Certified Developer Associate Notes

Welcome! The notes below should help you pass the AWS Developer Associate exam. These are my own course notes from the acloud.guru Developer Associate course.

I assume you have already taken a full course like the one at acloud.guru and just need a refresher on the AWS products covered in the exam. If you haven't taken an AWS class yet then I highly recommend you go over to acloud.guru and sign up for one! Their classes are well worth the money, and the practice exams there will help you ensure you are ready to sit the exam.

The notes below are grouped into categories by product or service. All the products mentioned here can (and will) show up on the exam.

ELB

There are 3 types of load balancers: - application (layer 7) - network (layer 4) - classic (layer 4/7)

- A 504 error means the gateway has timed out
- If the EC2 instances behind your load balancer need to know the IPv4 addresses of your users, enable the X-Forwarded-For header in the ELB settings.

Route 53

- Domains need a "hosted zone"
- Once a "hosted zone" is configured, you can set up DNS records (A, CNAME, etc..)
- One of Route 53's special features is the "alias" record type, which allows you to point at an AWS resource like an ELB or S3 bucket

S3

- File sizes can be from 0 B up to 5 TB
- Files are stored in *buckets* (like folders)
- Universal namespace, with globally unique (and more recently DNS compliant names)
- Consistency model is READ after WRITE for new PUTs
- Overwrites/deletes are "eventually consistent" (DELETE is not immediate - this is important!)

- Storage types: standard, IA, one-zone IA (20% less expensive than regular S3), Reduced Redudancy (no longer more cost effective than regular storage), Glacier (archive storage, long term)
- Glacier access times are 3-5 hours (but new feature: can be accelerated!)
- S3 supports several types of encryption:
  - SSE-S3 (AWS managed master key)
  - SSE-KMS (KMS service managed key)
  - SSE-C (client managed keys, the "key material" is provided by the customer)
- Client side encryption is also supported
- You can prevent unencrypted files from being uploaded using a bucket policy to force the x-amz-server-side-encryption flag to be set during upload requests
- CORS (Cross-Origin Resource Sharing): lets you access resources in Bucket A from a website or script that is hosted somewhere else (EC2, Bucket B)
  - When configuring CORS, always use the *buket website URL* not the regular bucket URL!

CloudFront

CloudFront is Amazon AWS's CDN.

Important terms:

- Origin (the site that hosts your content)
- Edge Loction (a CDN node near your user)
- Distribution (name given to the CDN - there are two types)
  - Web (static content acceleration)
  - RTMP (streaming video)
- Can upload objects to edge locations (called "transfer acceleration")
- Can run code at edge locations via Lambda@Edge
- Randomizing object names (avoiding sequential key names) improves GET/PUT performance (but this is NO LONGER NEEDED as of 2018)
  - Originally this was to limit the possibility of I/O contention, but is no longer necessary: it may still come up on the exam
  - When randomizing names, you add a random **prefix** not a random **suffix**

AWS Serverless Products and Services

Lambda

- Know which services are serverless
- Know what languages are supported
  - Java

    - Go
    - Node.js
    - Python
    - Ruby
    - .NET
- Lambda scales out (not up) automatically
- Lambda functions are independent, 1 event = 1 function
- Lambda is serverless
- Lambda functions can trigger other lambda functions
- Lambda can do things globally (not just one region)
- Remember as many of the AWS lambda triggers as you can
- Remember you can use X-ray for debugging

Lambda Triggers (as of February 2019)

- Amazon S3
- Amazon DynamoDB
- Amazon Kinesis Data Streams
- Amazon Simple Notification Service
- Amazon Simple Email Service
- Amazon Simple Queue Service
- Amazon Cognito
- AWS CloudFormation
- Amazon CloudWatch Logs
- Amazon CloudWatch Events
- AWS CodeCommit
- Scheduled Events (powered by Amazon CloudWatch Events)
- AWS Config
- Amazon Alexa
- Amazon Lex
- Amazon API Gateway
- AWS IoT Button
- Amazon CloudFront
- Amazon Kinesis Data Firehose

Lambda Version Control

- You can have multiple versions of lambda functions
- *Qualified* versions will use `$LATEST` in the name, unqualified versions do not have this
- Can split traffic among different versions of a lambda function using aliases
- Cannot split traffic to $latest, instead create an alias for $latest

API Gateway

- Remember what API Gateway *is* at a high level
- API Gateway has caching capabilities to increase performance
- Low cost, scales automatically
- Can throttle API calls to prevent attacks
- You can log results to CloudWatch
- If you are using Javascript/AJAX that uses multiple domains with API Gateway then you must enable CORS on API Gateway (remember CORS is *enforced by the client*)

Step Functions

- A great way to visualize your serverless application
- Step functions automatically trigger and track each step
- Step functions logs the state of each step so if something goes wrong you can track what went wrong and where

X-Ray

The SDK provides:

- Interceptors to add to your code to trace HTTP requests
- Client handlers to instrument AWS SDK clients that your application uses to call other AWS services
- An HTTP client to use to instrument calls to other internal and external HTTP web services

X-Ray integrates with:

- Elastic Load Balancer (ELB)
- Lambda
- API Gateway
- EC2
- Elastic Beanstalk

X-Ray supports these languages (same as Lambda):

- Java
- Go
- Node.js
- Python
- Ruby
- .NET

DynamoDB

AWS's NoSQL Database.

