**AWS Certified Developer Associate Certification 2017**

AWS certs are the most popular certifications in 2016. Partners drive certifications like technology partners and consulting partners. Use these notes in addition with CSAA

Identical sections:

* Section 6 – Databases – Overviews & Concepts
* Section 13 – VPC
* Route 53

Similar sections:

* Section 2 – AWS 10,000 Foot Overview
* Section 3 – IAM
* Section 4 – EC2
* Section 5 – S3
* Section 8 – SQS

<https://acloud.guru/course/aws-certified-developer-associate/dashboard>

<https://www.wired.com/2015/10/meet-walking-dead-hp-cisco-dell-emc-ibm-oracle/>

Exam focus:

* AWS Fundamentals – 10%
* Designing and Developing 40%
* Deployment & Security 30%
* Debugging – 20%

<https://media.acloud.guru/aws-cda/resource/b3d92ac6-91ee-4410-a71c-82e761f9442c_1:6/aws-cda-154e2655-4e24-49d3-bac6-75695dc47b91.pdf?Expires=1498450645&Signature=KQapB6fozQaV0XzuW/VVt4KrgPTjRxDOtHSo3DJlkDQwykF8SLEM8BHECItiOvoddDoUUGHPYZbMPrtZ7XZw53wFC9nba26UX5530dNu2NxTB173MHm00emjNajyD+IyIpRiCF6BmjtlxOOlH9NwZboDLewRxX4mAvy5g/sTn6YSRLDMkn/BdzaKd9w0GJBNK+AqHwNUjH46qaoec7/ip2gkuHSlj58OcZcdsjVmDwOEqfzi1P9D1WGbaRAxdrIl1nXIWg7Xl3O8sMmKAtP7vtqq0VT/pS37xyQvgttCP114XTbr9935g7OXlDiyNjcYnhO55DH+gcDZqBV0g6FCkw==&Key-Pair-Id=APKAISLU6JPYU7SF6EUA>

**AWS – History So Far**

* **AWS Infrastructure, compute, database, storage, network & content deliver, management tools, security & identity. Focuses on API calls and dynamoDB.**
* **Still need to read database, ec2, finish security whitepaper, s3, application services**
* **I’ve already read VPC, Route 53, possibly S3.**
* AZ could be a collection of datacenters.
* Elastic Beanstalk can spin up the underlying infrastructure when code is uploaded to deploy those services.
* DMS – SQL, MySQL, Oracle, Aurora, Postgres, SAP ESE.
* **SMS (Server Migration Service)** – migrate VmWare VM to cloud. 50 at the same time.
* **Athena** – SQL queries on S3
* **Cloud Search** – managed service to enable search capabilities on your website.
* **Config** – proactively monitor environment in AWS.
* **Step Functions** – a way to visualize what is going on in your application.
* **PinPoint** – enables and engage with your users. Gather data with users and the apps they use.

**IAM (Identity Access Management)**

* You can authenticate your services with **Web Identity Federation** like LinkedIn, FB, etc.
  + Done by STS
  + Found in IAM under “Web Identity Federation Playground”
  + API call is AssumeRoleWithSAML
    - Covered quite a bit in the quiz.

**EC2**

* Reserved – reserve capacity for continued workloads.
* Dedicated – Save costs by use server-bound licenses.
  + Useful for regulatory requirements
* EBS – put in an AZ to protect from storage array failures
  + Magnetic volumes is important for not frequently accessed data except for ST1
  + Magnetic Standard is basically cold but is bootable.
  + Can encrypt by using a third-party tool or by creating a copy and encrypting from that volume.
  + Can get data corruption if you are running an action while detaching a EBS volume.
  + Best to stop the instance when creating a snapshot although you can still take one while it is running.
  + Can change EBS volumes on the fly except magnetic standard drives.
  + Volumes must be in the same AZ. Can scale up only. Wait for 6 hours before making another change.
* Public key/private key are padlocks.
* System Status checks and instance status checks.
* 1 instance can have multiple security groups
* AWS command line
  + Best not to have a user but sometimes it is a must if your accessing the console remote via command line.
  + Use region command for pulling data from S3.
  + **Commands to know:**
    - **Aws configure**
    - **Describe-instances**
    - **Describe-images**
    - **Run-instances**
      * Use right key pair and security group
    - **Terminate-instances**
    - **--dry --run**
* Roles
  + Can attach existing roles to a running ec2 instance. Change happens immediately.

