Amazon Web Services

Certified Developer – Associate Level

Introduction

Who should take this exam?

- Exams cover different parts of services and some entirely different services
  - A lot of concepts overlap from AWS Certified Solutions Architect exam

- This exam focuses on using Amazon Web Services to develop applications
  - Understanding permission control
  - Creating highly available and fault tolerant applications
  - Understanding which services make more sense depending on needs

- Users looking to support or create applications on AWS
  - Developers
  - System Operators

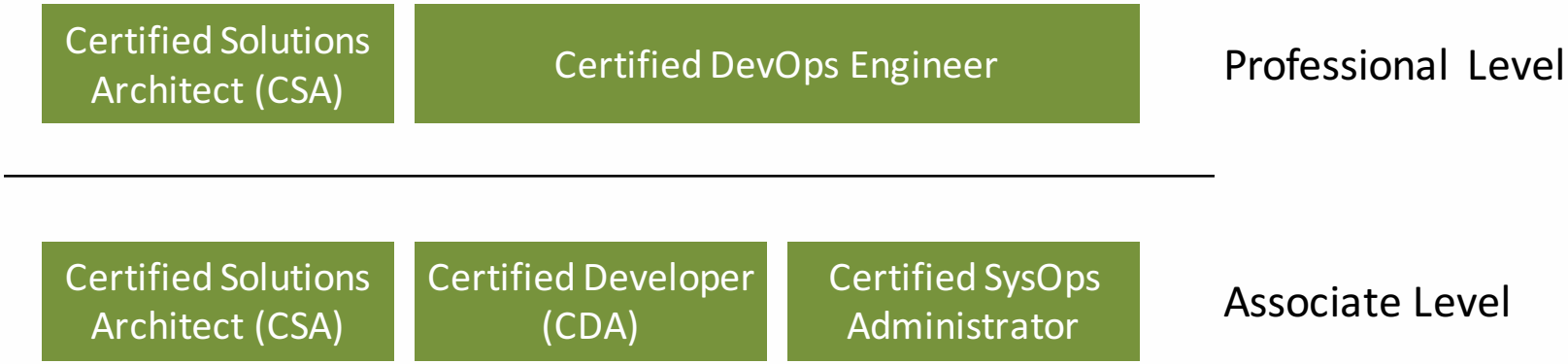
How to prepare for the exam

- Follow the video lessons:
  - Conceptual
  - Hands-on

- Use Live! Labs
  - Apply and practice what you learn for better retention

- Memorize important concepts and key terms

Certification Roadmap



Amazon Web Services

Certified Developer – Associate Level

IAM Roles and Access Keys

IAM Roles

- AWS identity
  - Has permission policies attached
  - Policies determine what the identity can and cannot do in AWS

- Intended to be assigned to multiple users or services

- Does not have password or access key credentials
  - Credentials are given to users assigned to a role

- Useful for giving access to:
  - Users
  - Applications
  - Services

Development Scenarios

- EC2 instances with applications on them
  - The applications need access to services (like S3)
  - Instead of manually assigning credentials (and rotating them), use IAM roles

- Grant access to users from one AWS account to another

- Allow a mobile app to use AWS resources
  - Rotating keys could be a nightmare
  - Keys could potentially be extracted

API Credentials with Access Keys

- Used to sign requests
  - The AWS SDKs use access keys
  - The Command Line Interfaces (CLIs)
- Can be disabled and deleted but not retrieved
- You can have temporary access keys which expire
- Useful when connecting from outside of AWS (like your computer or an application outside of AWS)

Amazon Web Services

Certified Developer – Associate Level

S3 Developer Requirements

S3 Limits & Restrictions

- An AWS account can own up to 100 S3 buckets
- No limit to the number of objects that can be stored in a bucket
- Bucket name restrictions:
  - Must be unique
  - Should comply with DNS naming conventions
  - Minimum of 3 and no more than 63 characters long
  - Can only contain lowercase letters, numbers, periods and hyphens
  - Must start with a letter or number, NOT period/hyphen
  - Periods and hyphens cannot follow each other (i.e linuxacdemy-.com)
  - Must not be an IP address format (i.e 10.2.1.2)

S3 Regions

- You can choose a region where Amazon S3 will store your buckets
  - Optimizes latency
  - Minimizes costs
  - Addresses regulatory requirements
- Objects stay in a region unless explicitly transferred

S3 Objects

- Object size:
  - Minimum of 0 bytes
  - Maximum of 5 terabytes

- Objects larger than 5 GB require using the Multipart Upload API
  - Can upload independently, in any order, and in parallel
  - If any part fails to upload, you can retransmit that part
  - You can pause and resume uploads
  - You can upload objects as they are being created
  - Object is reassembled after calling CompleteMultiPartUpload API

- Multipart Upload is recommend for files larger than 100MB

- Objects can be encrypted before being saved to disk, and decrypted when downloaded

S3 data consistency

- Buckets in all Regions have:
  - Read-after-write consistency for PUTS of new objects
  - Eventual consistency for overwrite PUTS and DELETES
- Read-after-write consistency lets you retrieve objects immediately after creating them
- Eventual consistency may return old data

Eventually Consistent Read	Consistent Read
Stale reads possible	No stale reads
Lowest read latency	Potential higher read latency
Highest read throughput	Potential lower read throughput

S3 Performance

- Amazon S3 can scale to very high request rates
- Bursts in the number of requests per second
  - > 300 PUT/LIST/DELETE
  - > 800 GET
  - Contact Amazon to prepare and avoid temporary limits
- Consistent high number of requests per second
  - > 100 PUT/LIST/DELETE
  - > 300 GET
  - Follow the best practice guidelines to avoid overwhelming the I/O capacity of a partition

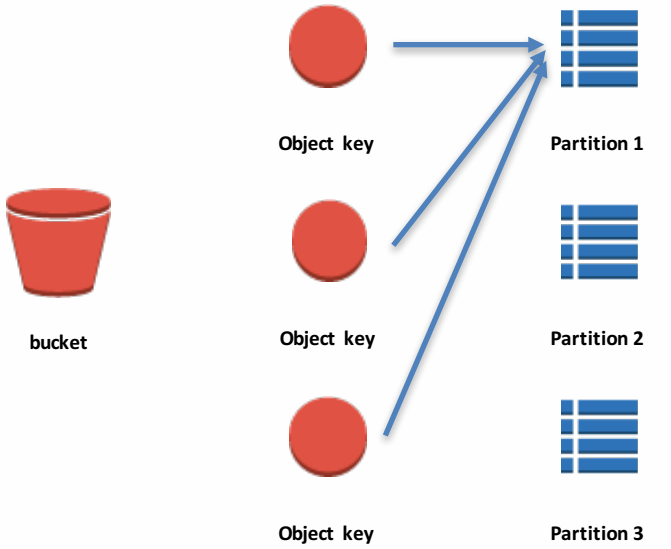
S3 Performance – Workloads with a mix of request types

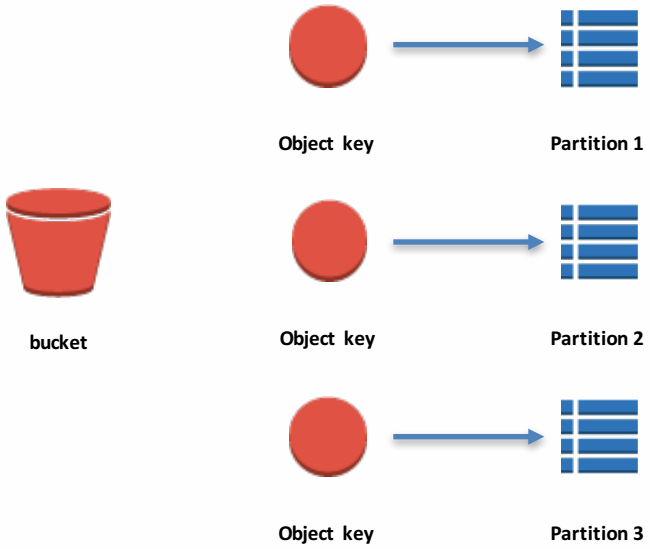
- This type of sequence pattern in names causes performance problems

```
examplebucket/2013-26-05-15-00-00/cust1234234/photo1.jpg
examplebucket/2013-26-05-15-00-00/cust3857422/photo2.jpg
examplebucket/2013-26-05-15-00-00/cust1248473/photo2.jpg
examplebucket/2013-26-05-15-00-00/cust8474937/photo2.jpg
examplebucket/2013-26-05-15-00-00/cust1248473/photo3.jpg
...
examplebucket/2013-26-05-15-00-01/cust1248473/photo4.jpg
examplebucket/2013-26-05-15-00-01/cust1248473/photo5.jpg
examplebucket/2013-26-05-15-00-01/cust1248473/photo6.jpg
examplebucket/2013-26-05-15-00-01/cust1248473/photo7.jpg
...
```

- Amazon S3 has an index of object key names in each AWS region
  - Object keys are stored across multiple partitions in that index
  - The key name is used to decide which partition the key is stored in
- Using sequential patterns stores a large number of keys in the same partition

```
examplebucket/2013-26-05-15-00-00/cust1234234/photo1.jpg
examplebucket/2013-26-05-15-00-00/cust3857422/photo2.jpg
examplebucket/2013-26-05-15-00-00/cust1248473/photo2.jpg
examplebucket/2013-26-05-15-00-00/cust8474937/photo2.jpg
examplebucket/2013-26-05-15-00-00/cust1248473/photo3.jpg
...
examplebucket/2013-26-05-15-00-01/cust1248473/photo4.jpg
examplebucket/2013-26-05-15-00-01/cust1248473/photo5.jpg
examplebucket/2013-26-05-15-00-01/cust1248473/photo6.jpg
examplebucket/2013-26-05-15-00-01/cust1248473/photo7.jpg
...
```




Object key
Partition 1
bucket
Object key
Partition 2
Object key
Partition 3

S3 Performance – Solving the sequence pattern problem

- Example #1

```
examplebucket/232a-2013-26-05-15-00-00/cust1234234/photo1.jpg
examplebucket/7b54-2013-26-05-15-00-00/cust3857422/photo2.jpg
examplebucket/921c-2013-26-05-15-00-00/cust1248473/photo2.jpg
examplebucket/ba65-2013-26-05-15-00-00/cust8474937/photo2.jpg
examplebucket/8761-2013-26-05-15-00-00/cust1248473/photo3.jpg
examplebucket/2e4f-2013-26-05-15-00-01/cust1248473/photo4.jpg
examplebucket/9810-2013-26-05-15-00-01/cust1248473/photo5.jpg
examplebucket/7e34-2013-26-05-15-00-01/cust1248473/photo6.jpg
examplebucket/c34a-2013-26-05-15-00-01/cust1248473/photo7.jpg
...
```

- Make key names random using a Hex Hash prefix
  - Use a hash (like MD5) of a character sequence
  - Pick a specific number of characters from that hash to use as the prefix

S3 Performance – Solving the sequence pattern problem

- Example #2

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



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

- Reversing key name strings
- The picture on the left creates this partition:
  - **examplebucket/2**
- The picture on the right creates these partitions:
  - **examplebucket/7**
  - **examplebucket/8**
  - **examplebucket/9**