Local secondary index

- Can only be created when you are creating your table
- It has the same Partition Key as your original table (but a different sort key)
- Any queries based on this sort key are much faster using the index than the main table

Global Secondary Index

- Can be created when you create your table, or you can add it later
- Can have a different partition key *as well as* a different sort key

Exam Tips

Indexes enable much faster queries, but only on specific columns.

- Local secondary can only be created when you create your table
- Global secondary can be created at table creation time or later
- Global can use a *different* partition key and sort key, local can only use a *different sort key*

Scan vs Query API

Queries

- Query results are always sorted by sort key in ascending order (numeric) or by ASCII character code values
- You can reverse the sort order using the "ScanIndexForward" parameter
- By default all queries are eventually consistent (you can force strong consistency by explicity setting this at query time)

Scans

- A scan operation examines every item in the table
- By default returns all data attributes
- Use the "ProjectionExpression" parameter to refine the scan to only return the attributes you want

Query or Scan?

- Query is much more efficient (scan dumps the entire table then filters output)
- Scan operation takes longer as the table grows in size
- A scan operation on a large table can use up the provisioned throughput for the table in just one operation!
- You can optimize scans and queries by setting a smaller page size which causes fewer read operations
- You can use parallel scans instead of sequential scans by logically dividing a table or index into segments and scanning each segment in parallel

DynamoDB Provisioned Throughput

- Measured in Capacity Units
  - 1x Write Capacity Unit = 1x 1 KB write per second
  - 1x Read Capacity Unit = 1x Strongly Consistent read of 4 KB per second
    - OR, alternatively, 2x Eventually Consistent reads of 4 KB per second (**this is the default**)

Provisioned Throughput Example

A DynamoDB table provisioned with 5x Read Capacity Units and 5x Write Capacity Units

This configuration can perform: - 5x 4 KB strongly consistent reads = 20 KB per second (or 40 KB per second eventually consistent) - 5x 1 KB writes = 5 KB per second

DynamoDB On-Demand Capacity

- With on-demand, you don't need to specify your requirements
- Great for unpredictable workloads
- Can change between on-demand and provisioned capacity *once per day*

DynamoDB Accelerator (or DAX) is a fully managed, clustered in-memory cache for DynamoDB

- Delivers up to a 10x read performance improvement
- Microsecond performance for millions of requests per second

- Ideal for read-heavy and bursty workloads (gaming, auctions, black friday sales)

When NOT to use DAX

- Don't use for applications which require strongly consistent reads
- Write intensive applications
- Applications that don't perform many reads
- Applications that don't require microsecond response times

DynamoDB Transactions

- ACID (atomic, consistent, isolated, durable)
- Read or write multiple items across multiple tables as an all-or-nothing operation
- Check for a prerequisite condition before writing to a table

DynamoDB TTL

- TTL (time to live) defines an expiry time for your data
- Expired items marked for deletion (will delete within the next 48 hours)
- Great for removing irrelevant or old data:
  - Session data
  - Event logs
  - Temporary data

DynamoDB Streams

- Time-ordered sequence of item-level modifications (insert, update, delete)
- Logs are encrypted at rest and stored for 24 hours
- Accessed using a dedicated endpoint
- By default Primary Key is recorded
- Before and after images can be captured
- This is a good event source for lambda (as events are recorded in near real time)

Exceeding provisioned throughput

- You will get a *ProvisionedThroughputExceededException*
- If you are using the SDK, retries will be handled automatically

