AWS Certified Developer Associate June 2018 Exam Study Guide / Notes

Domain	% of Exam
Deployment	22%
Security	26%
Development with AWS Services	30%
Refactoring	10%
Monitoring and Troubleshooting	12%

- 130 mins
- 65 questions: Multiple Choice / Multiple Response
- minimum passing score 720/1000
- Valid for 2 years
- Scenario based questions

EC2

Provides resizable compute capacity in the cloud. Reduces the time required to obtain and boot new server instances to minutes, scale capabity up and down as requirements change.

- On-Demand - fixed rate by the hour. no commitment
- Reserved - capaticy reservation at significant discount.
- Spot - flexible start and end times, enables you to bid whatever price you want for instance capacity, only feasible at very low compute prices. users with urgent need for large amount of additional computing capacity
    - If AWS kills your spot instance, you aren't charged for the hour. Only charged if you kill the instance.
- Dedicated Hosts - useful for regulatory requirements. can be on-demand. existing software licenses.

EBS

create storage volumes and attach them to EC2 instances. they are AZ locked.

- General Purpose SSD - GP2 - for less than 10,000 IOPS
- Provision IOPS SSD - desciened for i/o intensive apps.
- Throughput Optimizied HDD
- Cold HDD
- Magnetic

always use roles

IAM

Manage users and their level of access to AWS Console

- **Users** = end users (people)
- **Groups** = collection of users under one set of permissions
- **Roles** - create roles and then assign them to AWS resources
- **Policies** - a document that defines one or more permissions
- IAM is universal, not region-by-region.
- "root" has complete Admin access. Setup MFA for root account and then let only 1 Admin have access to it.
- New users have ***no*** permissions when first created.
- Access key id / secret key ≠ aws mgmt console login
- Always setup MFA

Rule of Least Privilege - give users the smallest amount of privileges to your system and only add what is necessary.

Roles:

- Create roles and apply them to EC2 instances to interact with AWS services on your behalf. Better than hardcoding secret keys into the instance.
- Roles allow you to not use Access Key ID's and Secret Access Keys
- Roles are preferred from a security perspective *exam question*
- Roles controlled by policies - JSON - key pairs
- Changes to role policies take affect immediately
- Attach/detach roles to running EC2 instances without having to stop/terminate them. *This is new.*

RDS

RDS - OLTP

- SQL
- MySQL
- PostgresQL
- Oracle
- Aurora
- MariaDB

DynamoDB - NoSQL RedShift - OLAP

Database:

- collection - table

- document - row
- key value pairs - fields

JSON / NoSQL

In non relational DBs, you don't need to know how many columns ahead of time.

Data Warehousing

Used for business intelligence. Pull in very large and complex datasets. ex. can query current performance vs targets. Production DB could push to data warehouse and that is where those complex querying happens.

OLTP vs OLAP

- Online Transaction Processing - online store, someone places order 20102, pulls up a row of data such as Name, Date, Address for delivery, etc.
  - "insert into ***this table***, ***this data***"
- Online Analytics Process - pulls in large number of records for complex querying. Infrequent
  - ex. net profit for EMEA and Pacific for the Digital Radio Product.

ElasticCache

in-memory cache in the cloud. improve performance of webapps by allowing you to retrieve information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases.

supported engines: Memcached and Redis. good for frequently accessed information to reduce load on production databases. Improve latency.

exam question: if you need multi-az use Redis. If you aren't concerned about redundancy use Memcached.

Different in practice.

ElastiCache manages Memcached nodes as a pool that can grow/shrink. Individual nodes are expendable.

Memcached:

- *object caching* is primary goal
- simple
- scale horizontally as you go

Redis:

- advanced data types, lists hashes, and sets.
- ***Does sorting and ranking datasets in memory (e.g., leaderboard)***
- run in multi-azs.
- pub/sub capability

If read-heavy DB is under stress use Elasticache.

- Redshift good if OLAP transactions.

RDS Lab

RDS Security Group - allow inbound from security group where web server is located over port 3306.

EC2 instance in one security group. RDS instance in another security group. Something is not connecting, open up port 3306 to the sec group the RDS group is in.