S3 Performance – GET-intensive workloads

- Follow similar guidelines we just went over
- Use Amazon CloudFront
  - Distributes content with low latency and high transfer rate
  - Caches objects
  - Fewer direct requests to S3

S3 – Error Handling

- Understanding common errors
- Error codes are generally handled with HTTP responses
  - 404 Not Found
  - 403 Forbidden
  - 400 Bad Request
  - 409 Conflict
  - 500 Internal Server Error

- http://docs.aws.amazon.com/AmazonS3/latest/API/ErrorResponses.html

Amazon Web Services

Certified Developer – Associate Level

Static Website Hosting

Amazon S3 – Static Website Hosting

- S3 allows you to host static HTML files
  - Can specify index file
  - Can specify custom error file
- Supports custom domains and redirects
  - Amazon S3 gives a default URL
  - Redirect from www.linuxacademy.com to linuxacademy.com
  - Route 53 integration for custom domains
  - Bucket name must match domain name

Amazon S3 - URL

- Every hosted bucket receives its own URL
  - Think in terms of namespaces

<bucket-name>.s3-website-<AWS-region>.amazonaws.com

Amazon S3 - Files

- Index document
- Error document
  - Supports custom error documents for 4XX clss errors
- Custom redirect rules
  - Supports various types of simple and advances redirects

Amazon Web Services

Certified Developer – Associate Level

IAM and Bucket Policies

Amazon S3 – Bucket Permissions

- Can use bucket and/or user policies
  - Resource-based policies
  - User policies
- Bucket permissions specify:
  - Who is allowed to access resources
  - What that user can do with those resources
- AWS gives full permissions to the owner of a resource (bucket, object)
- Resource owners can grant access to others, even cross-account
  - The bucket owner paying bills can deny access or modify objects regardless of who owns them

Amazon S3 – Bucket Policies

- Resource-based policy
- Uses a JSON file attached to the resource
- Can grant other AWS accounts or IAM users permission for the bucket and objects inside
- **Should be used** to manage cross-account permissions for all Amazon S3 permissions
- Limited to 20 KB in size

Amazon S3 – Bucket Policy Example

```
{
 "Version":"2012-10-17",
 "Statement":[
  {
   "Sid": "AddObject",
   "Effect": "Allow",
   "Principal": {"AWS": [ "arn:aws:iam::862465512403:user/christophe" ]},
   "Action": [ "s3:PutObject" ],
   "Resource": "arn:aws:s3:::examplebucket/*"
  }
 ]
}
```

Amazon S3 – Bucket Policy Example

```
{
  "Id": "Policy1462372129776",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1462372128560",
      "Action": [
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
        "StringEquals": {
          "s3:x-amz-acl": "public-read"
        }
      },
      "Principal": "*"
    }
  ]
}
```

Amazon S3 – ACLs

- Used for both buckets and objects
- Grant read/write permissions to **other AWS accounts**
- You cannot grant conditional permissions
- You cannot explicitly deny permissions
- An object ACL is the only way to manage access to objects not owned by the bucket owner

Amazon S3 – IAM policies

- Existing objects are unchanged
- Added objects are given unique version IDs
  - Automatically generated by S3

Amazon Web Services

Certified Developer – Associate Level

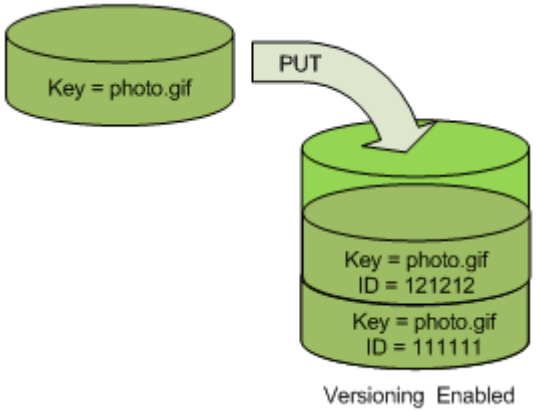
Object Versioning

Amazon S3 – Versioning

- Allows for multiple versions of an object
- Protects against unintended overwrites and deletions
- Automatically archives objects
- Versioning is at the bucket level
- Configured via the Console or SDKs

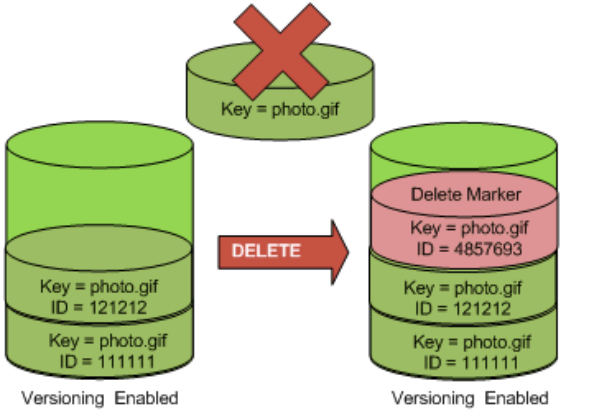
Enabling Versioning

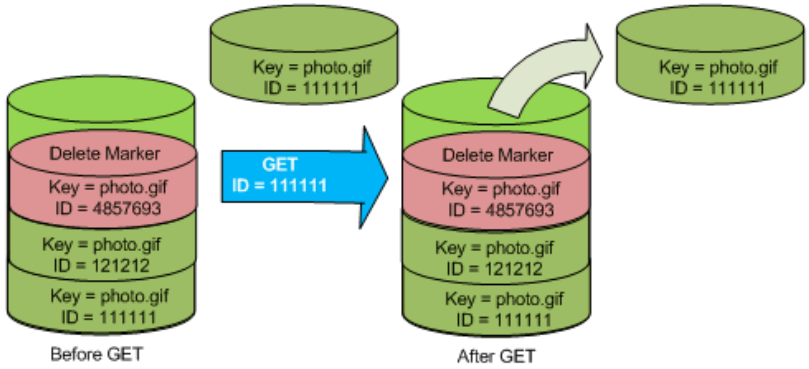
- Existing objects are unchanged
- Added objects are given unique version IDs
  - Automatically generated by S3



Deleting versioned objects

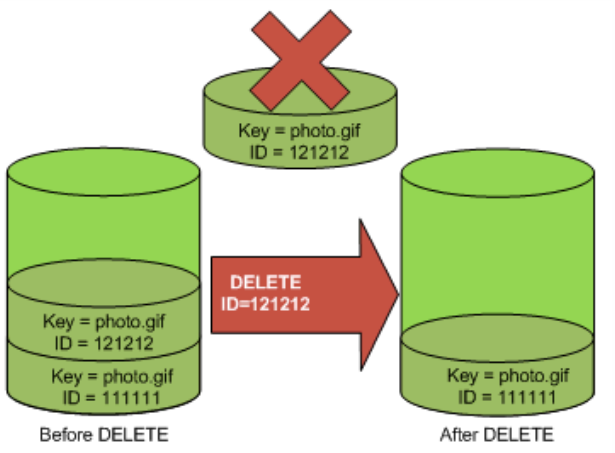
- All versions remain in the bucket but S3 inserts a delete marker
- The delete marker becomes the current version
- By default, **GET** requests retrieve the latest version
  - If the current version has a delete marker, it returns a **404 Not Found** error
  - You can get previous versions, however, by specifying an ID





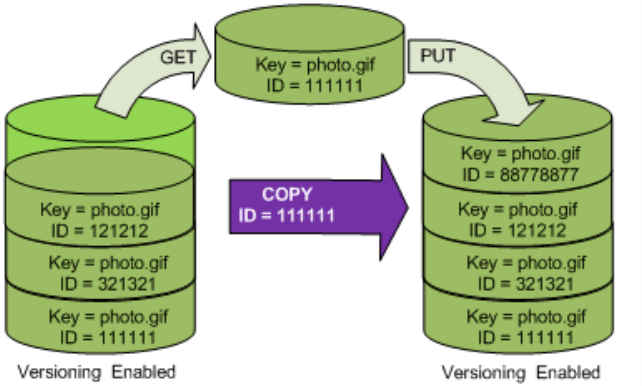
Permanently Deleting

- To permanently delete a version, specify an ID



Restoring versioned objects

- Any earlier version can be restored:
  - Copying a previous version into the same bucket will restore it as the current version
  - Permanently delete the current version (by specifying its ID)
- Copying an earlier version **GET**s the version and **PUT**s it in the bucket, giving it a new ID
  - That new ID is the used as the current version



Using Lifecycle Management with Versioning

- Lifecycle Management can dictate what happens to versions after a certain amount of time
  - Example: Send noncurrent versions to Amazon Glacier after 30 days
  - Example: Permanently delete objects 180 days after they become noncurrent
- Can have separate policies for current and noncurrent versions

Disabling Versioning

- Once version-enabled, a bucket *cannot* go back to being in an unversioned state
- You *can* suspend versioning
  - Future objects are given a version ID of null
  - Already versioned objects do not change
- To permanently delete a version, specify an ID

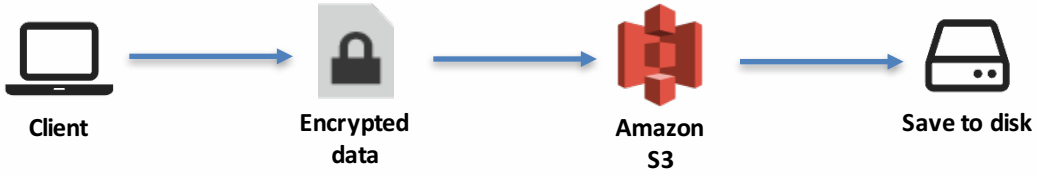
Amazon Web Services

Certified Developer – Associate Level

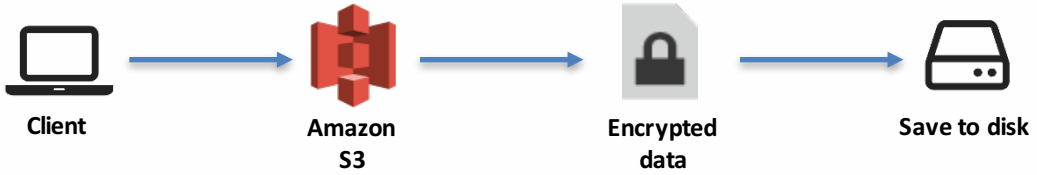
S3 Encryption

Amazon S3 – Protecting data using encryption

- Protecting data in-transit
  - Using SSL or client-side encryption



- Protecting data at rest
  - Request Amazon S3 to encrypt objects



Protecting data in-transit

- Using an AWS KMS-managed customer master key (CMK)
  - Client gets a unique encryption key for each object

- On upload:
  - Client first sends a request to AWS KMS for a key
  - AWS KMS returns an encryption key (plain text used to encrypt object data, and a cipher blob to upload to S3 as object metadata)
- On download:
  - Client downloads the encrypted object form S3 with the cipher blob stored in metadata
  - The client then sends that cipher blob to AWS KMS to get the plain text
  - Plain text is used to decrypt the object

Protecting data in-transit

- Using a client-side master key
  - Master keys and unencrypted data are never sent to AWS