- If not, you can:
  - Reduce request frequency
  - Use *exponential backoff*

DynamoDB Summary

- Amazon DynamoDB is a low-latency NoSQL DB
- Consists of tables, items, and attributes
- Supports both document and key-value data models
- Supported document formats are JSON, HTML, XML
- 2 types of primary keys: partition key and a combination of partition key + sort key (composite key)
- 2 consistency models: strong / eventual consistency
- Access controlled using IAM policies
- Fine-grained access control using the IAM "dynamodb:LeadingKeys" parameter to allow users access only to items where the partition key value matches their user ID
- Remember local secondary index vs global secondary index
- Query operation finds items in a table using only the Primary Key attribute
- Scan examines every item in the table
  - By default returns all atributes
- Can use ProjectionExpression parameter to refine results (or "filter" in the console)
- Query ops are always sorted by the sort key *in ascending order*
  - Set *ScanIndexForward* to false to reverse sort order, and this is **only for queries not scans**
- Reduce scan / query impact by:
  - Reducing page size (fewer read ops)
  - Isolate scan operations to specific tables and segregate from misison-critical traffic
  - Try parallel scans rather than default sequential scans
  - Avoid scan operations whenever possible
- Throughput measured in Capacity Units
  - 1 x write CU = 1 x 1 KB write per second
  - 1 x read CU = 1 x 4 KB strongly consistent read per second (or 2x 4 KB eventually consistent)

Calculate write capacity requirements

1. (size of each item) / 1 KB (for write CU)
2. Round up step 1 answer to next whole KB number
3. Multiply step 2 by number of writes needed per second

Calculate read capacity requirements

1. (size of each item) / 4 KB
2. round up answer in step 1 to nearest whole number
3. multiply by reads per second (for strong consistency)
4. divide answer in step 3 *by two* if eventual consistency is OK

Elasticache

In memory database caching.

- Good for read-heavy workloads
- Good for slow-changing data

2 types (Memcached and Redis)

Memcached

- Multi-threaded
- No multi-AZ capability
- Widely adopted

Redis

- Open source in-memory key-value store
- Supports complex data structures: sorted sets and lists
- Supports master/slave replication and multi-AZ for cross-AZ redundancy

Caching strategies

- Lazy loading: return "null" if object is not available or expired, it is then the application's responsibility to fetch that item from the source and write it into the cache

Advantages

- Only requested data is cached
- Node failures are not fatal (just a lot of cache misses if a node fails)

Disadvantages

- Cache miss penalty on initial request
- Stale data: if data is only updated on a miss, data can become stale (no automatic updates)
  - Note: we can set a TLL (time to live) for cached data to help deal with this

Write Through

Add or updates data in the cache whenever data is written to the DB

Advantages

- Data in the cache is never stale
- Users are more tolerant of additional delay at write time (as compared to read time)

Disadvantages

- Write penalty: writes involve an extra step (write to cache)
- If a node fails, data is missing until added or updated in the database (mitigate by *also* implementing Lazy Loading in conjuction with write-through)

Elasticache Exam Notes

- In memory cache
- 2 different caching strategies:
  - Lazy loading: cache data only when requested
    - Data can go stale
    - Soft failure (failed cache node just means cache misses)
  - Write through: writes data into cache whenever database changes
    - Write penalty at write time
    - Data is missing from cache until added/updated in DB
    - Wasted resources if most data is not accessed

KMS

- Encryption keys are managed through the IAM console
- You can generate keys in KMS or import your own "key material"
- Best practice is to have one user who manages the key(s) and one user who can encrypt/decrypt using the key(s): this should not be the same person

KMS Exam Tips

Know these 4 API calls:

- aws kms encrypt
- aws kms decrypt
- aws kms re-encrypt
- aws kms enable-key-rotation

KMS Envelope Encryption

Evenlope encryption is the process of encrypting your envelope key (your data key).

Customer Master Key -> decrypt the data key (envelope key) -> envelope key decrypts your data

Keys in KMS can *never* be exported. Keys in CloudHSM *can*. Note, CloudHSM is **dedicated HSM hardware**.

SQS

A messaging queue service. A way to decouple services from one another.

An Example

Imagine a meme generator:

- User uploads image to S3
- Triggers a lambda function that creates an SQS job containing (image URL, top meme text, bottom meme text)
- EC2 instances read jobs from the SQS queue, generate memes, and save them back to S3

This way if an EC2 instance dies, things keep working because SQS decouples the jobs to be done (meme data) from the instances performing them (in EC2).

Exam note: **SQS is always a pull-based system**

SQS can also provide some elasticity: jobs can wait in the queue if the workers pulling from the queue are not processing work fast enough. Auto Scaling can also be configured to increase the number of workers as the queue begins to fill up.