RDS: Backups, Multi-AZs, and Read Replicas

Backups:

- Automated - recover to any point in time within retention period (1-35 days). Will take full daily snapshot and transaction logs. AWS will first choose the most recent dail back up and apply transaction logs relevany to the day. Point in time recovery.
    - enabled by default
    - backup data stored in S3, free storage space equal to size of your db. RDS 10GB == 10GB worth of storage free.
    - backups taken within defined window. during window i/o may suspend, elevated latency.
    - deleted when rds instance is deleted.
- Database Snapshots
    - manually done. stored even after original db is deleted.

Restoring from snapshot makes new DNS endpoint. Completely separate DBs.

Encryption - at rest supported for all. Done with AWS KMS. Data stored at rest in underlying storage is encrypted, as are its automated backups, read replicas, and snapshots.

- today - can't encrypt existing DB instance. Must create copy and encypt copy. **exam question**

Multi-AZ - ***for disaster recovery only***. not for performance improvement (you use read replicas for that). Writes to us-east-1a are replicated to replica at us-east-1b.

You never deal with IP addresses when dealing with RDS instances, instead you deal with DNS names for endpoints so that Amazon can point the DNS at whatever instance it needs to for availbility mechanisms to work.

RDS will automatically failover to the standby so db operations resume quickly in the event of DB maintenance, db instance failover, AZ failure.

Read Replicas - ***use for scaling***. writes to db are copied out to other replicas for reading only. frequent read queries get sent to read replicas. scale out db reads to lessed load on production db. asynch replication. Can have up to 5 replicas. Replicas can be promoted to their own databases (breaks replication).

EC2 Exam Tips

- Pricing models: On-demand, Reserved, Spot, Dedicated Hosts.
- If a Spot instance is terminated by Amazon EC2 you will not be charged for a partial hour of usage. However, if you terminate the instance yourself, you will be chaged for the complete hour in which the instance ran.
- FIGHT DR MC PX - all instance types
- EBS Types:
    - SSD:
        - General Purpose SSD: balances price and performance for a wide variety of workloads
        - Provisioned IOPS - high performance, for mission critical, low latency ops.
    - Magnetic:
        - Throughput Optimizied: low cost HDD designed for frequently accessed, throughput intensive workloads
        - Cold HDD - low cost HDD designed for less frequently accessed workloads. Can't be boot volume
        - Magnetic - older. can be boot volume.
- ELBs:
    - Application
    - Network - expensive, millions of requests per second. think netflix.
    - Classic
    - Error 504 means gateway timeout. application not responding within idle timeout period. X-forwarded header sends IP address of your end user forward to the instance, otherwise it only sees private ip address of elb.
- Route53: DNS service, map domain names to S3 EC2 or ELBs.
- CLI: Least Privilege, create groups for users and apply policies to groups.
    - Secret Access Key:
    - Don't use just 1 access key to share across developers. Make one per developer. Safety to delete just one key if needed.
    - Roles allow you to not use Access Key ID's and Secret Access Keys.
    - Roles are preferred from security perspective.

Practice Quiz incorrect: HTTP 4xx worded strange. Client-side issue. Amazon ECS description Individual instances are provisioned in AZs. No encryption on the fly.

Route53

- Simple routing policy – Use for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website.
- Failover routing policy – Use when you want to configure active-passive failover.
- Geolocation routing policy – Use when you want to route traffic based on the location of your users.
- Geoproximity routing policy – Use when you want to route traffic based on the location of your resources and, optionally, shift traffic from resources in one location to resources in another.
- Latency routing policy – Use when you have resources in multiple AWS Regions and you want to route traffic to the region that provides the best latency.

- Multivalue answer routing policy – Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.
- Weighted routing policy – Use to route traffic to multiple resources in proportions that you specify.

Topics

S3

- Object based (key, value)
- Data is stored in **buckets**.
- Unlimited storage.
- S3 is a universal namespace, names must be unique globally. like an internet address.

Core Fundamentals:

- Key (name)
- Value (data)
- Version ID
- Metadata
- Subresource - used to manage bucket specific configuration
    - Bucket Policies, Access Control Lists (ACLs)
    - Cross Original Resource Sharing (CORS)
    - Transfer Acceleration

Data consistency

- **read after write** consistency for PUTS of *new* objects. can access file straight away.
- **eventual consistency** for overwrite PUTS and DELETES. can take time to propagate.

Availability: 99.99% for S3 platform. Up time, time service is available. Amazon guarantees 99.9%. **Durability**: 11 x 9s. Amount of data you can't afford to lose in a year so you want that to be close to 100%. Designed to sustain the loss of 2 facilities concurrently.