- On upload:
  - Client provides a master key to the Amazon S3 encryption client
  - The S3 client generates a random data key and encrypts it with your master key
  - The S3 client encrypts your data using the data key, and uploads a material description as part of the object metadata (*x-amz-meta-x-amz-key*)
- On download:
  - Client downloads encrypted object along with its metadata
  - That metadata tells the client which master key to use to decrypt
  - Using that master key, the client decrypts the data key
  - The data key is used to decrypt the object

Protecting data at rest – Amazon S3-Managed Encryption

- Amazon S3 provides server side encryption before saving data to disk
- Add the x-amz-server-side-encryption request header to your upload request
- Uses AES-256 (Advanced Encryption Standard)
- Bucket policies can require all objects to use server-side encryption

```
{
   "Version":"2012-10-17",
   "Id":"PutObjPolicy",
   "Statement":[{
         "Sid":"DenyUnEncryptedObjectUploads",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::YourBucket/*",
         "Condition":{
            "StringNotEquals":{
               "s3:x-amz-server-side-encryption":"AES256"
            }
         }
      }
   ]
}
```

Protecting data at rest - Alternatives

- KMS-Managed Encryption Keys
  - Uses customer master keys (CMKs)
  - Gives you more flexibility in controlling keys

- Customer-Provided Encryption Keys
  - Gives you the option to generate your own keys outside of the AWS environment
  - Amazon does not store your encryption key

```php
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
// $filepath should be absolute path to a file on disk
$filepath = '*** Your File Path ***';

// Instantiate the client.
$s3 = S3Client::factory();

// Upload a file with server-side encryption.
$result = $s3->putObject(array(
    'Bucket'               => $bucket,
    'Key'                  => $keyname,
    'SourceFile'           => $filepath,
    'ServerSideEncryption' => 'AES256',
));
```

Amazon Web Services

Certified Developer – Associate Level

DynamoDB Essentials

DynamoDB Essentials

- Fully managed NoSQL database
  - Can scale up and down depending on demand without downtime or performance degradation
  - Manage data, not hardware or software
  - Built-in monitoring

- Consistent and fast performance
  - Data is stored on fast SSDs
  - You control performance through read/write capacity
  - Can spread out load across servers and tables
  - Replication across multiple availability zones in an AWS region (high availability)

DynamoDB Features

- DynamoDB can be used via the AWS Console or API
  - Multi-language support through SDKs (JavaScript, PHP, Python, mobile, etc…)
  - Build-in features that speed up development
  - Command line interface
- Flexible data model with attributes and items
- Supports different levels of consistency
  - Eventually consistent
  - Strongly consistent
- Conditional updates and concurrency control
  - Atomic counter

API Credentials with Access Keys

- Used to sign requests
  - The AWS SDKs use access keys
  - The Command Line Interfaces (CLIs)
- Can be disabled and deleted, but not retrieved.
- You can have temporary access keys which expire
- Useful when connecting from outside of AWS (like your computer or application)

Amazon Web Services

Certified Developer – Associate Level

DynamoDB Overview and Limits

DynamoDB Essentials

- Fully managed NoSQL database
  - Can scale up and down depending on demand without downtime or performance degradation
  - Manage data, not hardware or software
  - Built-in monitoring

- Consistent and fast performance
  - Data is stored on fast SSDs
  - You control performance through read/write capacity
  - Can spread out load across servers and tables
  - Replication across multiple availability zones in an AWS region (high availability)

DynamoDB Features

- DynamoDB can be used via the AWS Console or API
  - Multi-language support through SDKs (JavaScript, PHP, Python, mobile, etc…)
  - Built-in features that speed up development
  - Command line interface
- Flexible data model with attributes and items
- Supports different levels of consistency
  - Eventually consistent
  - Strongly consistent
- Conditional updates and concurrency control
  - Atomic counter

DynamoDB Features

- Pay only for what you use (cost effective)
- Integrates monitoring
- Integrates with AWS big data services such as Elastic MapReduce and Redshift
- Integrates with other services through streams

DynamoDB: Provisioned Throughput

- Flexible read and write performance capacity
  - Set during table creation
  - Can be changed at anytime without downtime or performance degradation
- Automatically allocated machine resources
- Ability to reserve capacity

DynamoDB: Important terms

Primary Key - Unique identifier

- Partition Key (also known as hash attribute)
  - Simple primary key composed of one attribute
  - Used to retrieve data
  - Must be unique
- Partition and Sort (also known as range attribute) key
  - Composite primary key composed of two attributes: the partition key and sort key
  - Can have the same partition key, but must have a different sort key

DynamoDB: Important terms

Secondary Indexes

- Indexes let you query data using alternate keys (more flexibility)
- Can provide better performance
- Global Secondary Index
  - Partition key and Sort key can both be different from those on the table
  - Has its own provisioned throughput
- Local Secondary Index
  - Partition key must be the same but the sort key is different
  - “Local” because every partition is scoped to a table partition with the same partition key value
  - Uses table’s provisioned throughput
- Up to 5 global and local indexes per table

DynamoDB: Important terms

Projected Attributes - Attributes copied from the table to the index, in addition to the primary key attributes and index key attributes

ProjectionType

- KEYS_ONLY - Only the index and primary keys are projected (smallest index - more performant)
- INCLUDE - Only specified attributes are projected
- ALL - All attributes are projected (biggest index - least performant)

DynamoDB: Important Limits

- 256 tables per region (can be increased on request)
- Partition key length: 2048 bytes maximum, 1 byte minimum
- Sort key length: 1024 bytes maximum, 1 byte minimum
- Item size: 400KB including attribute name and value

API-specific limits

- Up to 10 **CreateTable**, **UpdateTable**, and **DeleteTable** actions running simultaneously
- A single **BatchGetItem** operation can get a maximum of 100 items
  - Must be < 16 MB in size
- A single **BatchWriteItem** operation can contain up to 25 **PutItem** or **DeleteItem** requests
  - Total size of all items must be < 16 MB
- **Query** and **Scan** result set is limited to 1 MB of data per call
  - **LastEvaluatedKey** in the response can be used to retrieve more

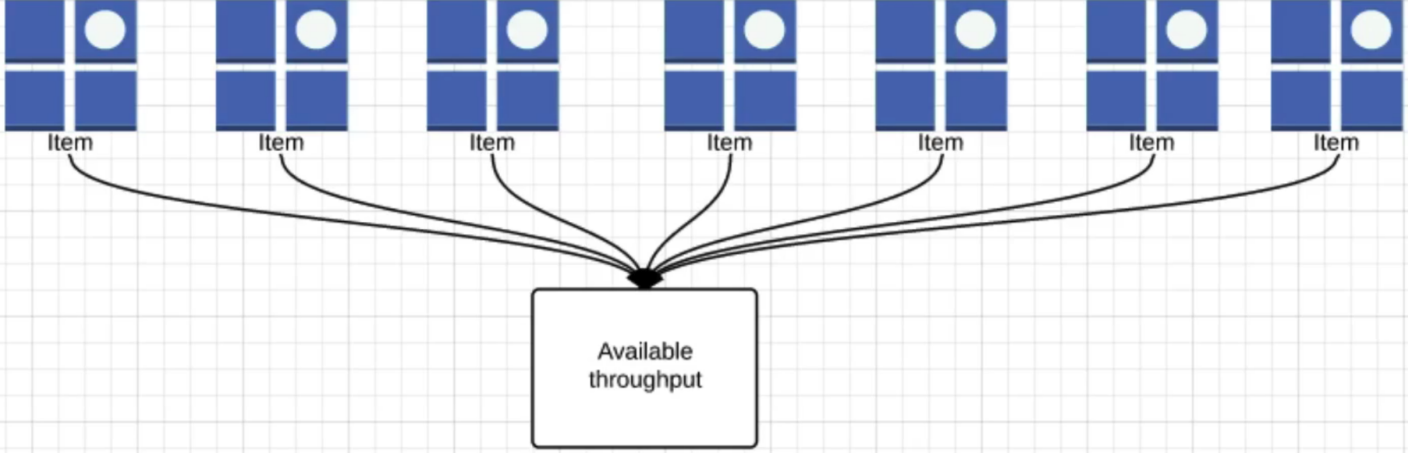
Amazon Web Services

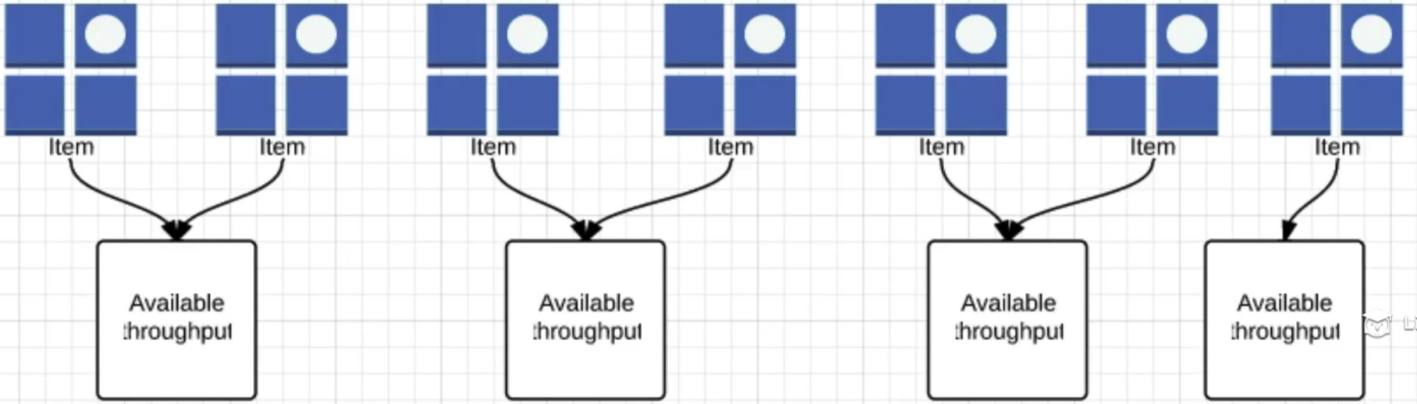
Certified Developer – Associate Level

Throughput Provisioning

DynamoDB Provisioned Capacity

- Unit of read capacity: 1 strongly consistent read per second or two eventually consistent reads per second for **items as large as 4 KB**
- Unit of write capacity: 1 write per second for **items up to 1KB**
- Key concepts needed:
  - Calculating required throughput
  - Understanding how secondary indexes affect throughput
  - Understanding what happens if your application's read/writes exceed throughput





Calculating Read Capacity

- Round up to the nearest 4 KB multiplier
- Items that are 3 KB in size can still only do 1 strongly consistent or 2 eventually consistent reads per second
- Example:
  - Item size 3 KB
  - Want to read 80 items per second from the table
  - How many read capacity units are required?

Calculating Read Capacity - Example

- Example: Your items are 3KB in size and you want to read 80 (strongly consistent read) items from a table per second
  - Item size 3KB
  - Want to read 80 items per second from the table
- Formula: (**ITEM SIZE** (rounded up to the next 4KB multiplier) / **4KB**) * # of items
- 80 * (3KB (round up to 4) / 4KB)
  - 80 * 1 = 80 required provisioned read throughput
- Bonus: Eventually consistent reads would cut that in half so:
  - (80 * 1) / 2 = 40 required read capacity

Calculating Read Capacity - Example #2

- Example: Your items are 10KB in size and you want to read 80 (strongly consistent read) items from a table per second
  - Item size 10KB
  - Want to read 80 items per second from the table
- Formula: (**ITEM SIZE** (rounded up to the next 4KB multiplier) / **4KB**) * # of items
- 80 * (10KB (round up to 12) / 4KB)
  - 80 * 3 = 240 required provisioned read throughput
- Bonus: Eventually consistent reads would cut that in half so:
  - (80 * 3) / 2 = 120 required read capacity

Calculating Write Capacity

- Round up to the nearest 1 KB multiplier
- Example:
  - Item size 1.5 KB
  - Want to write 10 items per second from the table
  - How many write capacity units are required?

