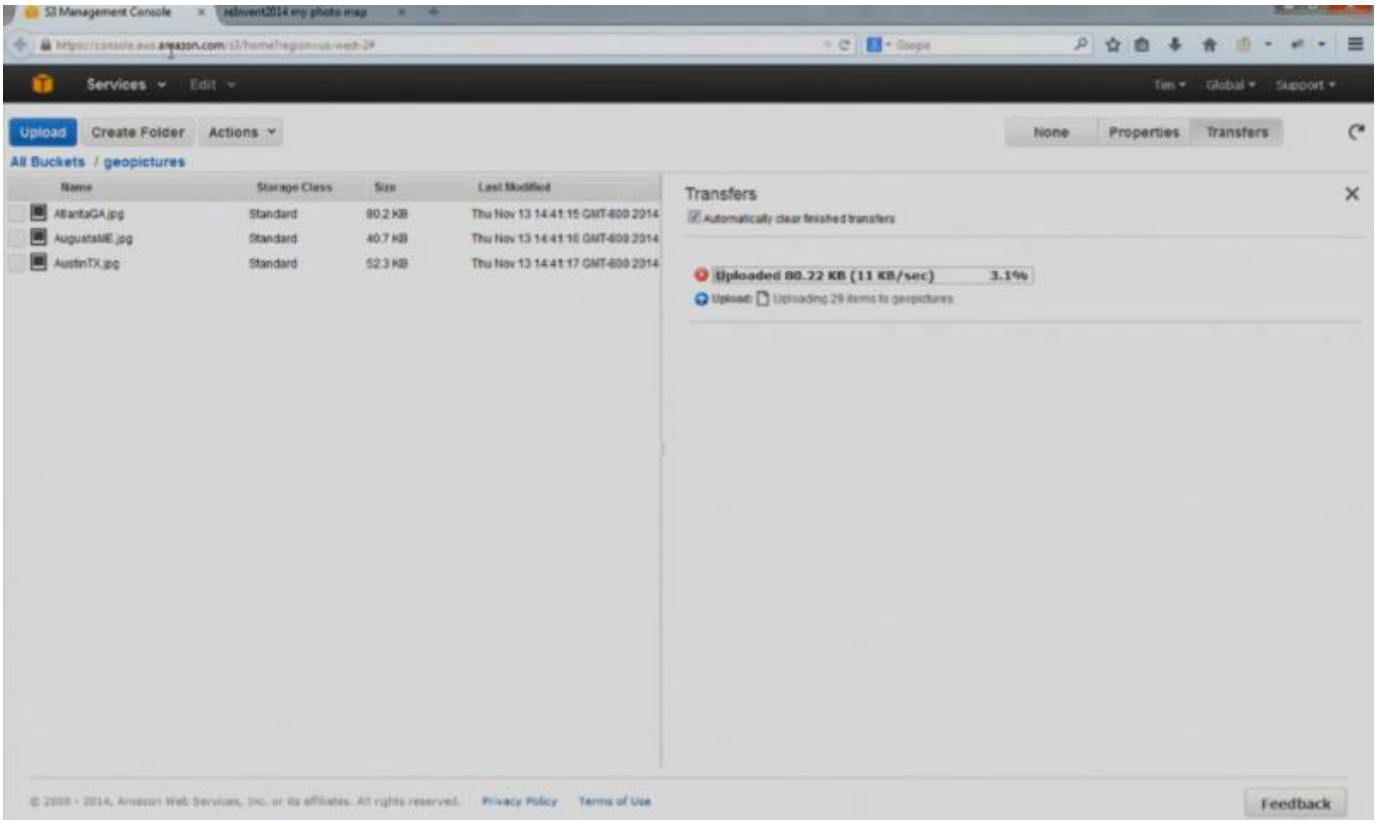
This is our source S3 bucket, it is set up to send notifications to the SQS queue