<https://aws.amazon.com/blogs/security/new-attach-an-aws-iam-role-to-an-existing-amazon-ec2-instance-by-using-the-aws-cli/>

<https://aws.amazon.com/blogs/security/easily-replace-or-attach-an-iam-role-to-an-existing-ec2-instance-by-using-the-ec2-console/>

Http responses can be done with API gateway using SDK’s with Lambda.

**Setting up PHP SDK:**

#!/bin/bash

yum update -y

yum install httpd24 php56 git -y

service httpd start

chkconfig httpd on

cd /var/www/html

echo "<?php phpinfo();?>" > test.php

git clone <https://github.com/acloudguru/s3>

curl –sS <https://getcomposer.org/installer> | php

php composer.phar require aws/aws-sdk-php

http://docs.aws.amazon.com/aws-sdk-php/v3/guide/getting-started/installation.html

* <https://aws.amazon.com/tools/> - Android, iOS, JavaScript, Java, .NET, PHP, Node.js, Python, Ruby, Go, C++
* Default Region is US-EAST-1.
* Some have a default regions (Java)
* Some do not (Node.js)

**ELB**

* ALB use target groups. ELB ties the instances from the AZ’s.

<https://aws.amazon.com/ec2/>

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/Storage.html>

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-instance-metadata.html>

**S3**

* Add random letter so they are evenly distributed across S3.
* Bucket name must start with a lowercase letter or number
* Bucket name must not contain uppercase characters
* Bucket name must be between 3 & 63 characters long

CORS Resource Sharing:

<!--CORS Code - Copyright ACloud.guru-->

<script src="https://code.jquery.com/jquery-1.11.0.min.js"></script>

<script src="https://code.jquery.com/jquery-migrate-1.2.1.min.js"></script>

<html><body><h1>Welcome to the Index Page!</h1>

<div id="get-html-from-other-s3"></div>

<script>

$("#get-html-from-other-s3").load("http://mycorstestbucketlondon.s3-website.eu-west-2.amazonaws.com/loadpage.html")

</script>

* Have to add the CORS configuration policy to the bucket to allow the main bucket to access the secondary bucket.

If you delete a version, it is gone forever. Objects can be restored.

Previous versions replicate on cross region replication. Delete markers for versions don’t replicate. Delete Markers are replicated. Object deletion do.

Previous versions with lifecycle are deleted permanently. Lifecycle can be used in conjunction with versioning.

**CDN**

* Origin – source of the content. S3/EC2/ELB. Can have your own default origin.
* Edge location separate from AZ. Edge locations are not just read only. Can be written in.
* Invalidations stop caches at edge locations. Cost’s money.
* Can whitelist/blacklist countries.
* Can upload files and not just download.

**Storage Gateway**

* Virtual tape library is backed by Amazon S3
  + Access is instantaneous
* Virtual Tape Shelf backed by Glacier.
  + If you need a lower RTO, go with virtual tape library based 24 hours.
  + Dell, Symantec, Microsoft, Veeam are supported.
* Need storage for local buffer and upload buffer. Open 443 and 80, 3260, 53.
* Can create stored volumes from snapshot.
* Stored gateway stored snapshots in S3
* Need more than 1 gateway using more than one location.

**Snowball**

* Snowball edge is not just storage but also has compute. Can run lambda functions

**Serverless**

* S3 should have the same name as the domain name to tie the website to a domain.

**DynamoDB**

* SSD storage
* GET/PUT operations
* Stored in 3 different locations
* Supports 1 byte up to 400 KB.
  + Data Consistency
    - **Eventual Consistent Reads (Default)**
      * Consistency reached within a second. Best Read performance
      * Consist of 2 reads per second
      * **Formula - (Size of Read rounded to nearest 4 KB chunk/4 KB) x # of items = read throughput**
        + **Dive by 2 if eventually consistent**
        + Read capacity units are rounded out in throughput
    - **Strongly Consistent Reads**
      * Returns a result that reflects all writes that received a successful response.
      * Consists of 1 read per second
* Items = rows
* Table – collection of data items.
* Attributes = columns
* Can nest 35 levels within a table.
* First 25 gb stored is free.
* Write is 10 units
  + All writes are 1 KB
  + All writes are 1 write per second
  + **Items \* KB per second = write units**
* Read is 50 units.
  + Free tier is 25 read/write units.
  + Rounded up to increments of 4 KB on throughput
* Can export to a CSV or export individual items
* Doesn’t do complex data joins.
* **If you exceed your throughput – 400 HTTP status code – provisionedThroughputexceededexception.**
* SimpleDB – another non relational database that has a storage limitation of 10 GB. Made for smaller workloads.
* API = CreateTable, UpdateTable, DeleteTable, DescribeTable, ListTables, PutItem, BatchWriteItem, UpdateItem, DeleteItem, GetItem, BatchGetItem, Query, Scan
  + 25 items supported in batches with total request size of 16 MB
* Supports Number, String, Binary, and Boolean
* Documents SDK – datatypes wrapper for JavaScript that allows easy interoperability between JS and DynamoDB datatypes.