Calculating Write Capacity - Example

- Example: Your items are 1.5KB in size and you want to write 10 items per second

- Formula: (**ITEM SIZE** (rounded up to the next 1KB multiplier) / 1**KB**) * # of items
- 10 * (1.5KB (round up to 2) / 1KB)
  - 10 * 2 = 20 required provisioned write throughput

Read Throughput with Local Secondary Indexes

- Uses the same read/write capacity from parent table
- If you read only index keys and projected attributes, the calculations are the same
  - You calculate using the size of the index entry, not the table item size
  - Rounded up to the nearest 4KB
- If queried attributes aren't projected attributes or keys, we get extra latency and read capacity cost
  - You use read capacity from the index AND for every item from the table. Not just the attribute needed

Write Throughput with Local Secondary Indexes

- Adding, updating, or deleting an item in a table also costs write capacity units to perform the action on the local index
- The cost of writing an item to a local secondary index depends on a few things:
  - If you write a new item to the table and that item defines an indexed attribute, or if you update an existing item and write an indexed attribute that was previously undefined, that will cost you **one write operation to put the item** in the index.
  - If you change the value of an indexed key attribute, **two writes are required**. One to delete the previous item from the index, and another to put the new item into the index.
  - If an update deletes an item that was in the index, **one write is required to delete the item** from the index.

Read Throughput with Global Secondary Indexes

- Global indexes have their own throughput capacity, completely separate from that of the table's capacity.
- Global indexes support eventually consistent reads, which means that a single global secondary index query can get up to 8 KB per read capacity unit (because we take 4KB and multiply it by 2)
- Reads in global indexes are calculated the same as in tables, except that the size of the index entries is used instead of the size of the entire item.

Write Throughput with Global Secondary Indexes

- Putting, Updating, or Deleting items in a table consumes the **index write capacity units**
- The cost of writing an item to a global index depends on a few things, and those are identical to the local secondary index rules

Exceeding Throughput

- Requests exceeding the allocated throughput may be throttled
- With global secondary indexes, all indexes must have enough write capacity or the write might get throttled (even if the write doesn't affect the index!)
- You can monitor throughput in the AWS Console

Amazon Web Services

Certified Developer – Associate Level

Scans and Queries

DynamoDB Queries

- Allow you to find items using only primary key values from a table or secondary index

- Benefits:
  - Returns the items matching the primary key search
  - Much more efficient because it searches indexes only
  - Returns all attributes of an item, or only the ones you want
  - Is eventually consistent by default but can request a consistent read
  - Can use conditional operators and filters to return precise results

DynamoDB Scans

- Reads every item in the table and is operationally inefficient
- Looks for all items and attributes in a table by default

- Benefits:
  - Scans can apply filters to the results to refine values
  - Can return only specific attributes with the **ProjectionExpression** parameter

- Negatives:
  - The larger the data set in the table the slower performance of a scan
  - The more filters on the scan the slower the performance
  - Returns only filtered results
  - Only eventually consistent reads are available

DynamoDB Scans - If you must use them

- What you should keep in mind:
  - You can reduce **Page Size** of an operation with the **Limit** parameter, to limit how much data you try to retrieve at the same time
  - Avoid performing Scans on mission-critical tables
  - Program your application logic to retry any requests that receives a response code saying that you exceeded provisioned throughput (or increase throughput)

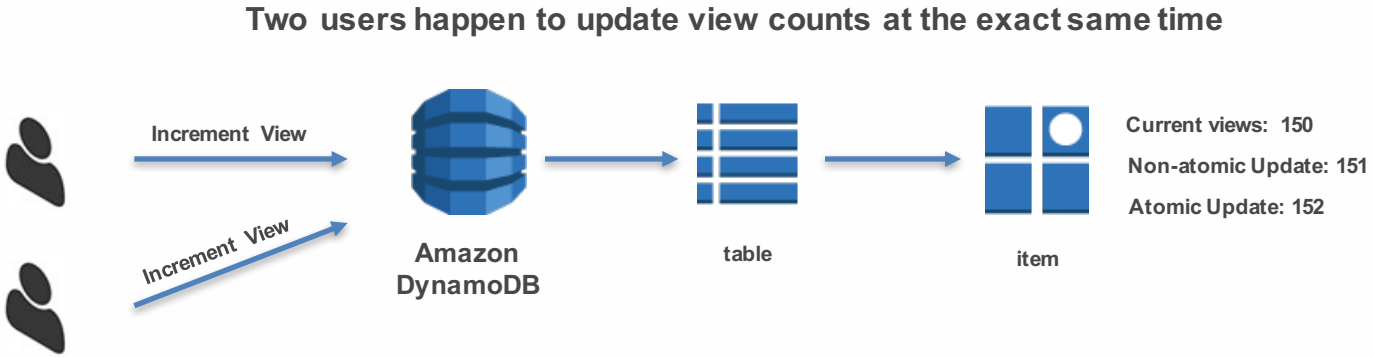
Amazon Web Services

Certified Developer – Associate Level

Atomic Counters and Conditional Writes

DynamoDB - Atomic Counters

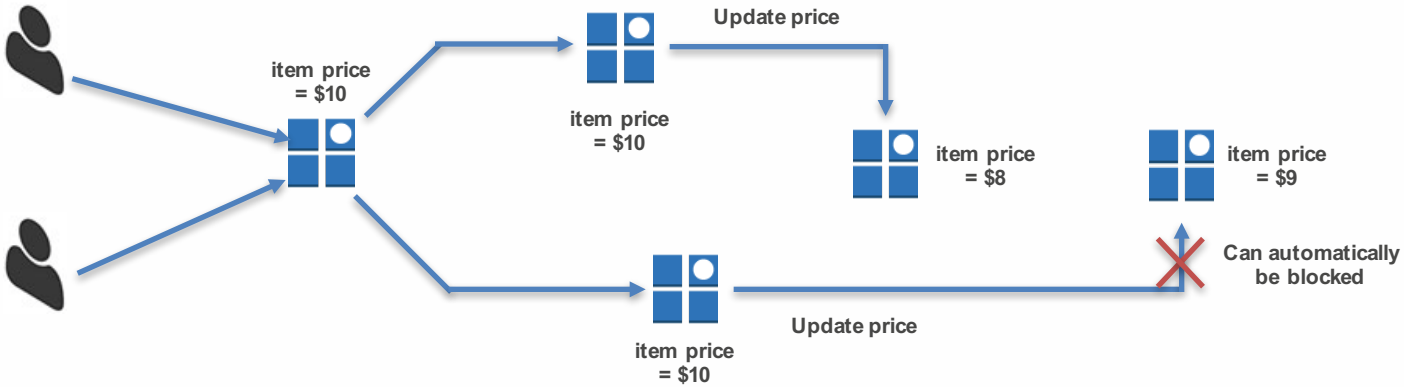
- Allow you to increment or decrement the value of an attribute without interfering with other write requests
- Requests are applied in the order that they are received
- Updates are not idempotent - it will update the value each time it is called
- Use case: Increasing view count



DynamoDB - Conditional Writes

- Helps coordinate writes
- Checks for a condition before proceeding with the operation
- Supported for **PutItem**, **DeleteItem**, **UpdateItem** operations

- Specify conditions in **ConditionExpression**:
  - Can contain attribute names, conditional operators, and built-in functions
- A failed conditional write returns **ConditionalCheckFailedException**



Amazon Web Services

Certified Developer – Associate Level

Temporary Credentials

DynamoDB - Granting temporary access

- You can use temporary security credentials to make calls to AWS services
- Example: Users on your mobile application need the ability to communicate with DynamoDB. How can you create permissions for each user to access DynamoDB resources needed for your mobile application to work?
- Options:
  - IAM roles
  - Web Identity Federation
  - Amazon Cognito

Identity Federation

- You can grant users access when they sign in to external systems outside of AWS
- IAM supports two types of identity federation:
  - **Enterprise identity federation** - authenticate users in your organization's network
  - **Web identity federation** - users can sign in with Amazon, Facebook, Google, or any OpenID Connect (OIDC) 2.0 compatible provider

Source: Amazon Web Services Documentation

Amazon Cognito

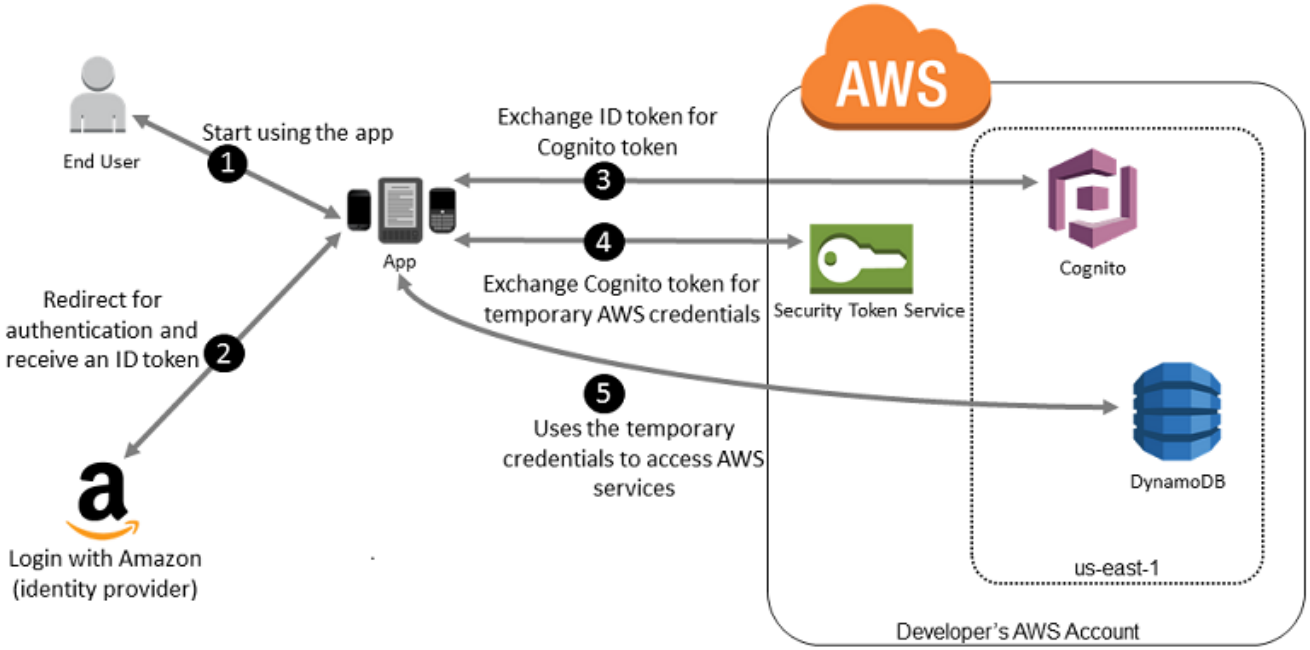
- Amazon Cognito Identity:
  - Create unique identities for users
  - Authenticate identities with identity providers, or your own auth process
  - Supports unauthenticated identities
  - Save mobile user data
  - Use credentials obtained to synchronize data with Cognito Sync
- Amazon Cognito Sync:
  - Sync user data across mobile devices and the web
  - Client libraries cache data locally