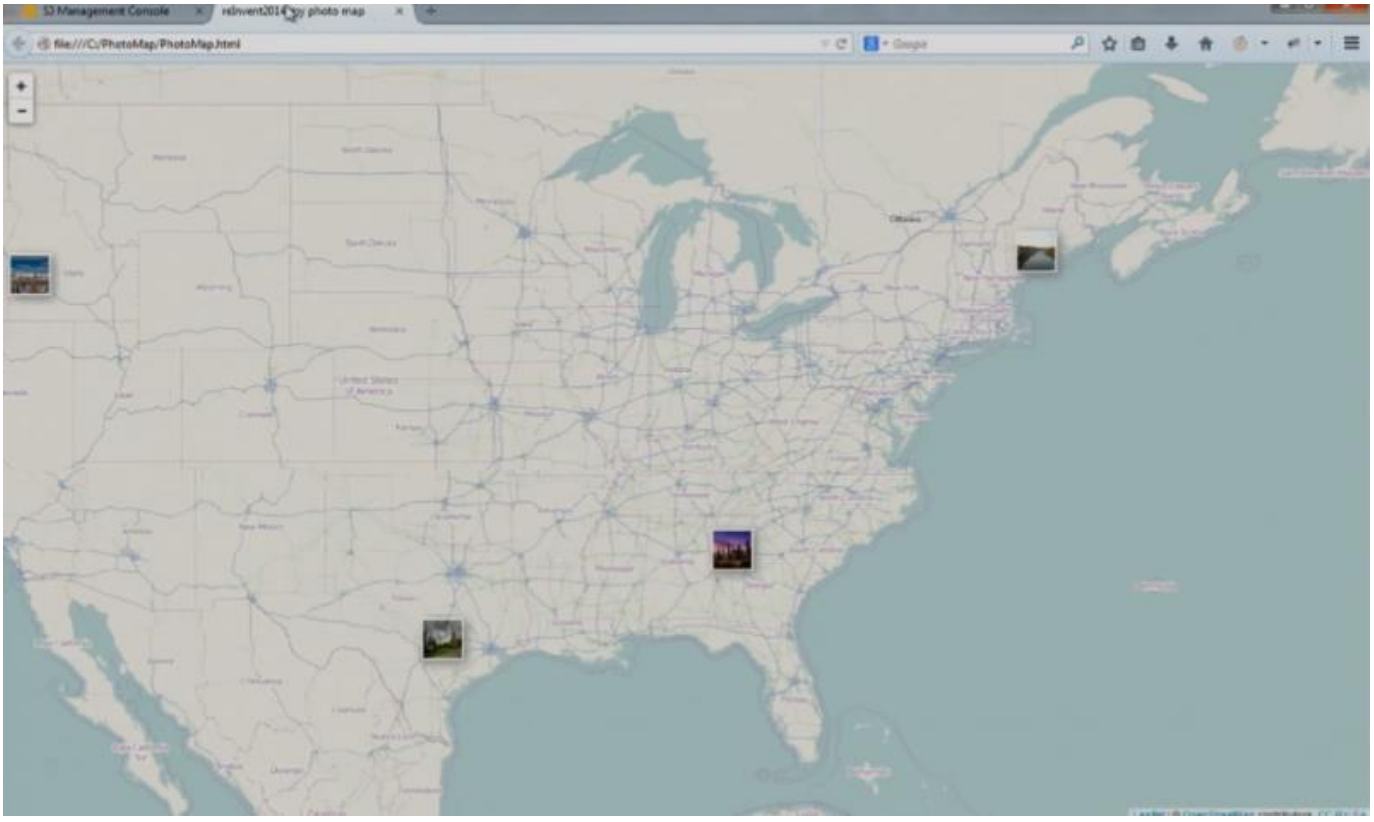


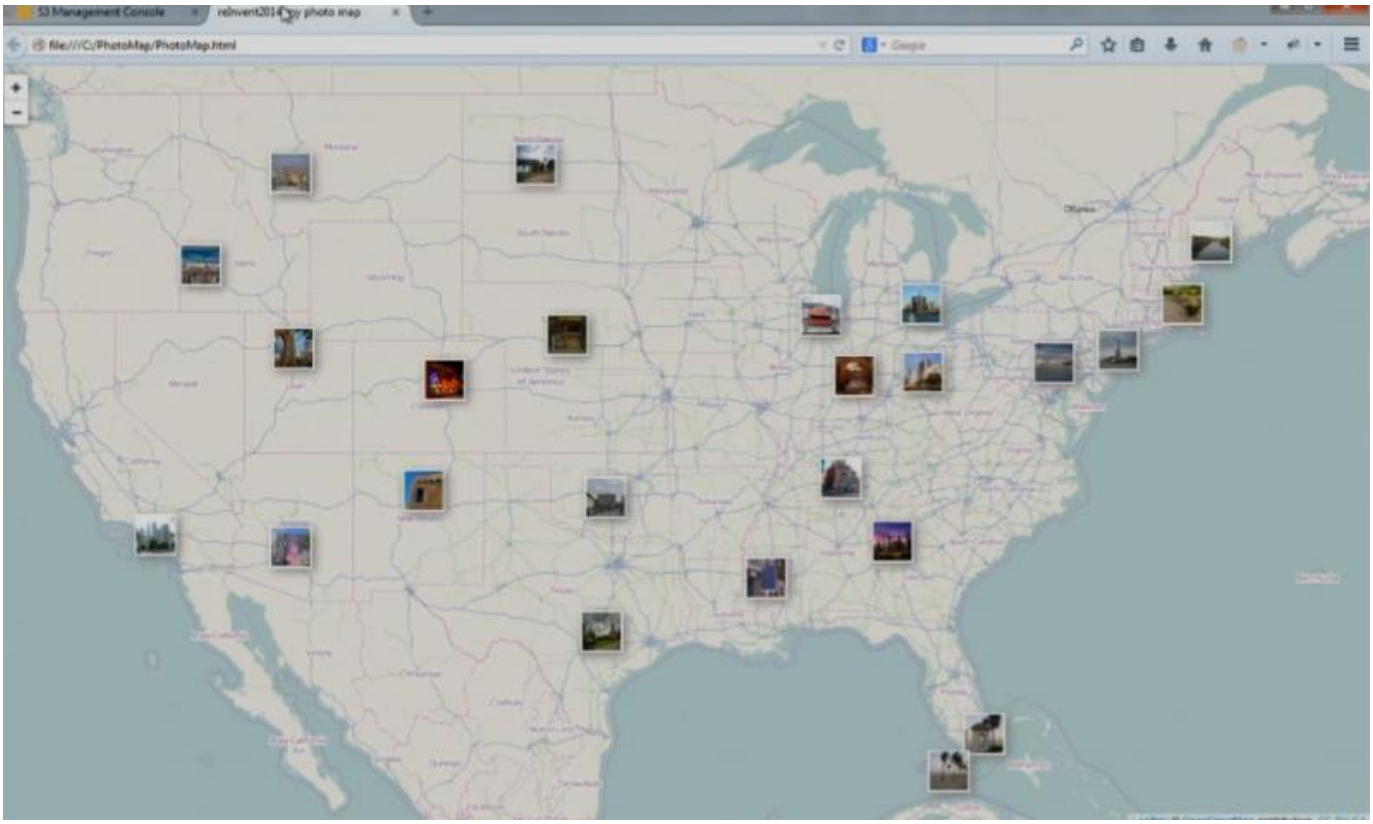




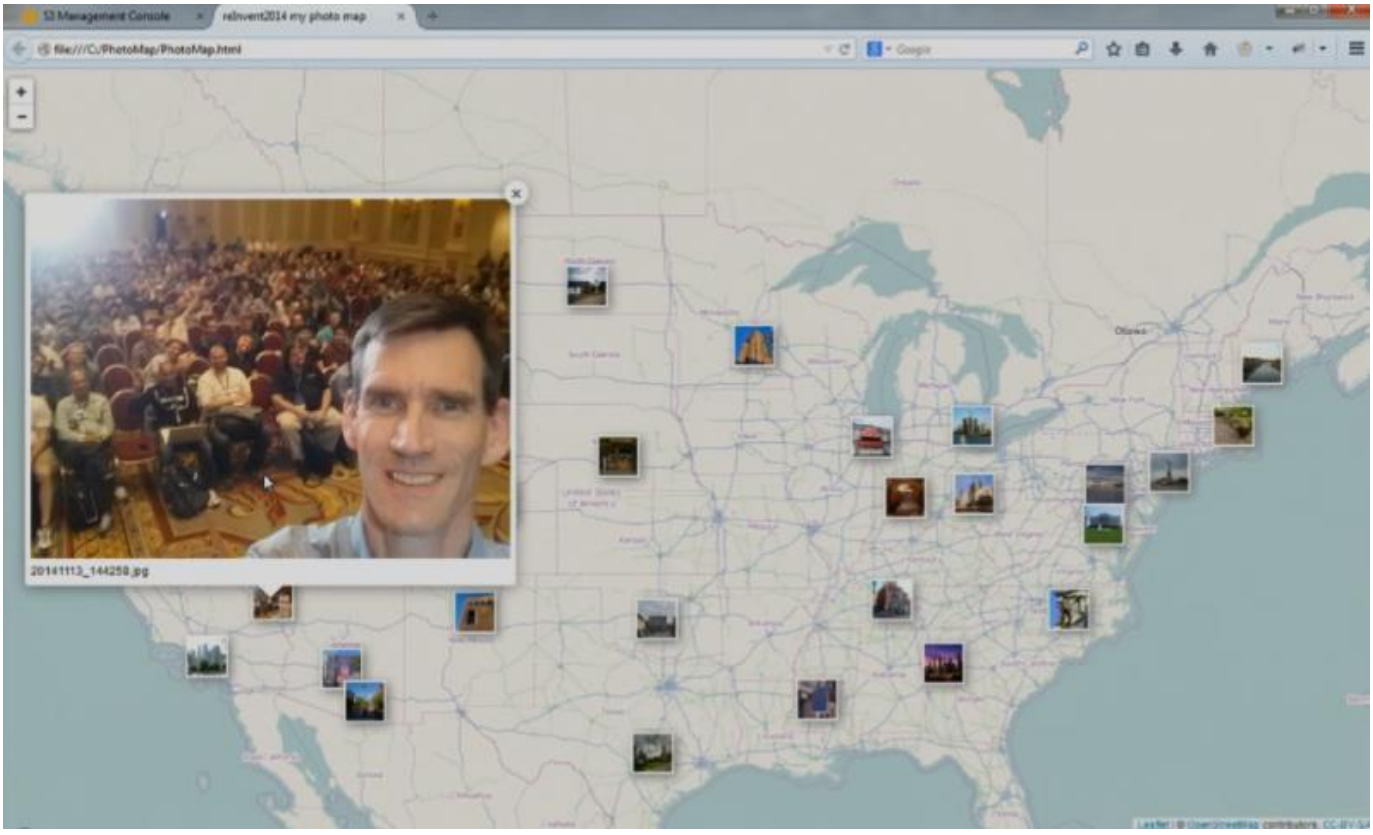
We then upload a bunch of files to the S3 bucket as above


S3 Management Console
reInvent2014 my photo map
Services
Edit
Upload
Create Folder
Actions
None
Properties
Transfers
All Buckets / geopictures
Name
Storage Class
Size
Last Modified
AtlantaGA.jpg
Standard
80.2 KB
Thu Nov 13 14:41:15 GMT-600 2014
AugustaME.jpg
Standard
40.7 KB
Thu Nov 13 14:41:16 GMT-600 2014
AustinTX.jpg
Standard
52.3 KB
Thu Nov 13 14:41:17 GMT-600 2014
Transfers
Automatically clear finished transfers
Uploaded 80.22 KB (11 KB/sec) 3.1%
Upload: Uploading 29 items to geopictures
Privacy Policy
Terms of Use
Feedback