Lab – DynamoDB Table:

#!/bin/bash

yum update -y

yum install httpd24 php56 git -y

service httpd start

chkconfig httpd on

cd /var/www/html

echo "<?php phpinfo();?>" > test.php

git clone <https://github.com/acloudguru/dynamodb>

Two types of primary keys:

* **Single attribute (unique id)**
  + Partition key (hash key)
    - Uses partition key as input to an internal has function. Output from has determines the physical location
    - No two items in a table can have the same partition key
* **Composite (unique ID and date range)**
  + Partition Key & Sort Key (Hash & Range)
    - Uses partition key as input to an internal has function. Output from has determines the physical location
    - Two items can have the same partition key but must have a **different sort key**
      * Same partition key is stored together.

**Indexes:**

* **Local Secondary Index**
* Same partition key, different sort key
* Can ONLY be created when creating a table. Can’t be removed or modified later
* Can have up to 5 per table
* **Global Secondary Index**
* Different partition key and different sort key.
* Can be created at table creation or added later.
* Can have up to 5 per table

**Streams:**

* Used to capture modification to the DynamoDB tables.
  + Captures an image of new item added plus attributes included
  + Captures before and after image of any attribute modified of an item.
  + Captures image of an item before it is deleted.
* Stores for 24 hours (maximum)

Query vs Scan:

**Query** – finds items in a table using only primary key attribute values. You must provide a partition attribute name and a distinct value to search for.

* Can provide a sort key attribute and use a comparison operator to refine the search results
* Returns all data attributes for items with specified primary keys. You can use **Projection expression** to only return some attributes.
* Sorted by the sort key. Always in ascending order with numbers. Test is done by ASCII character code. You can reverse order by setting **ScanIndexForward** parameter to false.
* Queries are eventually consistent. Can be changed to strongly consistent.
* Single response size is 1 MB

**Scan** – examines every item in the table. You can use projection expressions so scan only returns some of the attributes.

* Adds a step to remove data from the result set.
* Can be slow with large data sets.
* Design tables to use Query, Get or API’s.

**Web identity provider with DynamoDB:**

* Done by AssumeRoleWithWebIdentity API and other Open-ID Connect-compatible provider.
* Web Identity Token
* App ID of provider
* ARN of Role
* Provider responds with following
  + Access key ID, secret access key, Session Token, Assume Role ID, Expiration, Subject from Web identity token
* Steps to authenticate
  + User authenticates with ID provider
  + Passed token by their ID provider
  + Code calls Role API and provides token with ARN
  + App can now access DynamoDB between 15 minutes to 1 hour. (1 hour is default)

**Conditional Writes**

* If item = $10 then update to $12
* **Idempotent** – means you can send the same conditional write request multiple times, but it will have no further effect on the item after the first time DyanmoDB performed the update.
* ATTRIBUTE\_EXIST, CONTAINS, BEGINS\_WITH

**Atomic Counters**

* Where you use the UpdateItem operation to increment or decrement the value of an existing attribute without interfering with other write requests. (all write requests applied in the order in which they were received)
* **Not idempotent**

**Batch Operations**

* BatchGetItem API can retrieve up to 1 MB of data which may contain 100 items. Can retrieve items from multiple tables.

**Virtual Private Cloud (VPC)**

* Creating a VPC creates a SG, ACL, Route Table
* Nat Gateways scale up to 10GB.
* Need to add ephermal points to a custom VPC on the ACL’s.