IAM Roles

- We need roles for our users to assume
- Scenario:
  - User logs in to app
  - We generate temporary AWS credentials
  - The temporary credentials are associated with a specific IAM role
  - The IAM role gives access to DynamoDB to read/write progress

Temporary Credentials in your app

- SDKs
  - Android
  - iOS
  - JavaScript
  - etc…
- AWS Mobile Hub



Source: Amazon Web Services Documentation

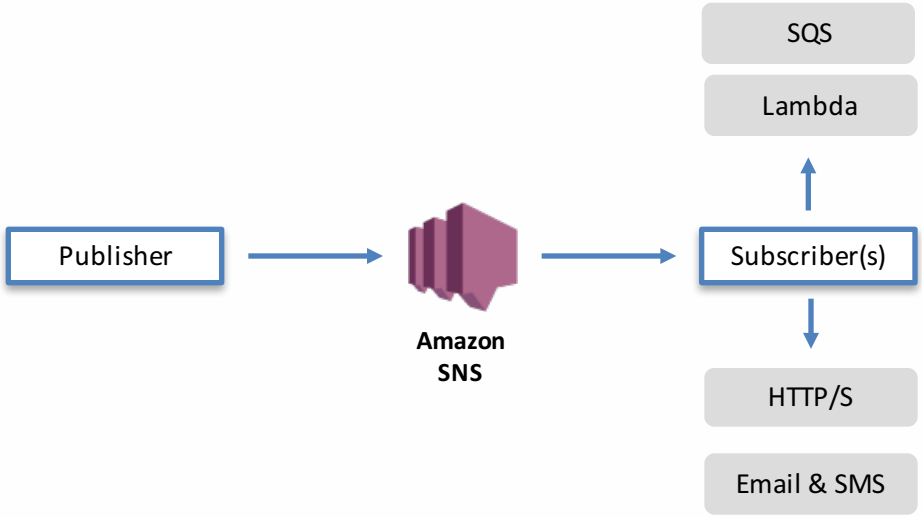
Amazon Web Services

Certified Developer – Associate Level

Introduction to Amazon SNS

Amazon SNS – What is it?

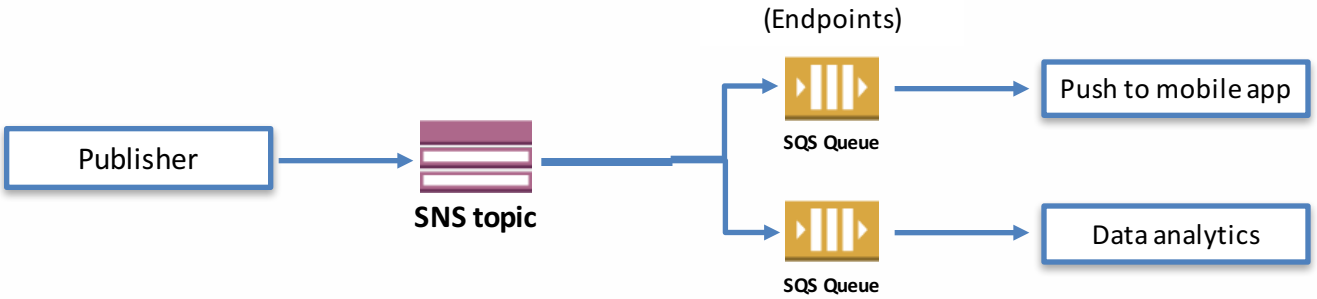
- **Pub-sub** service for messaging
- **Publishes** messages to subscribers via **topics**
- **Subscribers** receive the message when subscribed to those topics
- Scalable and highly reliable
- Supported through the Console, API, CLI, and SDKs



Amazon SNS - Examples

- The **fanout** scenario:
  - Message is sent to a topic and pushed to multiple endpoints
  - Allows for parallel asynchronous processing

- Example: A user completes the last puzzle of your mobile game. As a reward, you want to send that user a special offer. You also want to send the user's records to another service for data analysis. This data analysis is used to figure out how the user solved this very complicated puzzle.



Amazon SNS - Examples

- Alerts
  - Notifications triggered by thresholds
  - Sends immediate notifications
  - Works with other AWS services
- Example: You monitor your infrastructure with Amazon's CloudWatch. You create a CloudWatch alarm that is linked to an Amazon SNS topic. Subscribers to that topic will receive alerts as soon as there is an issue. Subscribers could send email/SMS messages.

Amazon SNS - Publishers

- The owner creates a topic and controls access to it
  - Policies determine which topic(s) publishers can write to
- You can publish from:
  - The command line interface (CLI)
  - Your applications (HTTP)
  - SDKs
  - AWS services

Amazon SNS - Topics

- Channel to send messages and subscribe to notifications
- Names must be unique
- Names are limited to 256 characters
- Alphanumeric (letters and numbers) characters plus hyphens (-) and underscores (_) are allowed
- Topics and messages are stored redundantly on multiple servers and data centers

Amazon SNS - Subscribers

- Subscribe to a topic to receive published messages
- Subscribers are endpoints:
  - Mobile apps
  - Web servers
  - Email addresses
  - Amazon SQS queue
  - AWS Lambda

Amazon Web Services

Certified Developer – Associate Level

Access Policies

Amazon SNS – Message data

- JSON formatted key-value pairs
- Allows developers to grab the message data and parse it
- POSTs to HTTP/S endpoints with specific headers
- Allows developers to verify the type and authenticity of the message

Amazon SNS – Message data

- Message – Message value specified when the notification was published to the topic
- MessageId – Universally Unique Identifier (UUID). Same id must be used for retries
- Signature – Base64-encoded "SHA1withRSA" signature of the Message, MessageId, Subject, Type, Timestamp, and TopicArn values
- SignatureVersion – Version of the Amazon SNS signature used
- SigningCertURL – The URL to the certificate that was used to sign the message
- Subject – Subject parameter. Optional parameter.
- Timestamp – The time (GMT) when the notification was published
- TopicArn – ARN for the topic that this message was published to
- Type – Type of the message. (ie: Notifications are type *Notification*)
- UnsubscribeURL – URL that you can use to unsubscribe

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: da42e39f-ee1d-467b-t012-a6edd3115ane
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-west-2:123456789012:MyTopic:2bcfbf39-05c3-41de
  -beaa-fcfcc21c8f55
Content-Length: 761
Content-Type: text/plain; charset=UTF-8
Host: ec2-50-17-44-49.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "da41e39f-ea4d-435a-b922-c6aae3915ebe",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "test",
  "Message" : "test message",
  "Timestamp" : "2012-04-25T21:49:25.719Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfaMpN2nk5HVSw7XnOn/49IkxDKz8Y
  rlH2qJXj2iZB0Zo2O71c4qQk1fMUDi3LGpij7RCW7AW9vYYsSqIKRnFS94ilu7NFhUzLiieYr4BKHpdTmdD6c0e
  sKEYBpabxDSc=",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
    f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.
    com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
    west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55"
}
```

Message Attributes

- Can send structured metadata along with messages
- Allows override default denies
- Explicit denies override allows
- Order of policies does not matter

Amazon Web Services

Certified Developer – Associate Level

Access Policies

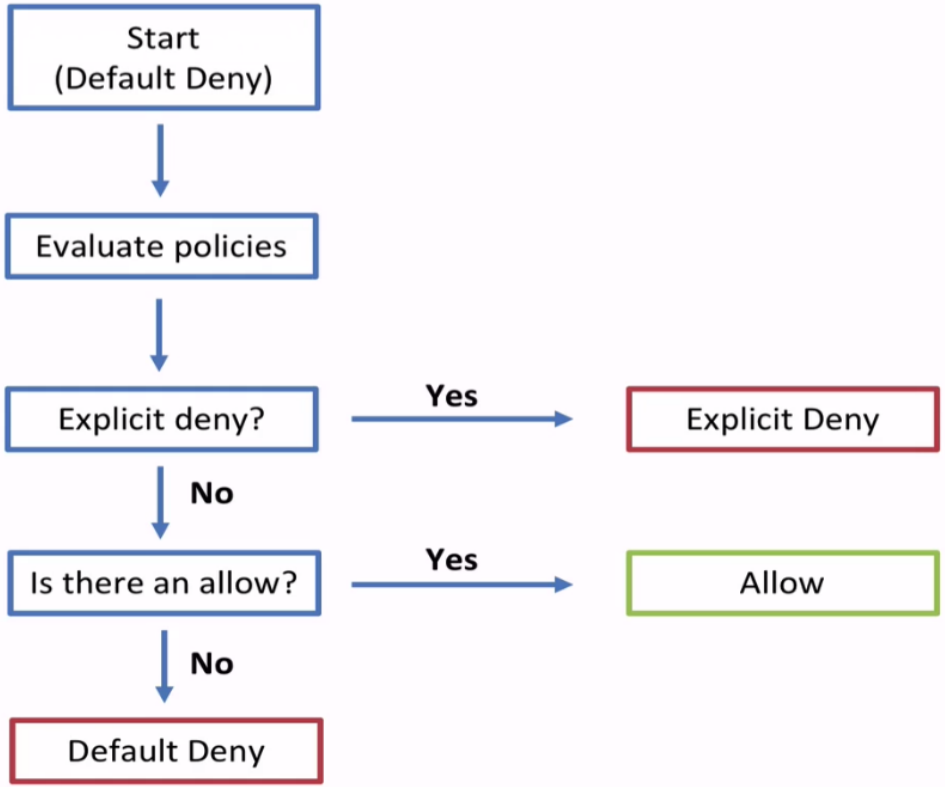
Amazon SNS – Managing Access

- Access controlled with policies
- Amazon SNS has its own permissions system based off of IAM
- Also integrates with IAM for control over your user's permissions
- You can control:
  - Who is allowed to publish to a topic
  - Who is allowed to subscribe to a topic
  - Under what conditions

Access Control Policies

- The AWS account owner has the only permissions by default
- Allows override default denies
- Explicit denies override allows
- Order of policies does not matter

Access Policy Evaluation - Steps



Access Policy Example Use Case

- Grant access to your SNS topic to another AWS account
- Use the Amazon SNS API action **AddPermission**
- Pass in the:
  - Topic
  - AWS account ID(s)
  - Actions
  - Label
- Amazon automatically generates an access control policy

IAM policies

- Control access for our own users
- IAM policies can't grant access to other AWS accounts
- Can be general or more specific and can include conditions
- We can use both IAM policies and Access Control policies at the same time
- IAM policies can grant temporary security crendentials

Amazon Web Services

Certified Developer – Associate Level

Mobile Push Notifications

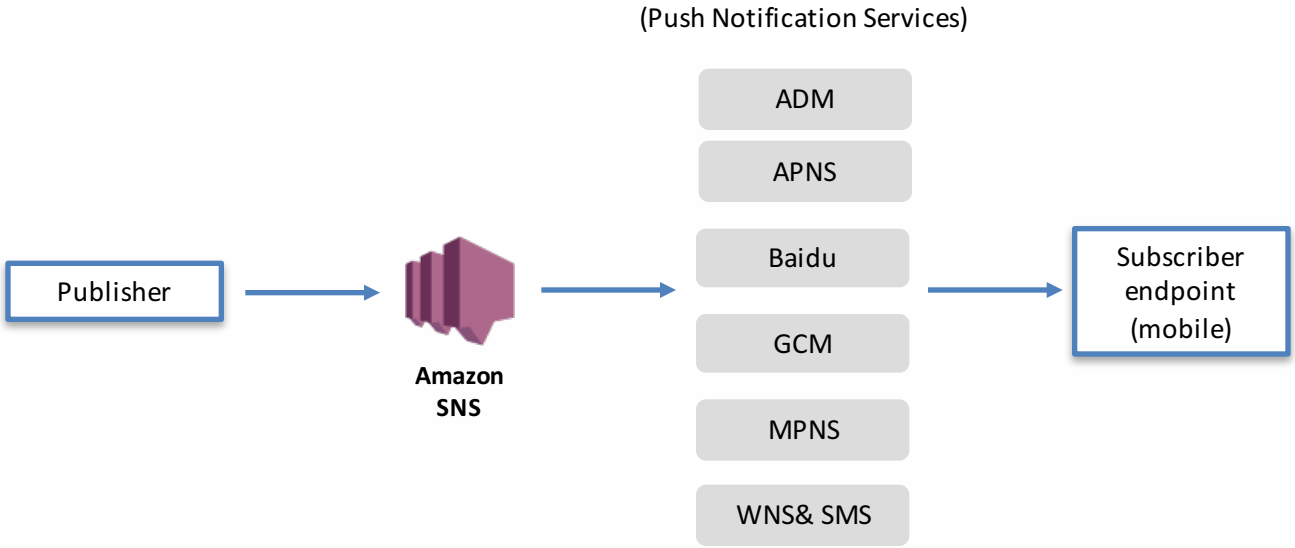
Amazon SNS – Mobile Push Notifications

- SNS provides the ability to send notifications directly to apps on mobile devices

- Notifications sent to a mobile end point have the ability to appear in the app as:
  - Message alerts
  - Badge updates
  - Sound alerts

Amazon SNS – Push Notification Services

- Mobile push notification services:
  - Amazon Device Messaging (ADM)
  - Apple Push Notification Service (APNS)
  - Baidu Cloud Push
  - Google Cloud Messaging for Android
  - Microsoft Push Notification Service for Windows Phone
  - Windows Push Notification Services


(Push Notification Services)
ADM
APNS
Baidu
GCM
MPNS
WNS& SMS
Publisher
Amazon SNS
Subscriber endpoint (mobile)

SNS Push Notifications Setup Process

- SNS needs a device token in order to send notifications to mobile endpoints
- There are device tokens and registration IDs, depending on the mobile platform

1. Request credentials from mobile platforms (ADM, APNS, etc…)
2. Request a token from mobile platforms
3. Create a platform application object
4. Create a platform endpoint object
5. Publish a message to the mobile endpoint

Platform Specific Payloads

- We can send push notifications at the same time to all apps/devices, no matter the mobile platform
- We can also send different messages depending on the platform