SQS Queue Types

- Standard queues:
  - Default type
  - Messages may be delivered out-of-order ("best effort" in-order delivery)
  - A message may be delivered more than once
- FIFO queues:
  - First in, first out (guaranteed delivery order)
  - Messages always delivered once, and messages are available until processed and deleted by a worker
  - Limited to 300 transactions per second

Important SQS Notes

- Message sizes can be up to 256 KB
- Can be queued from 1 minute to 14 days
- SQS guarantees messages are processed at least once
- Invisibility timeout: length of time a message is invisible after a worker pulls it
  - Maximum invisibility timeout is 12 hours

SNS

- A push-based notification services. Can send out push notifications to Apple, Google, Fire OS, Windows, etc… (even compatible with Baidu Cloud Push).
- Can deliver notifications via email, SMS text messages, SQS queues, or an HTTP endpoint
- Lambda functions can subscribe to SNS functions and will be triggered when an SNS push is sent out
- SNS delivers appropriately formatted copies of your message to each subscriber, even for different tech types (i.e. you have iOS, Android, Lambda, and more all subscribed to the same topic)
- Instantaneous push-based delivery (no polling)

SES (Simple Email Service)

An email service to help marketing teams and app developers send marketing, notification, and transactional emails to customers using a Pay-As-You-Go model.

- Can also be used to receive emails, which can be delivered to an S3 bucket

- Incoming mail can trigger SNS notifications
- Compare w/ SNS: many formats supported by SNS, SNS can fan out messages to large numbers of recipients, etc…
- Key difference w/ SNS: SES is *NOT* subscription based, you can send to *anybody*

Kinesis

A service to process streaming data.

Streaming data is generated continuously by thousands of data sources which send records simultaneously and in small sizes (KB-level). Think:

- Stock prices
- Purchases from online stores
- Game data (as players play)
- Social network data
- Geospacial data (GPS datapoints)

Kinesis Streams

Kinesis streams can store data from 24 hours up to 7 days (**24 hours is the default**).

Data is stored in something called a "shard":

- Consumers (EC2 instances, for instance) take the data from the shards and convert it into some end format
- Consumers then store their final results somewhere else (Redshift, DynamoDB, etc…)

Shards: - 5 transactions per second for reads - Maximum total data read rate of 2 MB per second and 1,000 records per second for writes, with a maximum total write rate of 1 MB per second - The data capacity of your stream is a function of the number of shards specified for the stream

Kinesis Firehose

- Producers send data to Firehose
- Firehose doesn't make you worry about shards/streams (automated!)
- Firehose lets you analyze data using Lambda in real time
- Data can be stored into S3

Firehose has no automatic data retention window (because you store data directly to S3)

Kinesis Analytics

Allows you to run SQL queries on your data as it exists inside Firehose or Streams, and you can then store results into S3, Redshift, or Elasticsearch Cluster

Exam tips

- Know firehose / streams key differences
- Understand what Kinesis Analytics is

Elastic Beanstalk

A service for deploying and scaling web applications written in many languages (Java, .NET, PHP, NodeJS, and more) and deploying onto popular application platforms like Apache Tomcat, NGINX, Passenger, and IIS.

Basically, you give Elastic beanstalk a ZIP file with your code in it, and it provisions the necessary resources for you (it auto-detects them).

Updating Elastic Beanstalk

All-at-once deployment updates:

- Deploys the new version to all instances simultaneously
- All of your instances are out of service while the deployment takes place
- You will experience an outage while the deployment is taking place: not ideal for mission-critical systems
- If the update fails, you need to roll back the changes by re-deploying the original version to all your instances

Rolling deployment policy:

- Deploys the new version in batches
- Each batch of instances is taken out of service while deployment takes place
- Environment capacity is reduced by the number of instances in a batch, while deployment takes place
- Not ideal for performance-sensitive systems
- If the update fails, you need to perform an additional rolling update to roll back the changes

Rolling with additional batch:

- Same as above, but deploys an additional batch during deployment

Immutable deployment updates:

- Deploys the new version to a fresh group of instances in their own new Auto Scaling group
- When new instances pass their health checks, they are moved to the existing auto scaling group, then old instances are terminated
- Impact of a failed update is lower and rollback only requires terminating new instances
- Preferred option for mission-critical applications

Exam tips

Remember the 4 different deployment approaches and where they are used:

- All at once
- Rolling
- Rolling with additional batch
- Immutable
  - Preferred option for mission critical production systems
  - Maintains full capacity
  - To roll back, just delete the newly created instances and their Auto Scaling group

Configuring Elastic Beanstalk

You can use YAML or JSON to configure your Elastic Beanstalk environment. You need to save your configuration in a file with the extension .config and the file(s) must be stored in a directory called .ebextensions in the top level directory of your application.

Elastic Beanstalk with RDS

There are two ways to integrate RDS with Elastic Beanstalk:

1. Let EB directly launch RDS (great for dev and test)
2. Manually configure RDS (better for production - RDS is decoupled from the application environment)
   - Can connect multiple environments to the same DB
   - Provides a wider choice of database types
   - Allows you to remove application environments without affecting your RDS instance

How do you manually configure RDS and EB to work together? - An additional security group must be added to the environment's Auto Scaling group - You need to provide connection info (url, username, pass) as EB environment variables

AWS Systems Manager

If you have info like passwords, DB connection strings, license codes, etc… you can store this in SSM (systems manager) Parameter Store.