**SQS (Simple Queue Service)**

* Queue is a temporary repository for messages that are awaiting processing
* Can contain up to 256 KB of text at any time.
* Process
  + Asynchronously pulls the task messages from the queue
  + Retrieves the named file
  + Processes the conversion
  + Writes the image back to Amazon S3
  + Writes a task complete message to another queue
  + Deletes the original task message
  + Checks for more messages in the worker queue.
* Visibility timeout doesn’t start until the application servers pulls the message down
* 12 hours visibility time out maximum. Default is 30 seconds.
  + Can be extended if you change the ChangeMessageVisibility value.
* Doesn’t offer FIFIO
* Billed at 64 KB chunks
* Can be delivered anytime or in any order.
* Maximum Long poll time out is 20 seconds.
* Can use SNS to fan out messages to other queues
* Need to implement application level tracking if application uses multiple queues
* Can be delayed for up to 15 minutes from a message standpoint
* 1 million requests for free tier
* **dead letter queues** – queues that received messages from other source queues.
* Messages can contain up to 10 metadata attributes.
* Tie in queue value can be requested by Sent Timestamp attribute
* SDK’s use the 20 second long poll default
* FIFO queues are limited to 300 transactions per second.
* Can enable SSE or KMS and encrypts the body of the message. Envelope encryption.
* PCI DSS certifies & HIPPA eligible.
* Larger messages can be sent using client library for Java.
* Names limited to 80 characters
* Supports anonymous access.

**SNS (Simple Notification Service)**

* Follows a publish-subscribe messaging paradigm.
* Delivered to SQS, email, email-JSON, text, HTTP
* Stored across multiple availability zones.
* **Topics** – access point for allowing recipients to dynamically subscribe for identical copies of the same notification.
* Subscriber must confirm first
* Messages can be customized for each protocol
* 1,000,000 requests on free tier is not charged. 100,000 over HTTP, 100 SMS, 1000 Email
* Can reuse a topic name.
* Doesn’t support FIFO SQS queues.
* More than one topic owner can allow users publish to a topic.
* Token confirmation messages last for 3 days.
* HIPPA eligible.
* Can’t delete a message once it has been published.
* SMS requires permissions from users to be sent to them.
* Transactional messages gets sent over routes with high reliability
* SNS comes from the same long code/short code called a sticky sender ID
* 100000 topics per account available and 10 million subscriptions per topic
* SMS contains up to 140 bytes. Total size limit is 1600 bytes
* **Direct addressing** – allows you to deliver notifications directly to a single endpoint rather than sending identity messages to all subscribers of a topic. Only is supported on mobile push endpoints.

**SWF (Simple Worflow Service)**

* **Workers** – programs that interact with SWF
* **Decider** – program that controls coordination of tasks
* Ensures task is assigned only once and never duplicated.
* **SWF Domain** – workflow, activity types, and execution. Container
* Maximum workflow can be 1 year. Measured in seconds.
* Keeps track of all the tasks of the applications.
* To coordinate application execution across workers, you write a program called the **decider** in your choice of programming language.
* SDK for this is AWS Flow Framework
* **Step functions** – fully managed service that is easy to coordinate the components of distributed applications and micro services using visual workflows.
* Video encoding is the #1 use case. 2nd use case is processing large product catalogs using Amazon Mechanical Turk
* Can setup domains for application resources
* Decision tasks contains information on inputs, outputs, and current state of previously initiated activity tasks
* Task list determine how tasks are assigned
* Workers use standard GET requests to get tasks from SWF
* Execution id’s limit tasks to be ran no more than once.
* Can have a maximum of 10,000 workflow and activity types. Can have 100 SWF domains
* Can have 100,000 open executions in a domain
* Execution history can grow up to 25,00 events
* 100 activity tasks
* 1,000 activity tasks per workflow execution
* Works across 3 availability zones

**CloudFormation**

* Used with different stacks
* Function:GetAtt retrieve specific outputs.
* Automatic rollback on error which causes everything successfully deployed up to the error to be deleted.
* Can have up to 60 parameters and 60 outputs in a template
* Template, Parameter, Output, Resource are limited to 4096 characters.

**ElasticBeanstalk**

* Don’t pay for using it but pay for the resources it configures.
* Uses unique domain names.
* Can SSH into the instance created.
* Can edit php.ini file but restarts application service.
* Can provide Access id and secret access id to access other services.
* Supports – ruby, python, Node.JS, IIS, PHP, Tomcat, Docker

**Shared Responsibility**

* AWS responsible
  + AWS endpoint, Global Infrastructure, Foundation services, AWS IAM
  + RDS/EMR don’t have access in the server via SSH/RDP only for infrastructure
  + S3/DynamoDB only for infrastructure
* Customer responsible
  + Customer data
  + Platform & Application management
  + OS, Network, Firewall, Encryption, Customer IAM

**Route53**

* Alias records are like CNAME records.
* Charged for CNAME access. Alias is not charged.
* Different NS types for failover if needed.
* Simple
* Weighted – does it over the length of the day
* Latency
* Failover
* Geolocation