S3 Management Console
file:///C:/PhotoMap/PhotoMap.html



The images are being displayed on the web page via the 2nd SQS message notifications as they become available and their thumbnails are being displayed

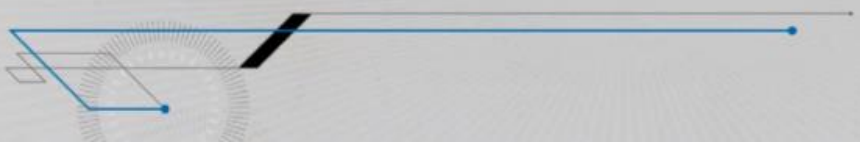


We can even do a real-time photo upload to the source bucket and see the thumbnail immediately as above

Key takeaways for event notifications

- Enables event-based computing
- Destination choices of Amazon SNS, Amazon SQS, or Lambda
- No additional charge from Amazon S3

Versioning + lifecycle policies



Versioning

- Protects from accidental overwrites and deletes with no performance penalty
- Generates a new version with every upload
- Allows easily retrieval of deleted objects or roll back to previous versions
- Three states of an Amazon S3 bucket
  - Default – Un-versioned
  - Versioning-enabled
  - Versioning-suspended

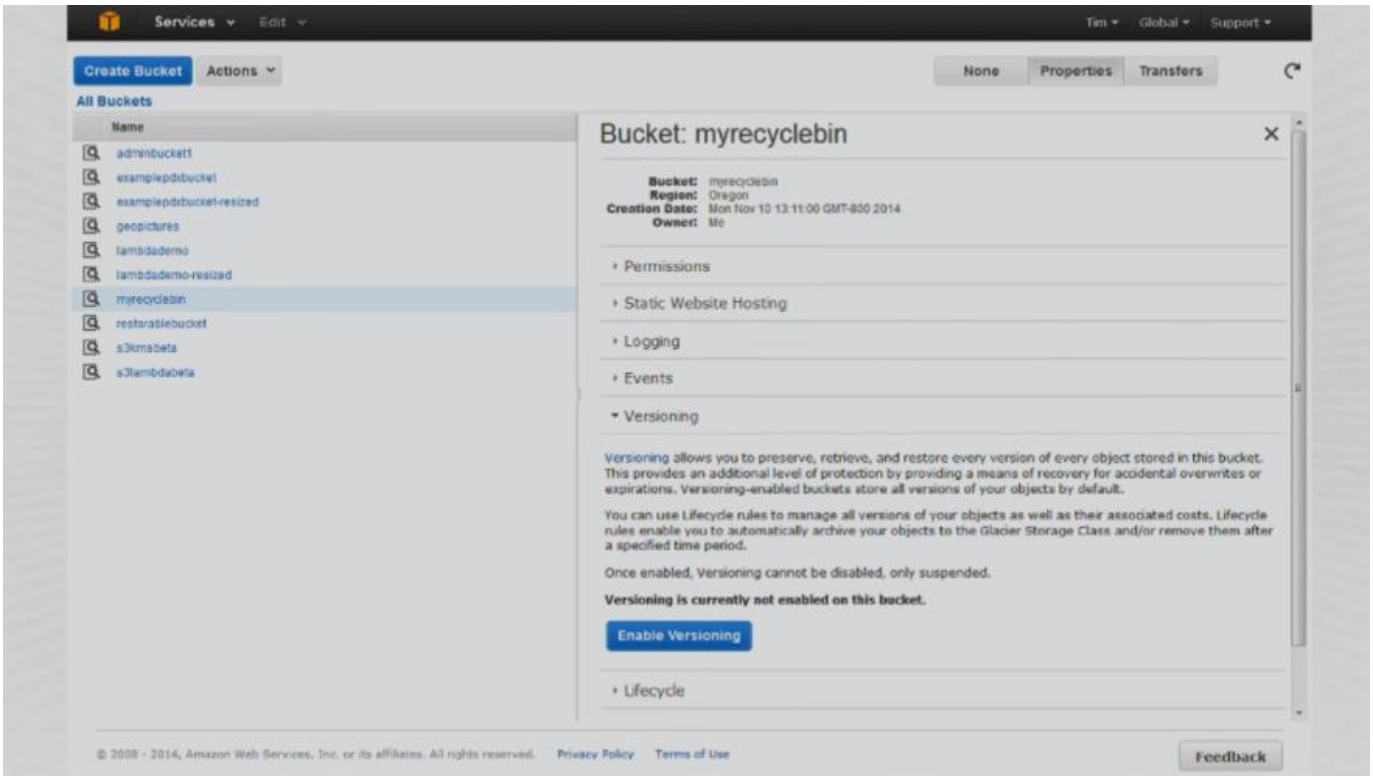
Lifecycle policies

- Provides automatic tiering to a different storage class and cost control
- Includes two possible actions:
  - Transition: archives to Amazon Glacier after a specified amount of time
  - Expiration: deletes objects after a specified amount of time
- Allows for actions to be combined – archive and then delete
- Supports lifecycle control at the prefix level