- You can store values as plaintext or encrypted
- You can reference values using their names
- You can use this service with EC2, CLoudFormation, Lambda, EC2 Run Command etc…

Other AWS Services (Summary)

SQS

Distributed message queueing system:

- Pull based not push based
- Standard queues
- FIFO queues
- Visibility timeout (default 30 seconds, maximum 12 hours)
- Two polling types:
  - short-polling (returns immediately even if no messages)
  - long-polling (waits until a message is in the queue or timeout period is reached)

SNS

- Scalable notification service
- Pub/sub model (users subscribe to topics)
- Push based rather than pull based
- Can send notifications to multiple people (can do *fan-out*)

SES

SES sends **emails only**.

- Can be used for incoming and outgoing mail
- Is not subscription based: can send to any valid email address

Kinesis

Know the difference between:

- Kinesis streams
  - Data streams

    - Video streams
- Kinesis firehose

Note: **Kinesis Analytics** analyzes data as it comes in.

Note: you can configure lambda to subscribe to Kinesis Streams to execute some action before sending data onwards.

Elastic Beanstalk

Automatically deploy and scale web apps.

- Support for PHP, Python, Ruby, Go, Docker, .NET, Node.js
- App server platforms (Tomcat, Passenger, Puma, IIS)
- Can fully manage EC2 instances yourself or let Elastic Beanstalk do it for you

Remember update types:

- All at once
- Rolling
- Rolling with additional batch
- Immutable

Advanced Elastic Beanstalk Tips

- You can customize your EB environment by adding config files written in YAML or JSON
- Files must have a .config extension
- Files must be saved in an .ebextensions folder at the top-level-directory that holds your application code

CI/CD

CI/CD = Continuous Integration / Continuous Deployment

CI/CD lets you automate deployment and some aspects of testing, allowing frequent small changes to software to be tested and deployed very frequently into production.

"CD" can mean:

- Continuous Delivery
- Continuous Deployment

The difference is that in *continuous delivery*, after testing takes place the decision whether or not to deploy is made manually (by a human). In *continuous deployment*, the deployment step is also automated.

- AWS CodeCommit
  - AWS code repository service (private git repository)
- AWS CodeBuild
  - Managed build service
- AWS CodeDeploy
  - Can deploy your code to EC2 instances, lambda functions, and even your on-premise servers
- AWS CodePipeline
  - Orchestrate the above 3 activities to fully automate the release of new software

Exam Tips

Read the AWS whitepaper on DevOps best practices!!! It might also be a good idea to AWS's whitepaper on blue/green deployments.

- CI is about integrating or merging code changes frequently
  - At least once per day
  - Enables multiple developers to work on the same application
- Continuous Delivery is about automating build, test, and deployment (but human intervention is needed to deploy into your environment)
- Continuous Deployment is Continuous Delivery + automated deployment

AWS CodeCommit

- Users create a copy (known as a branch) of the master repository, which they update independently without impacting other users
- Saved code changes which are ready to be applied to a repository are known as a "commit"
- When the updated code located in a branch is ready to be added to the master repository, the branch is **merged** into the master repository
- AWS CodeCommit provides all the functionality of Git

Other cool stuff

You can set up CloudWatch event rules to notify you (or other people) when changes (such as a pull request) take place in CodeCommit

CodeDeploy

An automated deployment service which lets you automatically deploy code to EC2, Lambda Functions, or even on-premise systems.

- Scales with your infrastructure

- Integrates with CI/CD tools (Jenkins, Github, Atlassian, AWS CodePipeline)

Two deployment approaches: - In-place - Blue/Green

In-place

Application is stopped on each instance in turn, and the latest revision is installed

If the instances are behind a load balancer, you can stop sending requests to instances being upgraded, and start sending requests again once the instance is updated

Note: this is *not supported for lambda*

Blue/Green

New instances are provisioned and the latest revision is installed on the new instances. Blue represents the active deployment, green is the new release.

- Switching back to the original environment is instantaneous
- Switching over to the new environment is also very fast (just route your load balancer to the new machines, release the original ones)
- No reduction in performance or capacity

Terms to know

- Deployment Group
- Deployment
- Deployment Configuration
- AppSpec File
- Revision
- Application

Exam Tips

Remember the different types of deployment:

- In place (rolling update)
- Blue/Green

CodePipeline 101

AWS CodePipeline can orchestrate the build, test, and deployment of your application.

This speeds up testing, feature adding, and bug fixing for your developers.