Storage Classes:

- S3 regular: durable, imeediately available, freq access
- IA: Infrequent Access. Durable. Lower fee but charged for each retrieval.
- One Zone IA: Cost is 20% less than regular IA. Don't have same level of availability.
- Reduced Redundancy Storage: designed for data that can be recreated if lost—e.g., like thumbnail, they can be recreated later.
- Glacier - very cheap but for infrequent archival storage. 3-5 hour for data access.

Charged for:

- Storage per GB
- Requests (get, put, copy)
- Storage Management Pricing - inventory, analytics, and object tags
- Data Management Pricing - data transferred out of S3

- Transfer Acceleration - uses CloudFront to accelerate.

- Max filesize transferred **via PUT request - 5 GB**.
- The total volume of data and number of objects you can store are **unlimited**.
- *Individual Amazon S3 objects* can range in size from a **minimum of 0 bytes to a maximum of 5 terabytes**

To achieve better performance, add hex hash to prefix. Key name determines which partition it will store file on. If you want to enable a user to download your private data directly from S3, you can insert a pre-signed URL into a web page before giving it to your user.

S3 Security

Allowing access to buckets.

- By default, all new buckets are PRIVATE. Hitting a url to a new file upload in a bucket will failed Access Denied. Have to grant public read access.
- Bucket Policies - bucket level. apply to everythig inside.
- Access Control Lists - object level
- S3 buckets can be configured to create access logs, which log all requests made to the S3 bucket. These logs can be written to another bucket.

Encryption

1. In Transit: SSL/TLS (HTTPS)
2. At Rest:
    1. Server Side Encryption
        - *AES-256*, S3 Managed Keys: each object with master key encryption. AWS manages keys. **SSE-S3**
        - AWS Key Management Service, Managed Keys, **SSE-KMS**
        - Server Side Encryption with Customer Provided Keys **SSE-C**
    2. Client Side Encryption - you encrypt yourself before it hits the cloud.

Enforcing Encryption

- PUT request each time a file is uploaded

```
PUT /myFile HTTP/1.1
...
Expect: 100-continue
```

Expect: 100-continue - don't send the body until acknowledged by S3 that content headers are set a certain way.

- If file is to be encrypted at upload time, use `x-amz-server-side-encryption` will be included in the request header.
- two options available:
    - `x-amz-server-side-encryption`: AES256 (SSE-S3 - S3 managed keys)
    - `x-amz-server-side-encryption` ams:kms (SSE-KMS - KMS managed keys)

- When this parameter is included in the header of the PUT request, it tells S3 to encrypt the object at the time of the upload, using the specified encryption method.
- You can enforce the use of Server Side Encryption by using a Bucket Policy which denies any S3 PUT request which doesn't include the `x-amz-server-side-encryption` parameter in the request header.

This tells S3 to encrypt the file using SSE-S3 at the time of upload:

```
PUT /myFile HTTP/1.1
...
Expect: 100-continue
x-amz-server-side-encryption: AES256
```

CORS allowing code in one bucket or resource to access code in another. by default they can't access other resources in other buckets because they are PRIVATE by default upon creation.

CloudFront

CDN service

- edge location - where content is cached. separate to an AWS Region/AZ.
  - READ and WRITE
  - Used by *S3 for Transfer Acceleration* - optimized network path for s3 transfer path.
- origin - could be an s3 bucket, ec2 instance, etc. where files originate.
- distribution - name given the CDN, which consists of edge locations.
  - Web Distribution - HTTP/HTTPS, used for websites.
  - RTMP Distribution - used for media streaming.

delivery of content using global network of edge locations/ requests for your content are autoamtcially routed to the nearst edge location, so content is delivered with the best possible performance.

CloudFront + S3 Transfer Acceleration

S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your end users and an S3 bucket.

Transfer Acceleration takes advantage of Amazon CloudFront's globally distributed edge locations. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path.

Signed URLs or Signed Cookies are a way to restrict users access via URLs. WAF helps protect you at the application layer

S3 Performance Optimizations

GET intensive workloads you should use CloudFront.

```
optimize if you get > 100 PUT/LIST/DELETE or > 300 GET requests per
second.
```

Mixed Request Type Workloads - key names can impact performance. avoid sequential key names for your S3 objects. Instead, add a random prefix like a hex hash to the key name to prevent multiple objects from being stored on the same partition. Reduces likelihood of I/O contention.