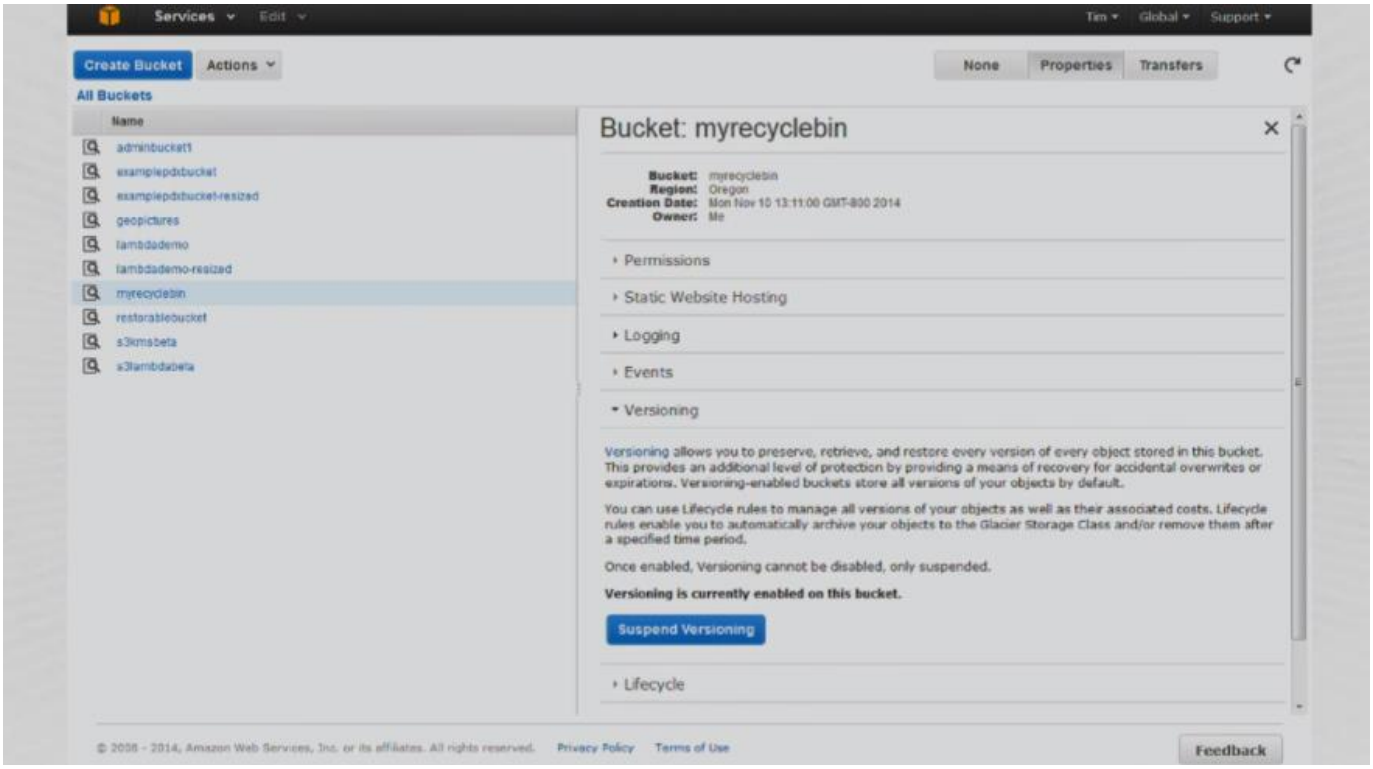
Recycle bin with automatic cleaning

Versioning → Recycle bin

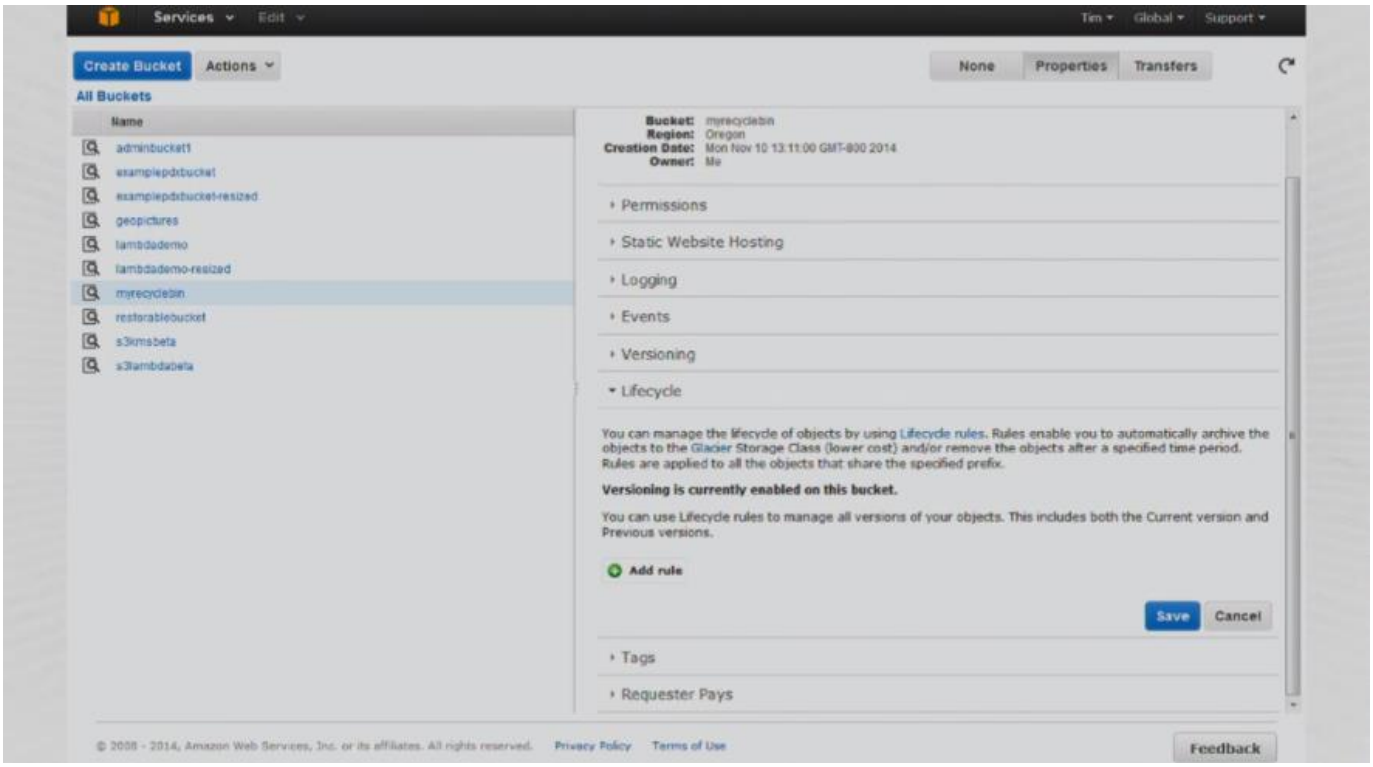
Lifecycle policies → Automatic cleaning



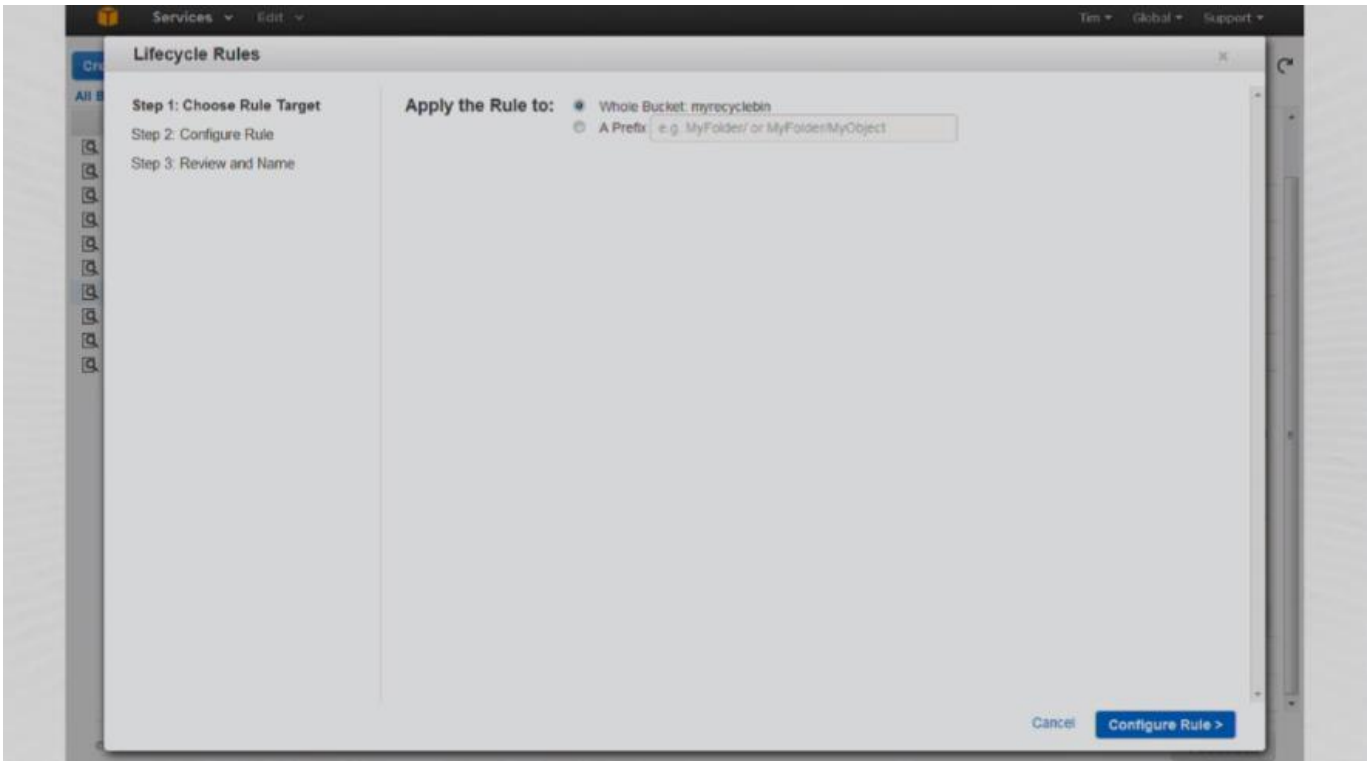
Let us now see how to set up this recycle bin functionality on your S3 buckets. You can simply click the Enable Versioning button to have it turn on for this bucket