Integrates with:

- CodeCommit
- CodeBuild
- CodeDeploy
- Lambda
- Elastic Beanstalk
- CloudFormation
- Elastic Container Service
- Outside tools like GitHub and Jenkins

Each time you update your code, the pipeline is triggered. A failure at any stage results in stopping / a rollback, making failures easier to detect and fix.

Exam Tips

- Automates your end-to-end software release process
- Can automatically trigger your pipeline as soon as a change is detected in your source code repository
- Integrates with services like CodeBuild and CodeDeploy as well as other services (GitHub, Jenkins, etc…)

The AppSpec File

Used to define the parameters that will be used for a CodeDeploy deployment.

The file structure depends on whether you are deploying to Lambda or EC2 / On-premises.

For Lambda, the AppSpec file may be written in YAML or JSON and contains the following fields:

- version: reserved for future use, currently the only allowed version is 0.0
- resources: the name and properties of the lambda function to deploy
- hooks: specifies lambda functions to run at set points in the deployment lifecycle to validate the deployment e.g. validation tests to run before allowing traffic to be sent to your newly deployed instances

Exam tips

- Know what the AppSpec file does (defines deployment parameters)
- The appspec.yml file must be placed in the root dir of your revision and must be written in YAML for EC2 / On-premise deployments (can be in JSON instead for Lambda)
- **Know the order of hooks in a CodeDeploy deployment!**

Docker

An open source technology for creating applications based on Windows or Linux "containers"

A container is a lightweight standalone executable package containing all the stuff your software needs to run: the runtime environment, supporting tools, configuration etc…

Amazon Elastic Container Service (ECS) lets you run Docker containers on AWS.

Exam Tips

Understand the build, tag, and push commands (docker) to put our docker image into our docker repository.

You can put your own build commands into the CodeBuild console, to override or replace buildspec.yml.

CloudFormation

A service that allows you to provision AWS infrastructure as code.

Resources are defined using a CloudFormation template (both YAML and JSON are supported).

Benefits:

- Configure infrastructure in a really consistent way
- Less time and effort than configuring things manually
- You can version control and peer review your templates
- Free to use (you are charged for what you create)
- Can be used to manage + update dependencies
- Can be used to roll back and delete the entire stack as well (handy if you are just doing testing)

CloudFormation Template Format

Template parameters include:

- AWSTemplateFormatVersion
  - This is the first line in your template
  - Only ONE supported value: "2010-09-09"
- Outputs
  - Lets you define output data (such as EC2 instance IDs) which will be printed by your script when it finishes running

You can store reusable pieces of YAML and JSON code in S3, and can reuse them from within CloudFormation.

Stuff to remember

Remember the main sections in the CloudFormation template:

- Parameters: Input custom values
- Conditions: Provision resources based on environment
- Resources: AWS resources to create
- Mappings: Create custom mappings like region->AMI
- Transforms: reference code located in S3

Introduction to SAM

SAM = Serverless Application Model

Has its own CLI interface called the SAM CLI. You get several commands:

- sam package: takes a template (YAML) as input to package a deployment
- sam deploy: takes the sam template and an IAM role, which allows CloudFormation to create an IAM role to allow the function to execute

Exam Tips

Know that SAM is the Serverless Application Model and it is an extension to CloudFormation to allow you to deploy serverless applications.

CloudFormation Nested Stacks

Allow you to re-use nested stack elements (say for a load balancer or web application server group).

You can store these common elements in their own CloudFormation templates, then reference them from other CloudFormation templates.

Developer Theory Summary

CI/CD and related tools are worth a few exam points: read the AWS whitepapers on the subject:

- Developer Theory Whitepaper
- Blue/Green deployment

Other things to know:

- CI is about automating the process of merging changes into a code base
- CD is about automating build, test, and deploy parts of the release lifecycle

- Continuous Deployment fully automates the release process as well (as opposed to Continuous Delivery)

AWS tools to know:

- AWS CodeCommit
- AWS CodeBuild
- AWS CodeDeploy
- AWS CodePipeline

AWS CodeCommit:

- Based on Git
- Tracks and manages code changes

AWS CodeDeploy:

- Automated deployment of code (can be triggered by changes in CodeCommit)
- Two deployment types: in-place (rolling update) or Blue/Green

The AppSpec file defines all the parameters needed for deployment.

For EC2/On-premises systems, the appspec.yml file must be placed in the root directory of your application revision.