> 2018 July, new increase in performance negates the best practice of using sequential / random hash prefixes in keynames.

Serverless

Lambda

Event-driven compute service. Compute service to run your ode in response to HTTP requests using AWS API Gateway. Supports: NodeJS, Python, Java, C#, Go Price: 1st 1 million requests per month are free. $0.20 per 1 million requests after that.

- Calculated on duration of function execution Scales out (not up) automatically 1 event = 1 function 1 event can trigger any number of events.

Lambda Optimizations

1. Separate the Lambda handler (entry point) from your core logic.

```
exports.myHandler = function(event, context, callback) {
        var foo = event.foo;
        var bar = event.bar;
        var result = MyLambdaFunction (foo, bar);

        callback(null, result);
}

function MyLambdaFunction (foo, bar) {
        // MyLambdaFunction logic here
}
```

1. Take advantage of Execution Context reuse to improve the performance of your function.
2. Use AWS Lambda Environment Variables to pass operational parameters to your function
3. To control the dependencies in your function's deployment, package all dependencies with your deployment package.
4. Minimize your deployment package size to its runtime necessities.
5. Reduce the time it takes Lambda to unpack deployment packages
6. Minimize the complexity of your dependencies (prefer simpler frameworks.)

7. Avoid using recursive code. Could lead to unintended function volume and escalated costs. To prevent on accident, set *function concurrent execution kimit* to 0 immediately to throttle all invocations to the function.

Lambda Encryption

To encrypt 1 MB, you need to use the Encryption SDK and pack the encrypted file with the lambda function

AWS Lambda environment variables have a maximum size of a few KB. Additionally, the direct "Encrypt" API of KMS also has a few KB limit.

Lambda Triggers

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 Versioning

- $LATEST is the latest version of a function
- Can have multiple versions. You can publish one or more versions of your Lambda function.
- you can work with different variations of your Lambda function in your development workflow such as dev/beta/prod.
- Each Lambda function version has a unique ARN - after you publish a version it is ***immutable***
    - Qualified ARN - the function ARN with the version suffix `arn:aws:lambda:aws-region:acct-id:function:hellowworld:$LATEST`
    - Unqualified ARN - the function ARN without version suffix: `arn:aws:lambda:aws-region:acct-id:function:hellowworld`
- Can split traffic using aliases to different versions
    - cannot split traffic with $LATEST, instead create version instead.

Aliases

- with a function version $LATEST, you can publish a version 1 of it.
    - by creating an alias named PROD that points to version 1, you can now use the PROD alias to invoke version 1 of the Lambda function.
    - then, update the code (the $LATEST version) with all of your improvements, then publish another stable and improve version 2. Promote version 2 to production by remapping the PROD alias so that it points to version 2.

API Gateway

API is like a waiter relaying orders from customers to chefs and bringing results from chefs to customers.

Fully managed service for APIs at scale. Expose HTTPS endpoints to define a RESTful API Serverlessly connect to service like Lambda and Dynamo Send each API endpoint to a different target Scale effortlessly

Usage & Billing Track and control usage by API key. Limit usage with *Usage Plans with API Keys* Throttle API requests Connect to Cloudwatch Versioning of API

Import APIs

- Import custom on-prem API using Swagger Importer tool

1. Define an API (container)
2. Define resources and nested resources (url paths)
    1. select supported HTTP methods
    2. set seuciruty
    3. choose target
    4. set rquest and response transformations
3. deploy api to a stage

API caching for endpoint response

- Caching lets you reduce the number of calls made to your endpoint and also improve the latency.
    - ex. McDonald's has most commonly ordered burgers ready to go to reduce latency.

Same-origin policy Cross-origin resource sharing - can relax same-origin policy restrictions

Authorization:

> An Amazon API Gateway **Lambda authorizer** is a Lambda function that you provide to control access to your API methods. A Lambda authorizer uses bearer token authentication strategies, such as OAuth or SAML. It can also use information described by headers, paths, query strings, stage variables, or context variables request parameters.