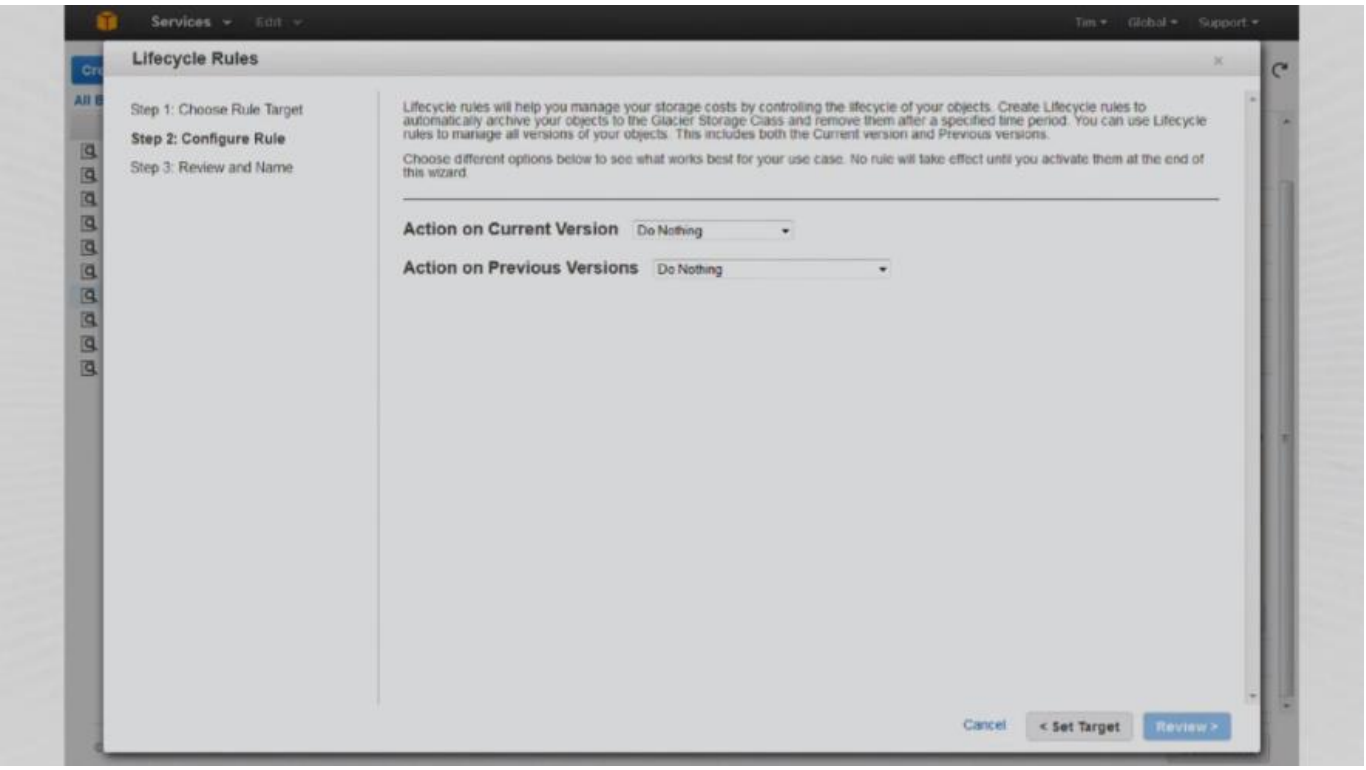
You can Suspend Versioning if you want later



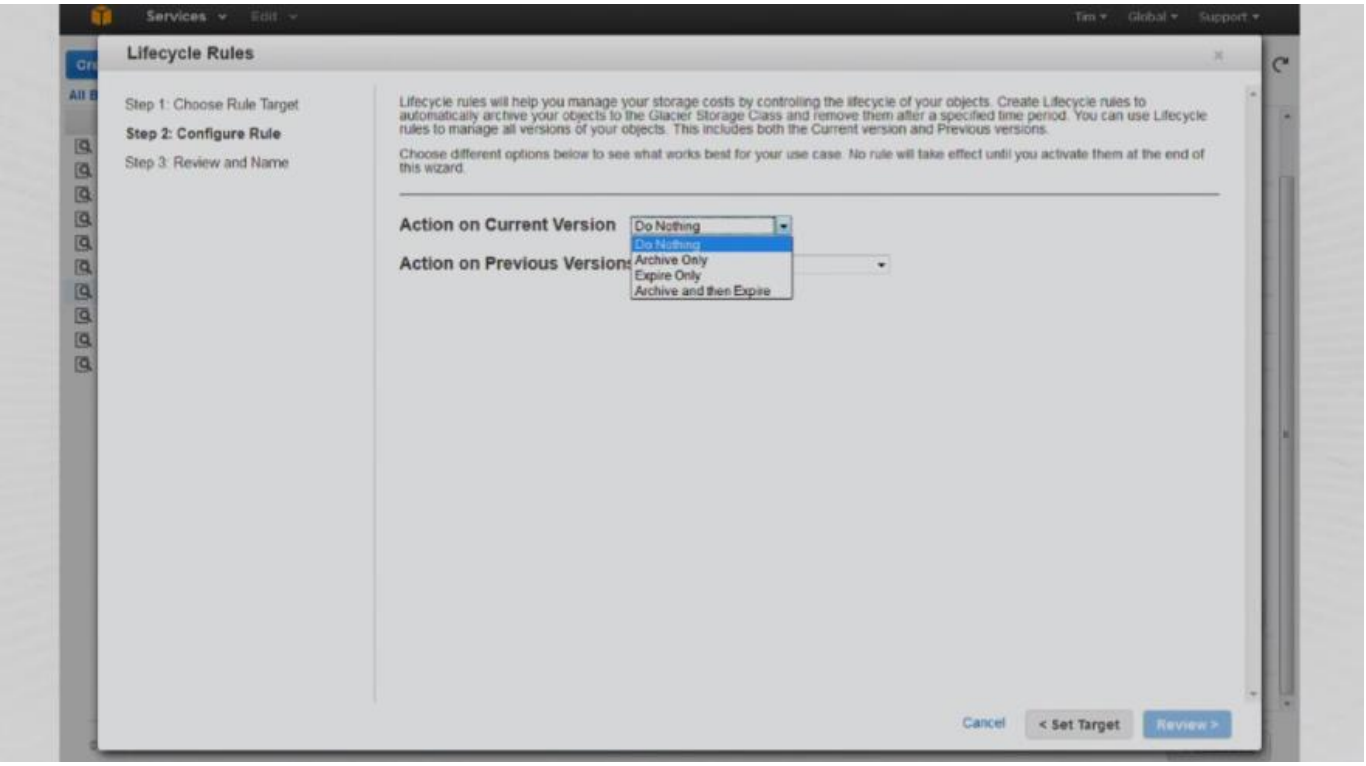
In the Lifecycle section for this bucket, we can turn on lifecycle rule for this bucket also by clicking the Add Rule link