Try to remember the RUN ORDER of the hooks in the CodeDeploy deployment:

- Phase 1:
  - BeforeBlockTraffic -> Block Traffic -> AfterBlockTraffic
- Phase 2:
  - Application Stop
  - BeforeInstall
  - Install
  - AfterInstall
  - ApplicationStart
  - ValidateService
- Phase III:
  - ApplicationStart
  - ValidateService
  - BeforeAllowTraffic -> AllowTraffic -> AfterAllowTraffic

AWS CodePipeline Exam Tips:

CodePipeline is a CI/CD service. It automates your end-to-end release process based on a user-defined workflow.

It can be configured to automatically trigger your pipeline as soon as a change is detected in your source code repository.

Docker and CodeBuild Exam Tips

Docker lets you package your software into containers which you can run via AWS ECS.

A docker container includes everything the software needs to run including code, libraries, runtime and environment variables.

A Dockerfile specifies the instructions needed to assemble your docker image.

There are some docker commands you should know to build, tag, and push images to an ECR repository:

- `docker build -t myimagerepo .`
- `docker tag myimagerepo:latest <image tag>`
- `docker push <image tag>`

CodeBuild Exam Tips

A fully managed build service. It can build source code, run tests, and produce software packages based on commands that you define yourself.

By default the buildspec.yml defines the build commands and settings used by CodeBuild to run your build

You can override the buildspec.yml by adding your own build commands in the console when you launch the build

If your build fails, check the build logs as they appear at the bottom of the CodeBuild console (or view the full build logs in CloudWatch)

Web Identity Federation

You can allow sign-in and temporary access to AWS resources via Facebook, Google, or Amazon.

Amazon Cognito service provides this type of identity federation, plus:

- Access for guest users
- Sign in and sign up
- Acts as an identity broker between your application and web ID providers
- Synchronizes user data for multiple devices
- Recommended for all mobile applications accessing AWS services

Cognito means applications don't need to embed or store AWS credentials locally, and users get a seamless experience across all mobile devices.

Cognito User Pools

- User directories used to manage sign-up and sign-in functionality for mobile and web applications.
- Successful authentication generates a number of JSON Web tokens (JWTs).
- Cognito tracks the association between user identity and various devices they sign in front.

To provide a seamless experience, Cognito uses push synchronization to push updates and synchronize user data across multiple devices.

SNS is used to send a silent push notification to all devices with a given user identity, whenever data stored in the cloud changes.

Advanced IAM Policies

There are 3 types of policies:

- Managed policies
- Customer managed policies
- Inline policies

Managed Policy

An IAM policy created and administered by AWS. Provided for common use cases based on job function (DynamoDB Full Access, EC2 Read Only, etc).

A single managed policy can be attached to multiple users, groups, or roles.

You *cannot* change the permissions which are part of AWS managed policies.

Customer Managed Policy

Can be attached to multiple users, groups, and roles, but only within your own account. Created and administered by *you*.

Recommended for cases where existing policies don't quite meet your needs.

Inline Policy

An inline policy is an IAM policy which is actually embedded within the user, group, or role to which it applies.

Cannot be attached to multiple users, groups, or roles.

If you delete the user/group/role the policy is embedded in, then the policy will be deleted.

Typically Managed Policies are recommended, but in certain cases where you need to ensure a policy applies to **NO ONE ELSE** except a specific user/group, or role, then these are useful.

STS:AssumeRoleWithWebIdentity

This is what Cognito uses "under the hood" to authenticate clients who have already authenticated with a third party like Facebook, Google, or (of course) Amazon.

Once these services provide a JWT token, STS can exchange the JWT token for an STS token using the `AssumeRoleWithWebIdentity` API call.

Typically for mobile applications, AWS recommends you use Cognito. However, you can also directly use `AssumeRoleWithWebIdentity` in regular web applications.

Cross Account Access

Lets you "switch role" from one account to another inside the AWS console, without logging out and logging back in (uses the "assume role" functionality).

Steps:

- Identify your account numbers
- Create a group in IAM called "Dev"
- Create a user in IAM called "Dev"
- Log in to "Production"
- Create the "read-write-app-bucket" policy
- Create the "UpdateApp" Cross Account Role
- Apply the newly created policy to the role
- Log in to the Developer Account
- Create a new inline policy
- Apply it to the Developer group
- Log in as "John (dev user)"
- Switch accounts

Advanced IAM Summary

- Federation allows users to authenticate with a Web Identity Provider (Google, Facebook, Amazon).
- The user first authenticates w/ the web ID provider, they then receive an authentication token which is exchanged for temporary IAM credentials

- Cognito is an identity broker which handles interaction between your applications and the Web ID provider (you don't need to write your own code to do this)
    - Provides sign-up, sign-in and guest user access
    - Syncs user data for a seamless experience across devices
- Cognito uses Push Synchronization to send a silent push notification of user data updates to multiple devices associated with a user ID

Remember the 3 types of IAM policies:

- Managed Policy
- Customer Managed Policy
- Inline Policy

In most cases, AWS recommends using Managed Policies over Inline Policies

CloudWatch

Can monitor all kinds of things:

- Storage gateway
- CloudFront
- DB and Analytics
- RDS Instances
- EMR Job Flows
- Redshift
- SNS Topics
- OpsWorks
- Estimated charges on bills

And more!