```
{
"default": "This is the default message which must be present when publishing a message
    to a topic. The default message will only be used if a message is not present for
    one of the notification platforms.",
"APNS": "{\"aps\":{\"alert\": \"Check out this awesome message!\",\"url\":\"www.
    linuxacademy.com\"} }",
"GCM":"{\"data\":{\"message\":\"Check out this awesome message!\",\"url\":\"www.
    linuxacademy.com\"}}",
"ADM": "{ \"data\": { \"message\": \"Check out this awesome message!\",\"url\":\"www.
    linuxacademy.com\" }}"
}
```

Amazon Web Services

Certified Developer – Associate Level

Introduction to Amazon SQS

Amazon SQS – What is it?

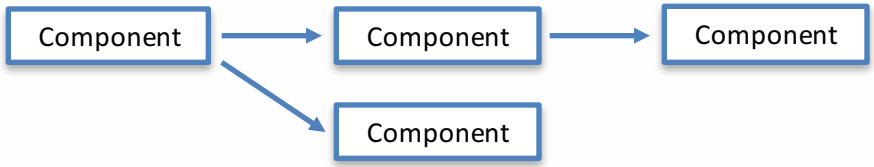
Amazon SQS is a scalable messaging queue service. Queues store messages that travel between different components of your application architecture. Messages in queues are generated by one component from your application and consumed by another.

Advantages from using a queue:

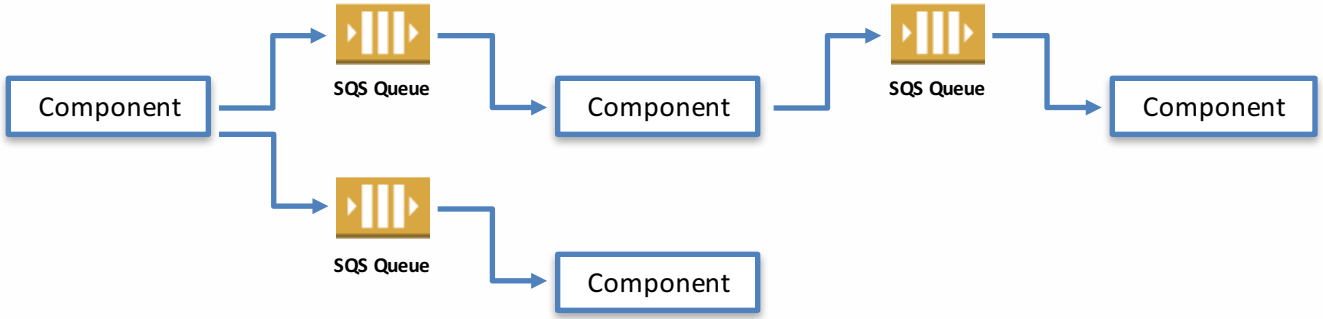
- Components are loosely coupled for high availability, reliability, and scalability
- Protection against losing data on application failure

Amazon SQS - Example

Tight Coupling



Loose Coupling



Amazon SQS – Core Features

- Redundant Infrastructure – Guarantees delivery of your messages **at least once**
- Multiple writers and readers – Multiple components can send/receive messages at the same time
- Queues can be configured – You can have different settings for queues depending on your needs
- Access Control – You control who sends and receives messages
- Delay Queues – Control how much time goes by before a message is available
- PCI Compliant – Compliant with Payment Card Industry (PCI) Data Security Standard (DSS), which means we can support the processing, storage, and transmission of credit card data by a merchant or service provider

Amazon SQS – Message Lifecycle

1. Component 1 sends Message A to a queue, and then the message is redundantly distributed across SQS servers

2. When component 2 is ready, it retrieves the message from SQS. While Message A is being processed, it remains in the queue but is not returned to any subsequent request for the duration of the *visibility timeout*

3. Component 2 deletes the message from the queue during that visibility timeout, or else it will get processed again.

Amazon SQS – Visibility Timeout

- Messages aren't deleted by SQS once they are received and processed, instead they must be deleted by the component
- Visibility timeout is used to block other components from processing a message
- You can choose what the timeout is, and you can extend it for individual messages if necessary
- This can be controlled via the SQS API

Amazon SNS – Long Polling vs. Short Polling

Short polling returns a result immediately, even if the queue is empty. It also only checks a subset of servers, which could cause false empty responses.

Long polling doesn't return a response until there is a message in the queue. It also checks every server to avoid false empty responses.

Long polling should be used when possible, because it can reduce costs and false empty responses.

Amazon SQS – Terms and limitations

- Message size – Messages can contain up to 256kb of text in any format
- Up to 120,000 inflight messages
- Message Retention Period – Amount of time a message will live in a queue if it is not deleted
- Receive Message Wait Time – If set to a value greater than 0, long polling is enabled. This is the maximum amount of time that a long polling call will wait for a message to become available before returning empty.
- Dead letter queues – Queues that other queues can send messages to, when those messages could not be successfully processed. You can then analyze those messages.

Amazon SQS – Terms and limitations

- Delay Queues
  - Min: 0 seconds
  - Max: 15 minutes
- Message Retention Period
  - Min: 1 minute
  - Max: 14 days
- Visibility Timeout
  - Min: 0 seconds
  - Max: 12 hours
- Receive Message Wait Time
  - Min: 0 seconds
  - Max: 20 seconds

Amazon Web Services

Certified Developer – Associate Level

CloudFormation Essentials

Amazon CloudFormation

- CloudFormation allows you to create and provision resources in a reusable template fashion
- CloudFormation turns your resources into stacks that work as units
- CloudFormation allows you to source control your infrastructure
- Templates are JSON compatible

Anatomy of a CloudFormation Template

- Templates can have 8 main sections
  - AWSTemplateFormatVersion
  - Description
  - Metadata
  - Parameters
  - Mappings
  - Conditions
  - Resources
  - Outputs

Version and Description

- AWSTemplateFormatVersion
  - Specifies which template version you want to use

- Description
  - This section follow the template version section
  - Descriptions help clearly differentiate between templates

Template Metadata

- Metadata
  - JSON objects that provide details about the template

Template Parameters

- Parameters
  - Values you can pass in right before template creation
  - Allows you to customize templates
  - Can have default values as well as allowed values

Template Mappings

- Mappings
  - Lets you map keys to values
  - For example: You can make different values for different regions

Template Conditions

- Conditions
  - Conditions can check values before deciding what to do
  - Allows you to create different resources in the same template depending on the condition evaluation
  - Example: can create different environments for development and production

Template Resources

- Resources
  - This is where you create different resources, like S3, EC2 instances, etc…
  - The only required section

Template Outputs

- Outputs
  - Can output values that you'd like to see from the console or from API calls
  - Example: can return the bucket name that you just created in S3

Intrinsic Functions

- Used to pass in values that are not available until run time
- Fn::GetAtt example

```
"Resources" : {  "S3Bucket" : { … } },
"Outputs"  : {
        "WebsiteURL" : {
                "Value" : { "Fn::GetAtt"  : [ "S3Bucket", "WebsiteURL" ] },
                "Description"  : "URL for website hosted on S3”
        }
}
```

Intrinsic Functions

- Fn::FindInMap – Returns the value of a key from a specified mapping