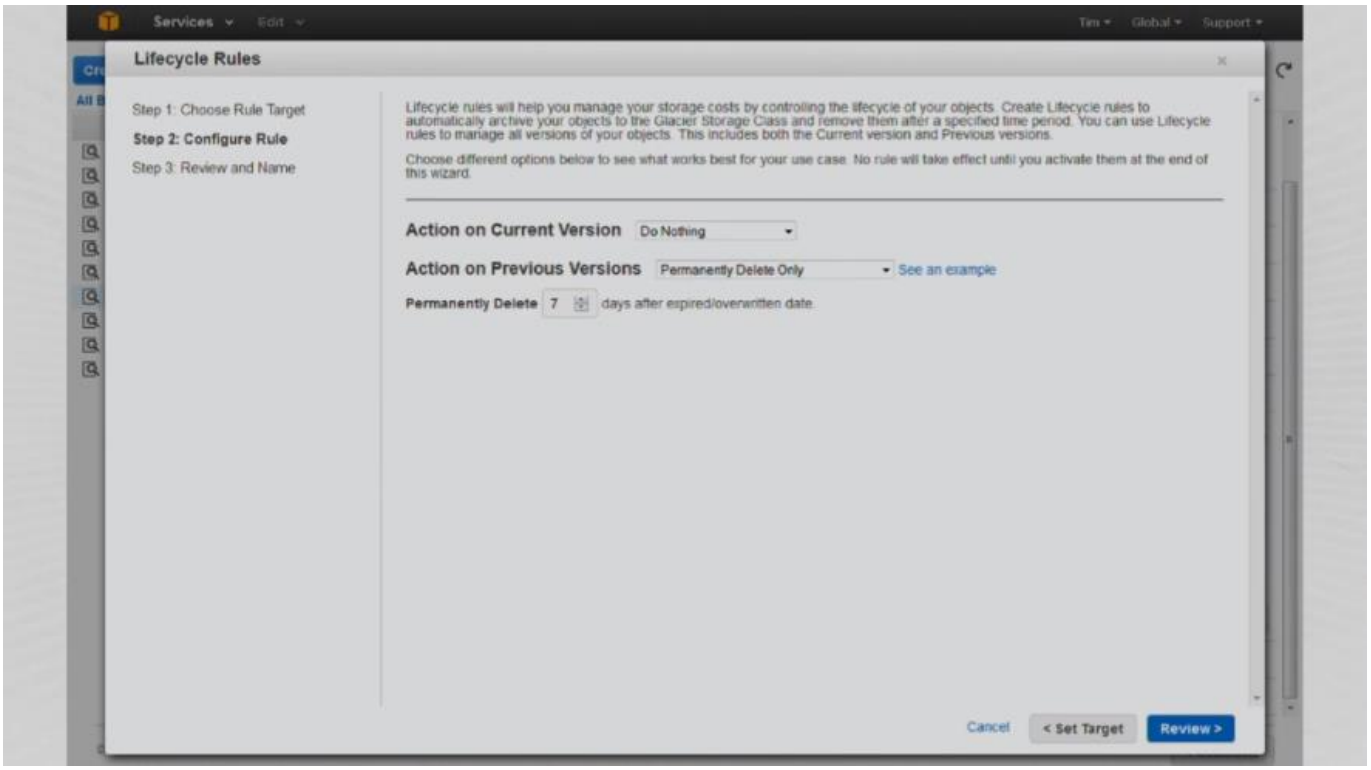
This gives us a 3-step process to define a lifecycle policy for this S3 bucket. We decide if we want the rule to apply to the entire s3 bucket or just a prefix of it

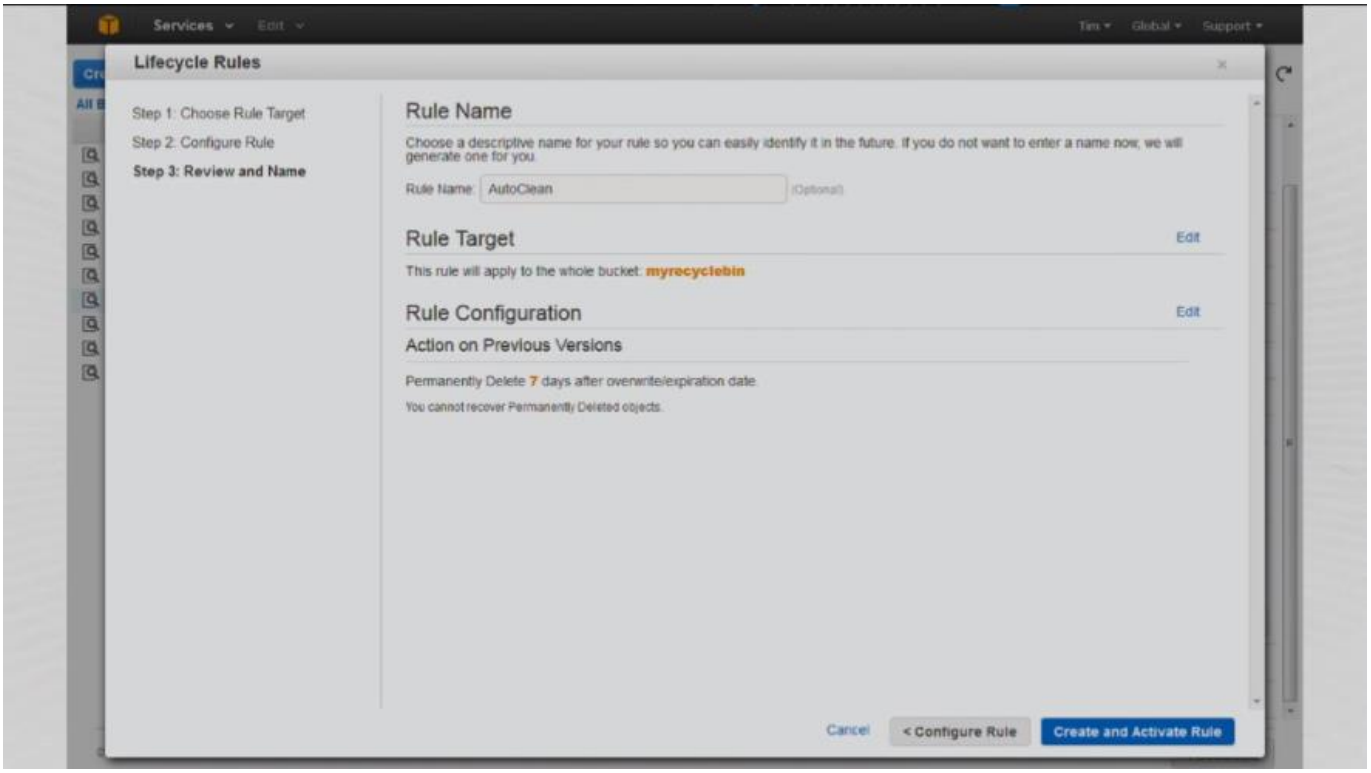


Then we chose what action we want to take both on the current version and on all the previous versions if they exist.