Things CloudWatch monitors by default on EC2:

- CPU
- Network
- Disk
- Status Check

These are called "host level metrics".

Exam tip: RAM utilization is a custom metric! EC2 monitoring is on a 5 minute interval unless you enable detailed monitoring (1 minute intervals) which costs money

How long are CloudWatch metric stored? By default log data is stored *indefinitely.*

You can retrieve data using 3rd party tools or the `GetMetricStatistics` API

Metric granularity *depends on the AWS service.* It could be 1 minute, 3 minutes, or 5 minutes depending on the service.

Custom metric *minimum granularity* is one minute.

Alarms

You can create alarms to monitor any CloudWatch metrics in your account. This can include EC2 CPU Utilization, Elastic Load Balancer Latency, or even charges on your AWS bill.

Note: CloudWatch **can be used on premises**: it is not limited to just AWS resources. There is a CloudWatch Agent + SSM Agent which you can download and install.

Exam Tips

Host level metrics are:

- CPU
- Network
- Disk
- Status Check

CloudWatch vs CloudTrail vs Config

- CloudWatch monitors performance (CPU, RAM, etc. . .)
- CloudTrail monitors API calls (provisioning a new EC2 instance, new S3 bucket, tracks who-what-when)
- AWS Config records the state of your AWS environment and can notify you of changes

Stuff I Got Wrong During Practice Exams

Monitoring and Troubleshooting

- Lambda max execution time is 900 seconds
- For questions about "sticky sessions" and storing the session data, choose *ElastiCache*
- X-Ray works with ECS, Elastic Beanstalk, Lambda, EC2 (NOT S3)

Deployment

- SQS queue maximum retention time: 14 days
- Multi-part uploads are recommended for files over 100 MB
- OrderDate would make a good sort key in DynamoDB (why? will make it easy to query later: sorts + queries are the job of the sort key)
- If you need to roll back a lambda function, point the PROD alias to an older version of your function (you cannot change $LATEST)

CloudFormation

Get the DNS name of a LoadBalancer using CloudFormation code

```
Fn::Join" : [ "", [ "http://", { "Fn::GetAtt" : [ "ElasticLoadBalancer",
"DNSName" ]}]]
```

What will you see if your CloudFormation stack encounters an error?

ROLLBACK_IN_PROGRESS

When will a template NOT roll back?

When there is invalid JSON syntax in the template.

Choosing different instance types based on environment type

This can be achieved in your CloudFormation template via:

- Conditions
- Mappings

CodeCommit

To receive an email when changes are pushed, you must:

- Configure notifications in the console (this auto-creates an SNS topic which triggers emails to be sent out)

Development with AWS Services

- Elastic Beanstalk supports Tomcat, Docker, and Passenger but **not JBoss**
- appspec.yml goes in the root of your applicatoin source, **not the .ebextensions folder**

- Session state can go in *DynamoDB* or *ElastiCache*: RDS and EC2 are poorer choices, and Lambda functions can't store data

Kinesis

Records a processed by consumers according to the **sequence number** assigned when items are written to the stream

Lambda

RDS is not an event source for Lambda!

DynamoDB

- You can list tables using `ListTables`
- DynamoDB uses **optimistic concurrency control**
- Items in a DynamoDB table can have an unlimited number of attributes
- **DynamoDB Streams** allows you to capture a time-ordered sequence of modifications to items in a DynamoDB table over the past 24 hours

SQS

The API can return a status code 400 for each of these errors:

- MissingParameter
- MissingAction

Dealing with messages which are bigger than the SQS message maximum

The way to handle this is to use the SQS Extended Client Library to send an SQS message which references a message body in S3.

Billing

Message size maximum is 256 KB but **messages are billed at 64 KB increments**.

SWF

- EC2 instances can perform a worker task
- A server living outside AWS can perform a worker task
- Decision tasks occur when the state of the workflow changes

Refactoring

- RDS and ElastiCache are both OK places to store session state, but locally in memory or on EC2 are poor choices

Security

- Cognito sign up and sign in is managed by **user pools**
- CloudFront can force HTTPS by setting the **Viewer Protocol Policy**
- When deploying with Elastic Beanstalk, you need to create a security group allowing access to your RDS DB and add it to the auto-scaling group: your app also needs to know the DB connection string
- **Again: sign up and sign in for Cognito are handled by the user pool**