```
"Mappings" : {
        "RegionMap" : {
                "us-east-1" : { "32" : "ami-6411e20d", "64" : "ami-7a11e213" },
                "us-west-1" : { "32" : "ami-c9c7978c", "64" : "ami-cfc7978a" },
                "eu-west-1" : { "32" : "ami-37c2f643", "64" : "ami-31c2f645" }
        }
},
"Resources" : {
        "myEC2Instance" : {
                "Type" : "AWS::EC2::Instance",
                "Properties" : {
                        "ImageId" : { "Fn::FindInMap" : [ "RegionMap", { "Ref" : "AWS::Region" }, "32"]},
                        "InstanceType" : "m1.small”
                }
        }
}
```

Other Intrinsic Functions

- Fn::Join – Concatenates elements, separated by a specified delimiter
- Ref – Return a resource or value based on a logical name or parameter
- Fn::GetAZs – Get the Availability Zones for a specified region
- Fn::Select – Returns a single object from a list of objects by index

CloudFormation Rollback

- If a stack fails to create a resource, by default the stack will rollback
- Rollback – Removal of all created resources after a failed stack creation, or after cancelling creation
- Rollback can be disabled from the API

CloudFormation Advanced Concepts

- CloudFormation templates allow you to declare cloud-init scripts for EC2 resources

- CloudFormation templates allow the use of regular expressions in certain declarations

Amazon Web Services

Certified Developer – Associate Level

Amazon SWF Concepts and Overview

Amazon Simple Workflow Service

A task coordination and state management service for cloud applications

Features:

- Distributed
- Can scale up or down depending on the task
- Works with on-premises applications or applications on the cloud
- A workflow can consist of human events
- Workflows can last up to 1 year
- Guarantees order of execution

Amazon Simple Workflow Service: Domains

- A domain is used to help determine the scope of workflows
- Multiple work flows can live inside of a domain
- Workflows cannot interact with work flows in other domains

Amazon Simple Workflow Service: Workers & Deciders

- Activity worker – Process that performs an activity that is part of the workflow
- Activity workers poll SWF for new tasks that they need to perform
- After receiving the task, the activity worker will process the task however it is instructed to do so, and then report back to SWF
- Workers can consist of a server (code being executed) on EC2 or on-premises

Amazon Simple Workflow Service: Tasks

Activity Task

- A task assigned to a worker

Example: Encoding videos or checking inventory

Decision Task

- Tells the decider that the state of the workflow execution has changed
- Allows the decider to determine what the next activity is
- Decision tasks happen whenever the state of the workflow changes (task completed)

Amazon Web Services

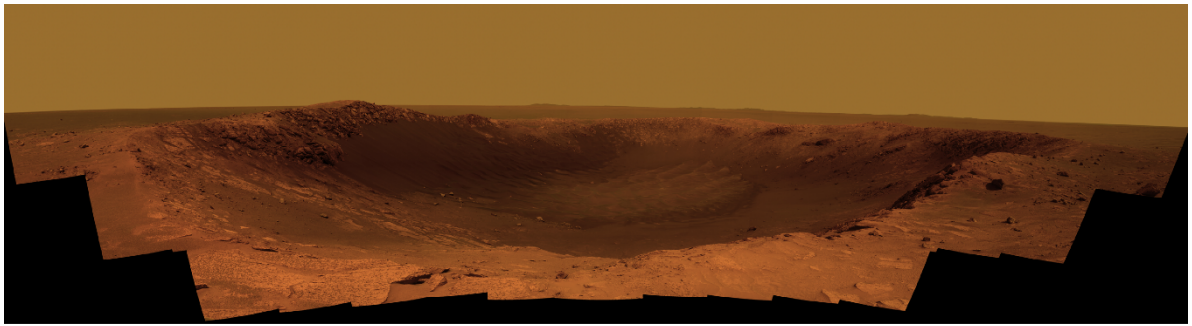
Certified Developer – Associate Level

SWF Examples

Amazon Simple Workflow Service

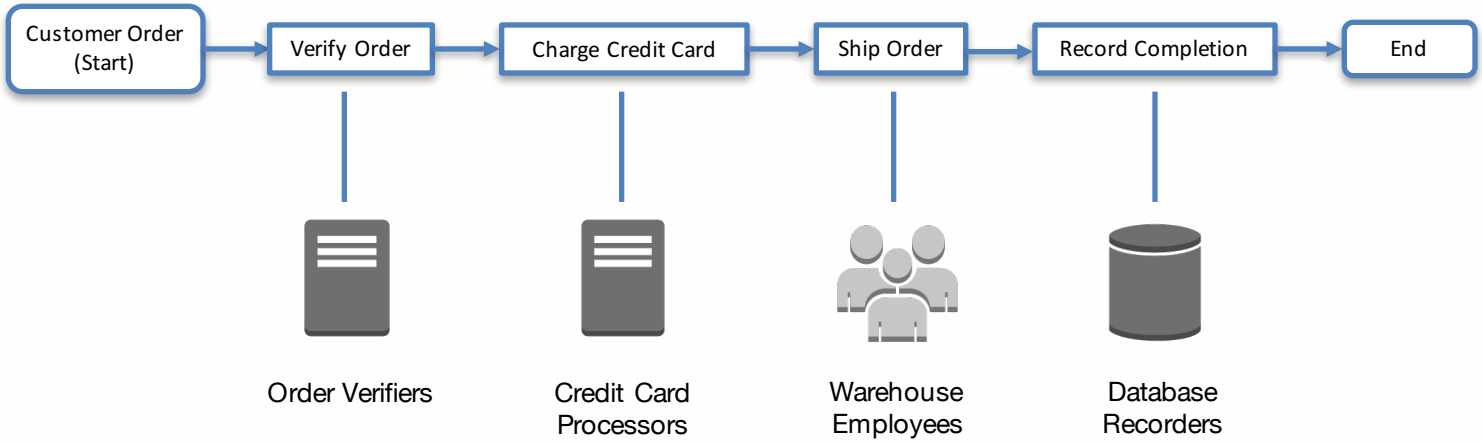
Examples:

- Transcoding Videos
- E-Commerce processing
- NASA



https://aws.amazon.com/swf/testimonials/swfnasa/

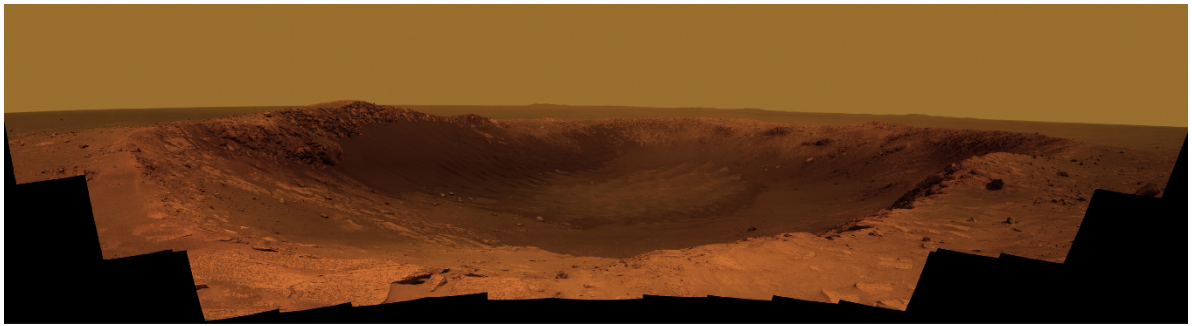
Amazon Simple Workflow Service: E-Commerce example



Amazon Simple Workflow Service

Examples:

- Transcoding Videos
- E-Commerce processing
- NASA



https://aws.amazon.com/swf/testimonials/swfnasa/

Amazon Simple Workflow Service: SQS vs. SWF

- Both are used to create distributed systems
- Both allow each task/component to scale independently
- SQS uses a “best effort” messaging order and could have duplicates
- SWF guarantees execution order and uses deciders for next instructions
- SWF can have a human task as part of the workflow
- SQS messages live up to 14 days
- SWF workflows or task executions can last up to 1 year
- SWF allows for synchronous and asynchronous distributed processing

Amazon Web Services

Certified Developer – Associate Level

VPC Essentials

Amazon VPC: What is a Virtual Private Cloud?

A VPC resembles:

- Private data centers
- Private corporate networks

Private Network

- Private and public subnets
- Scalable infrastructure
- Ability to extend corporate/home network to the cloud as if it were part of your network

Amazon VPC: Benefits of a VPC

- Ability to launch instances into a subnet
- Ability to define custom IP address ranges inside of each subnet (private and public)
- Ability to configure route tables between subnets
- Ability to configure internet gateways and attach them to subnets
- Ability to create a layered network of resources
- Extending our network with VPN/VPG controlled access
- Ability to use Security Groups and Subnet network ACLs

Understanding the default VPC

- Default VPC is a different setup than a non-default VPC
- Default VPC gives users easy access to a VPC without having to configure it from scratch
- Default VPC subnets have internet gateways attached
- Each instance added has a default private and public IP address
- If you delete the default VPC, the only way to get it back is to contact AWS

Understanding the non-default VPC

- Non-default VPCs have private IP addresses but not public IP addresses
- Can only access resources through elastic IP addresses, VPNs, or gateway instances