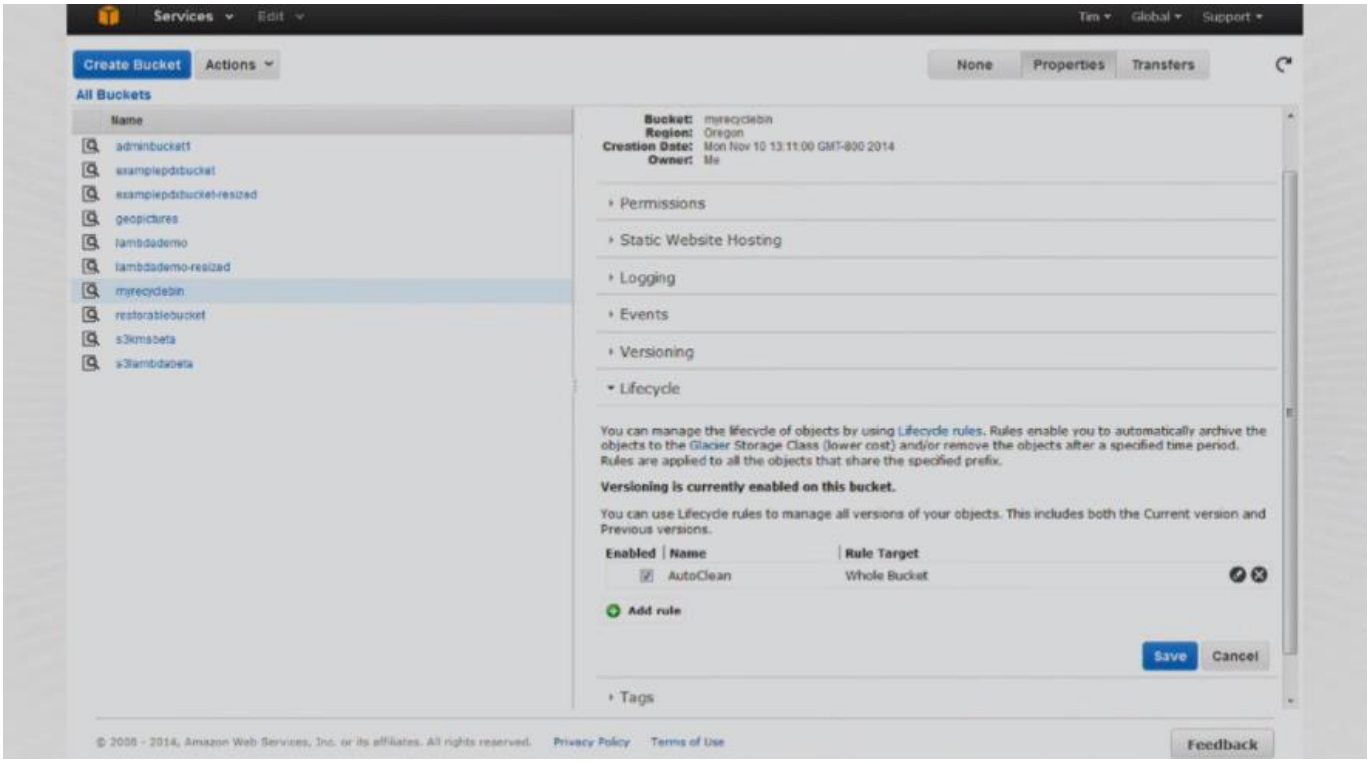


We have 4 choices listed above

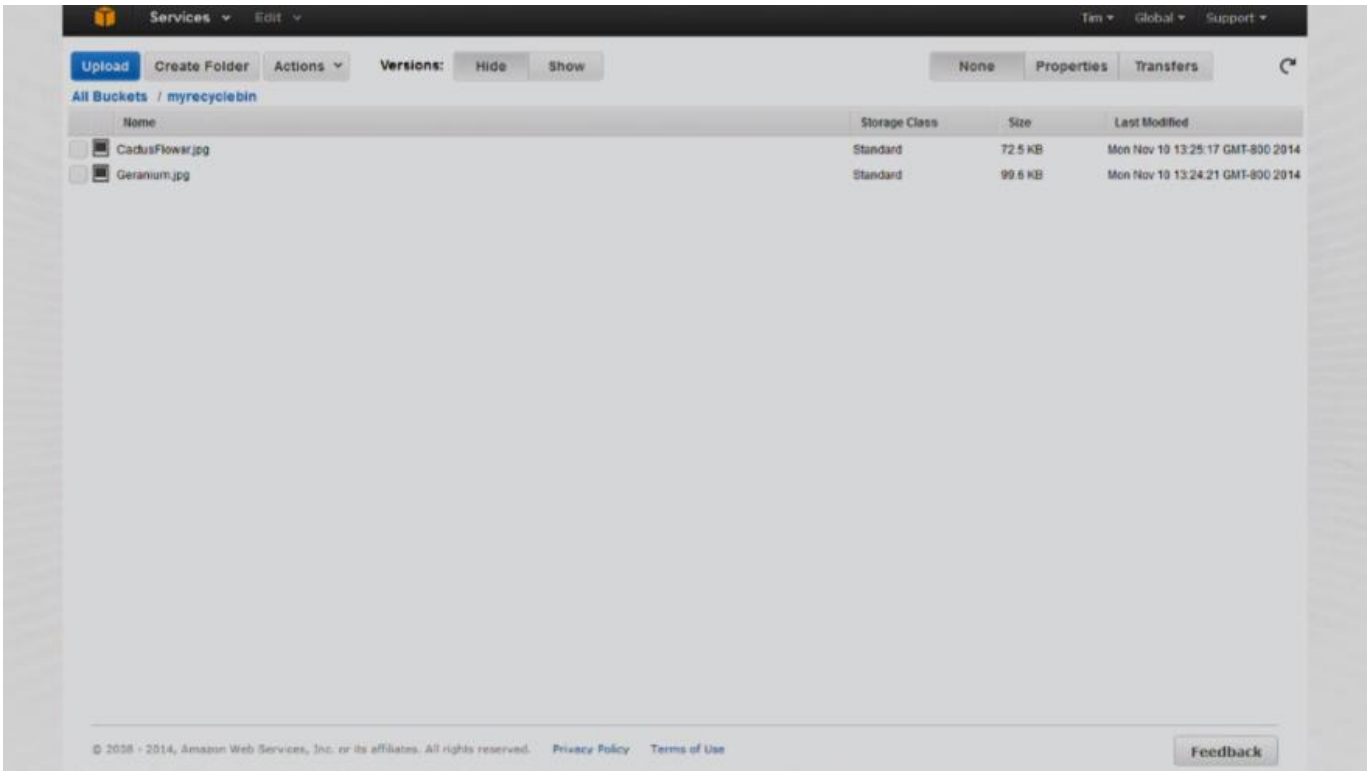




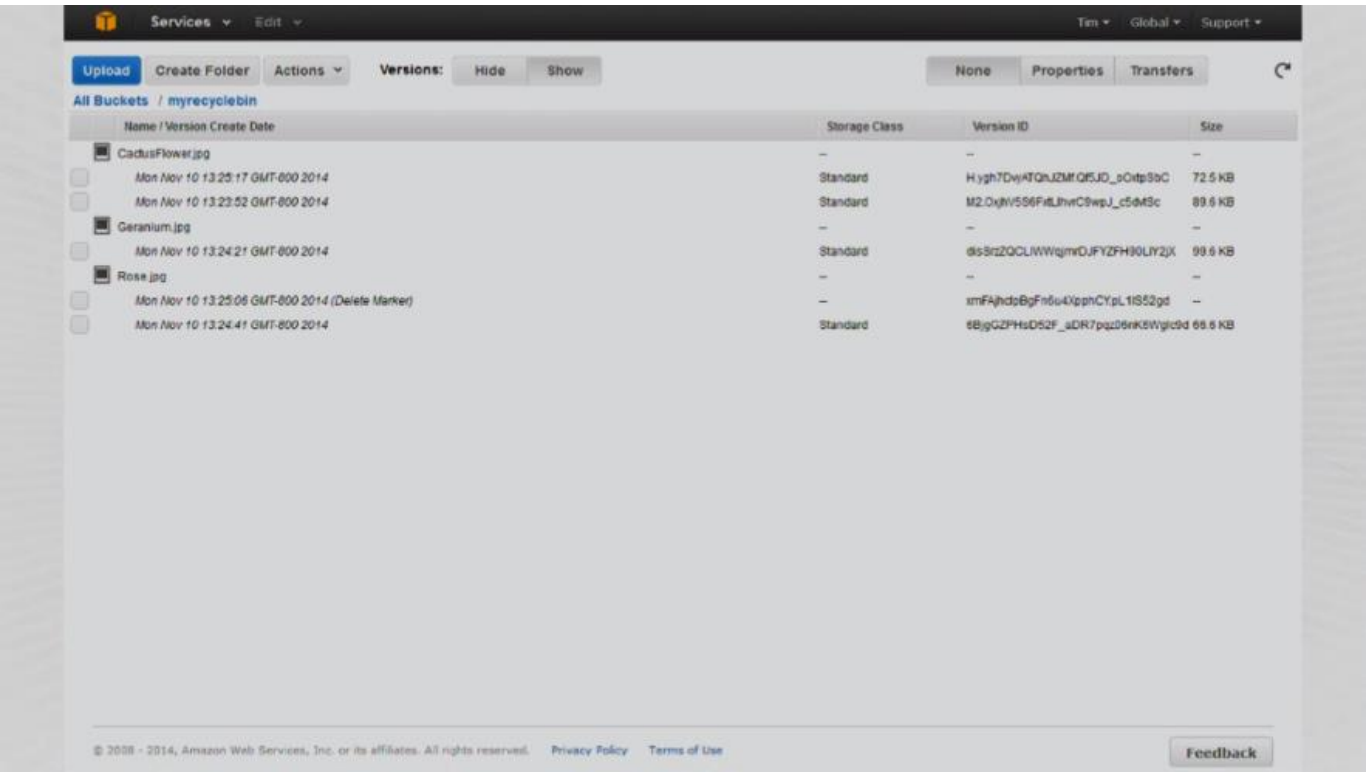
We then give the rule a name, review it, then save it