- Cognito User Pools
- IAM user permissions

API Gateway Deployment

> In a canary release deployment, total API traffic is separated at random into a production release and a canary release with a pre-configured ratio. Typically, the canary release receives a small percentage of API traffic and the production release takes up the rest. The updated API features are only visible to API traffic through the canary. You can adjust the canary traffic percentage to optimize test coverage or performance

Stage variables are name-value pairs that you can define as configuration attributes associated with a deployment stage of an API. They act like environment variables and can be used in your API setup and mapping templates. With **deployment stages** in API Gateway, you can manage multiple release stages for each API, such as alpha, beta, and production. Using stage variables you can configure an API deployment stage to interact with different backend endpoints.

Step Functions

Visualize serverless application Automatically triggers and tracks each step logs the state of each step so if something goes wrong you can track what went wrong and where.

AWS X-Ray

Service that collects data about requests that your application serves, and provides tools you can use to view filter, and gain insights into that data to identify issues and opportunities for optimization.

```
<App with X-Ray SDK installed> sends JSON to <X-Ray Daemon> listening on UDP,
send JSON to <X-RAY API>, visualizes in <X-Ray Console>
```

SDK:

- Client handlers to instrument AWS SDK clients that your app 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 Java Go Node Python Ruby .NET -> anything supported by Lambda

> The X-Ray SDK applies a sampling algorithm to efficiently trace and provide a representative sample of the requests that your application serves. This can be utilised post the data is being successfully being sent to X-Ray and in no way helps us in determining the cause of failure to send the data to X-Ray.

IAM rules: Create an IAM role with write permissions and assign it to the resources running your application. AWSXRayDaemonWriteAccess includes permission to upload traces.

If your Lambda function runs on a schedule, or is invoked by a service that is not instrumented, you can configure Lambda to sample and record invocations with active tracing.

CI / CD

Multiple developers contributing to the same application. Committing code to the same repository.

Code repository integrated with build management system. Code changes trigger an automated build.

Test framework helps prevent breaks from commits. Focuses on small code changes to the main repository.

CD = development practice where merged changes are automatically build, tested,

Dev commit -> repo -> build management system -> test framework -> deploy packaged app

CD deploys new code automatically foolowing successful testing.

CodePipeline - automate and orchestrate all of the pipeline activities.

AWS CodePipeline copies files or changes that will be worked on by the actions and stages in the pipeline to the Amazon S3 bucket. These objects are referred to as artifacts, and might be the

source for an action (input artifacts) or the output of an action (output artifacts).

if an automated test fails, the pipeline stops immediately.

CodeCommit (code repo) -> CodeBuild (code management system) -> CodeDeploy (deploy pakcaged app)

- [ ] read the whitepaper

CI is about integrating merging the code changes frequently at least once per day

CD all about automaticng build test and deployment. CD fully automates

CodeCommit

- based on Git, enables collaboration, just like any other git hosting

CodeDeploy

Fully managed automated deployment service and can be used as part of CD process.

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

```
## Lambda

version: future use only
resources: name and props of lambda function to deplouy
    - myLambdaFunction
        - Type
        - Properties
            - Name
            - Alias
hooks: run at set points  in deployment lifecycle
    BeforeTrafficAllowed
    AfterTrafficAllowed
```

Before and After traffic allowed is good to test that the deployment is ready to accept traffic and that it is working as expected once traffic shifted to it.

BeforeBlockTraffic - run tasks on instances before they are deregistered from a load balancer BlockTraffic - deregister instances from a load balancer AfterBlockTraffic - run tasks on instances after they are deregistered from a load balancer.

ValidateService - the last deployment life cycle event used to verify the deployment completed successfully.

```
## EC2

version: 0.0
os: linux
```

```
files:
    source: file.txt
    location: where/it/is
```

In Place

- application is stopped on each instance and the latest fevision installed.
- The instance will be out of service / cpaacity is reduced
- ELB can stop sending requests for instances that are down
- Roling Update

Blue/Green

- new instances are provisioned and the latest revision is installed on the new instances. Blue represents the active, Green is the NEW release.
- The new instances are registered with an ELB traffic is then routed to the new instances and the oroginal intances are eventually terminated.

ADVANTAGE: new instances can be created ahead of time, release means you just switch ELB routing to new servers. Switching back to original env is faster and more reliable and is just a acase of routing the traffic back to the original servers.

Deployment Group - set of EC2 instances that accept a deployment

CodePipeline

fully managed CI/CD service

Orchestrate the build, test, and deployment of app on every change.