- Do not have internet gateways attached by default

VPC Peering

- VPC Peering allows you to setup direct network routing between different VPCs using private IP addresses
- Instances will communicate with each other as if they were on the same private network
- VPC Peering can occur between other AWS accounts and other VPCs within the same region

Scenarios:

- Peering two VPCs – Company runs multiple AWS accounts and you need to link all the resources as if they were all under one private network
- Peering TO a VPC – Multiple VPCs connect to a central VPC but cannot communicate with each other, only the central VPC (file sharing, customer access, Active Directory)

VPC Scenarios

- VPC with public subnet only – Single tier apps
- VPC with public and private subnets – Resources that don't need public internet access/layered apps
- VPC with public and private subnets and hardware connected VPN – Extending to on-premises
- VPC with a private subnet only and hardware VPN access

VPC Limits

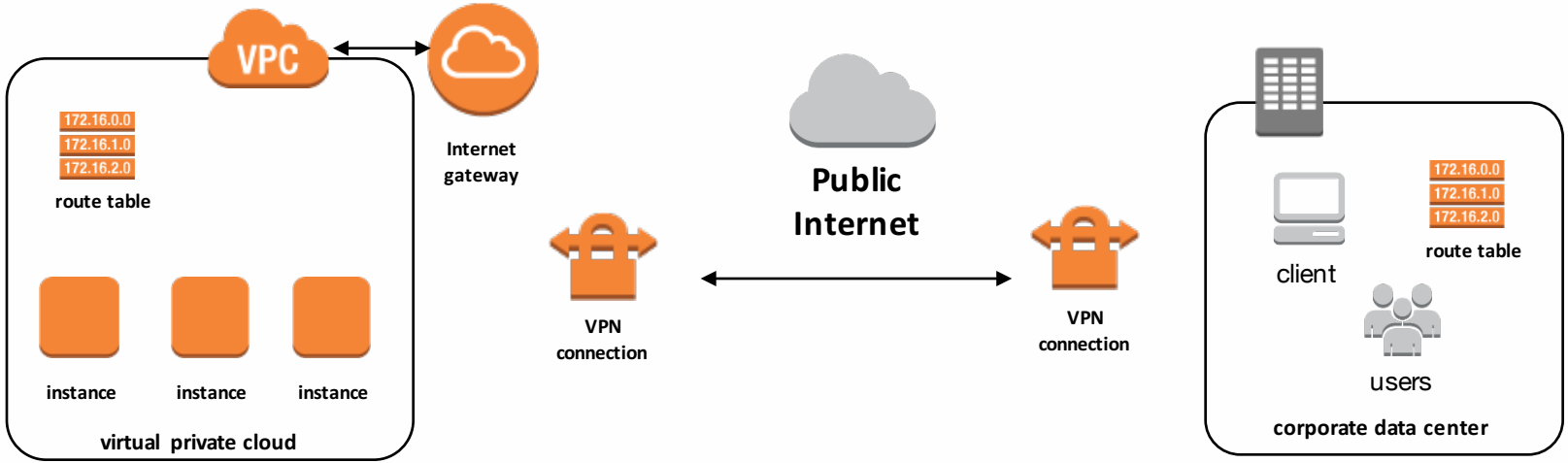
- 5 VPCs per region
- 200 Subnets per VPC
- 50 Customer gateways per region
- 5 Internet gateways per region
- 5 Elastic IP addresses per region for each AWS account
- 50 VPN connections per region
- 200 route tables per region
- 500 security groups per region

Amazon Web Services

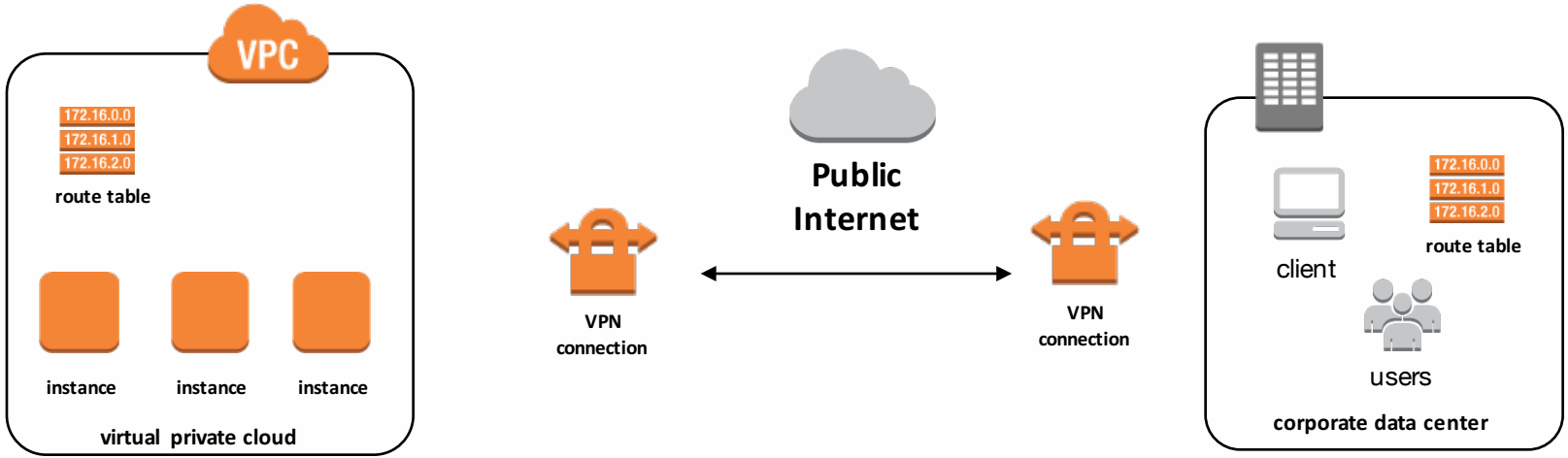
Certified Developer – Associate Level

VPC Networking Essentials

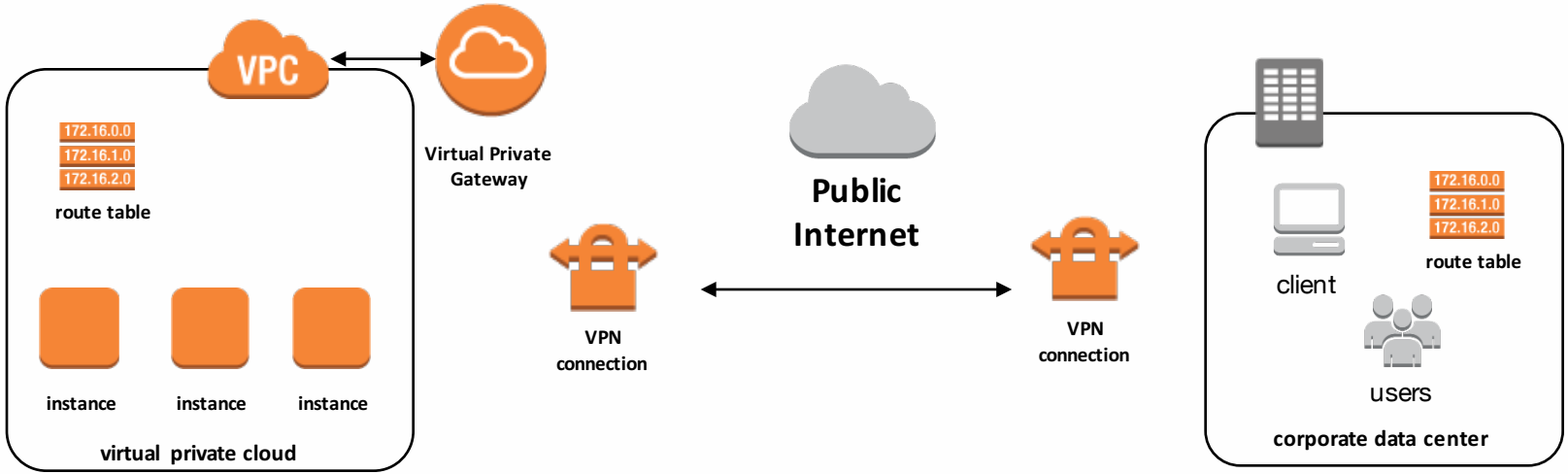
Amazon VPC: Networking



Amazon VPC: Networking



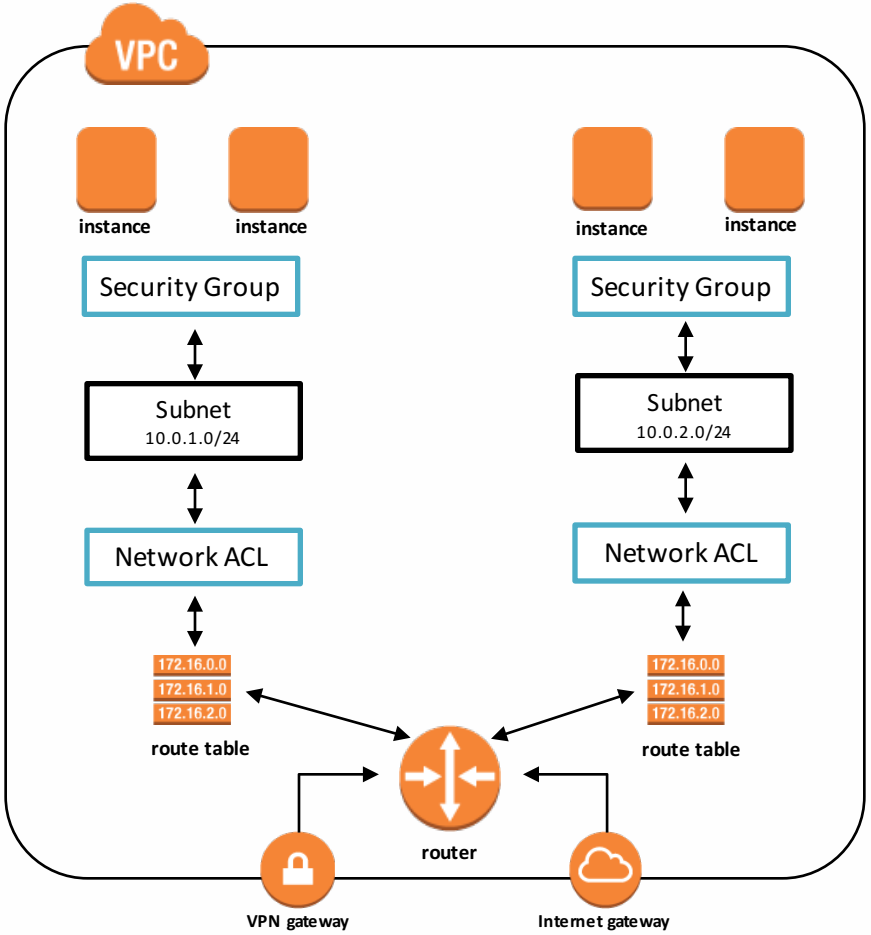
Amazon VPC: Networking



Amazon Web Services

Certified Developer – Associate Level

VPC Security


VPC
instance
instance
instance
instance
Security Group
Security Group
Subnet
10.0.1.0/24
Subnet
10.0.2.0/24
Network ACL
Network ACL
172.16.0.0
172.16.1.0
172.16.2.0
route table
172.16.0.0
172.16.1.0
172.16.2.0
route table
router
VPN gateway
Internet gateway

Amazon Web Services

Certified Developer – Associate Level

Amazon STS Concepts

AWS Security Token Service

Amazon Security Token Service (STS) allows you to grant a trusted user temporary and controlled access to AWS resources.

- Grant temporary access
  - To existing IAM users
  - To web-based identity providers: Facebook/Amazon/Google
  - To your organization's existing identity system

- Credentials are associated with an IAM access control policy that limits what the user can do

- Amazon STS API
  - AWS SDKs
  - AWS CLI
  - AWS Tools for Windows Powershell

AWS Security Token Service

- STS returns temporary security credentials
  - These consist of an **access key** and a **session token**
- Access Key
  - Consists of an access key ID and a secret key
- Session Token
  - Used to validate our user's temporary security credentials
- Credentials expire after a certain amount of time

AWS Security Token Service: Key Terms

- Federation
  - Creating a trust relationship between an identity provider and AWS
  - Users can sign into an identity provider like Amazon, Facebook, Google, or any other recognized provider

- Identity broker
  - The broker is in charge of mapping the user to the right set of credentials

- Identity Store
  - An identity store is something like Facebook, Google, Amazon, or Active Directory

- Identities
  - A user or “identity” within an identity store

Temporary Credentials with Amazon EC2

- Assign an IAM role to the EC2 instance
- Get automatic temporary security credentials from the instance metadata using the AWS SDKs/CLI
- You don't have to explicitly get credentials

Temporary Credentials with AWS SDKs

- Call the AWS STS API (*AssumeRole*) with your code
- Extract the credentials and session token, and use those values for future calls to AWS
- Make sure you renew credentials before the old ones expire (some SDKs do this for you)

Temporary Credentials with APIs

- Sign requests with your temporary security credentials that you get from AWS STS
- Use the access key ID and secret access key, and add your session token to the API request
  - Add the session token to an HTTP header
  - OR add it to a query string parameter named X-AMZ-Security-Token

Example Scenario

A corporate web application is deployed within an Amazon VPC, and is connected to the corporate data center via IPSec VPN. The application must authenticate against the on-premises LDAP server. Once authenticated, logged-in users can only access an S3 keyspace specific to the user.

Solution:

- Develop an identity broker to authenticate against LDAP
- Identity broker calls the STS API to receive temporary credentials
- Application can then access the temporary AWS permissions

Amazon Web Services

Certified Developer – Associate Level

Taking the Exam

Taking the Exam

- https://www.webassessor.com/wa.do?page=publicHome
- Register online or at an AWS event
- Associate level exams are $150 per exam
- Exams are taken at Kryterion exam centers

Amazon Web Services

Certified Developer – Associate Level

What Now?

What now?

- Congratulations!
- AWS Certified Solutions Architect Associate Level
- AWS Certified SysOps Administrator Associate Level
- Linux+/LPIC Level 1 Exam 101
- Get ready for our upcoming, and exciting, new courses!