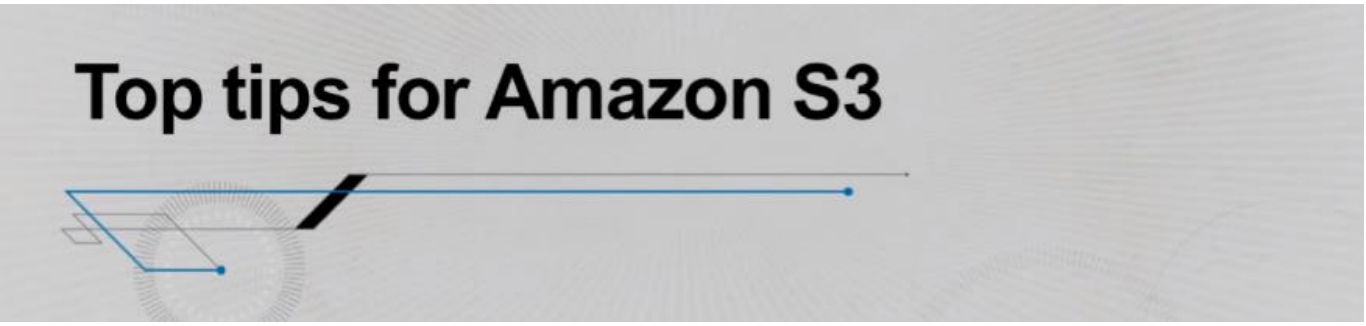
We now have a new lifecycle policy created for this S3 bucket.

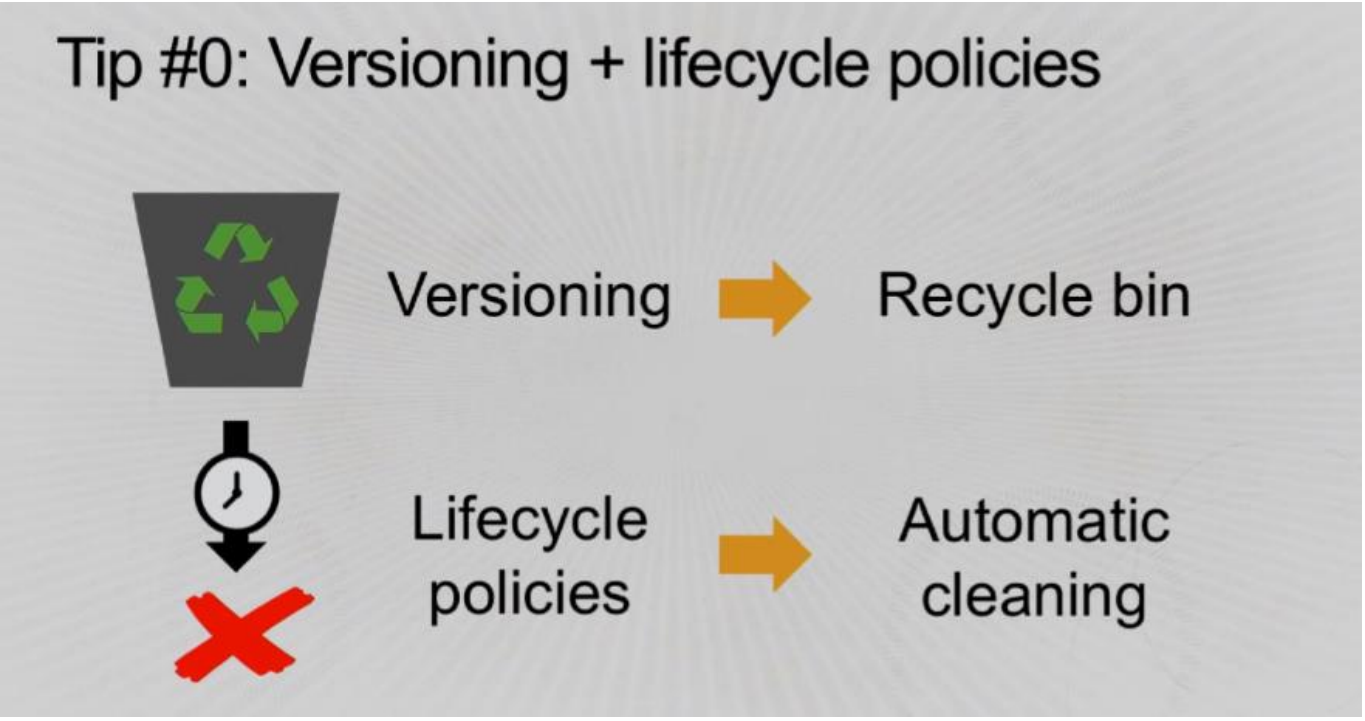


We can see what versioning looks like in the console, we have a bucket with 2 objects



We can click the Show button to see more information about the objects in our versioned S3 bucket





Tip #1: 3 server-side encryption options

- SSE with Amazon S3 managed keys
  - "Check-the-box" to encrypt your data at rest
- SSE with customer provided keys
  - You manage your encryption keys and provide them for PUTs and GETS
- New SSE with AWS Key Management Service managed keys
  - Keys managed centrally in AWS KMS with permissions and auditing of usage

Tip #2: Detailed billing reports

- Provide object counts, storage GB, requests, and data transfer usage down to the bucket level
- Turn them on via the Preferences page in the Billing and Cost Management console
- Delivered to an Amazon S3 bucket you specify and downloadable from the Billing and Cost Management console

Tip #3: Restricting deletes

- Bucket policies can restrict deletes
- For additional security, enable MFA (multi-factor authentication) delete, which requires additional authentication to:
  - Change the versioning state of your bucket
  - Permanently delete an object version
- MFA delete requires both your security credentials and a code from an approved authentication device

Tip #4: Maximizing performance

- Use a key naming scheme with randomness at the beginning for high TPS
  - Most important if you regularly exceed 100 TPS on a bucket
  - Avoid starting with a date
  - Consider adding a hash or reversed timestamp (ssmmhhddmmyy)
- Multipart upload provides parallelism
  - Allows you to upload a single object as a set of parts
  - Enables pausing and resuming, and beginning before the total size is known
  - Encouraged for objects larger than 100MB; required above 5GB

AWS re:Invent

Please give us your feedback on this session.
Complete session evaluations and earn re:Invent swag.

SDD413

http://bit.ly/awsevals

amazon web services

Join the conversation on Twitter with #reinvent