Can be configured to auto trigger your pipeline on changes.

Docker commands for exam: **BUILD, TAG, PUSH** to an ECR repository.

```
docker build –t mydockerrepo .
docker tag mydockerrepo:latest url.com/mydockerrepo:latest
docker push url/mydockerrepo:latest
```

Use `buildspec.yml` to define the build commands and settings used by CodeBuild to run your build.

Encrypt output of Codebuild by specifying KMS key to use.

You can override/insert build commands directly using an editor inline or use a `buildspec.yml` predefined.

Check CodeBuild console for logs or see in CloudWwatch.

CloudFormation

- manage configure and provision infrastructure as code.
- YAML or JSON
- idempotent instances and configurations
- free to use except what underlying you use
- rollback/delete easily
- manage updates and dependencies
- the order in which resources are created must not be specified.

> The stack name is an identifier that helps you find a particular stack from a list of stacks. A stack name can contain only alphanumeric characters (case-sensitive) and hyphens. It must start with an alphabetic character and can't be longer than 128 characters. ***Must be unique within a region***

- Which pseudo parameter can you use to make your CloudFormation independent of the accounts they're running under? `AWS::AccountId`. AWS::AccountId returns the AWS account ID of the account in which the stack is being created Read more: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/pseudo-parameter-reference.html
- The `Fn::GetAtt` intrinsic function returns the value of an attribute from a resource in the template. We can use the above function to get the regions ID attribute of the required EC2 instance by passing region ID as the attributeName and EC2 instance ID as logicalNameOfResource.
- The intrinsic `function Ref` returns the value of the specified parameter or resource.When you specify a parameter's logical name, it returns the value of the parameter.When you specify a resource's logical name, it returns a value that you can typically use to refer to that resource, such as a physical ID.

Template

- upload template using S3, CF reads it and calls API calls to make it happen.
- **Parameters**: custom values, environment type prod or test.
- **Conditions**: provision resources based on environment.
  - cannot be applied to Parameters
- **Mappings**: create custom mappins like Region:AMI.
- **Tranformation**: reference code yaml/json in S3 and run it. Code re-use here.
- **Resources**: resources you are deploying. **MANDATORY**
- **Outputs**: user-defined

Only the exported CloudFormation, Export names must be unique within a region for each AWS account.

- The optional Transform section specifies one or more macros that AWS CloudFormation uses to process your template. The Transform section builds on the simple, declarative language of AWS CloudFormation with a powerful macro system. INTRINSIC FUNCTION.
-

CloudFormation and SAM

Serverless Application Model (SAM) CloudFormation template and it starts with:Transform: 'AWS::Serverless-2016-10-31' == it's a SAM template

- `AWS::Serverless::Function` creates a Lambda function, IAM execution role, and event source mappings which trigger the function.
- `AWS::Serverless::Api` creates a collection of Amazon API Gateway resources and methods that can be invoked through HTTPS endpoints.
- `AWS::Serverless::SimpleTable` resource creates a DynamoDB table with a single attribute primary key.

The **AWS::Serverless transform** is specifically used for transforming an entire template written in the AWS Serverless Application Model (AWS SAM) syntax into a compliant AWS CloudFormation template. Read more: https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/transform-aws-serverless.html

SQS the first AWS service

"Pull-based" Message queue - enables web app components to queue messages for other components to read from. ***No limit on max number of messages stored in an SQS queue.***

- **FIFO queues** - msg delivered only once, order guaranteed, good for banking with strict ordering.
- **Standard Queues** (default) - best effort ordering, message delivered at least once.

Dead Letter Queues Amazon SQS supports dead-letter queues, which other queues (source queues) can target for messages that can't be processed (consumed) successfully. Dead-letter queues are useful for debugging your application or messaging system because they let you isolate problematic messages to determine why their processing doesn't succeed.

- Short poll - return immediately if no messages in queue.
- Long poll - periodic polling of queue. only return response when a message is in the queue or the timeout is reached.
    - MAX LONG POLL == 20 seconds

> Once a message is received by one of the distributed consumers, the message cannot be read again until the visibility timeout of that particular message expires. If the message is not processed within the visibility timeout, there are chances that the message will be received again by the distributed consumers leading to duplication in processing of message. By setting a sufficient visibility timeout for the specific message, we can ensure that the same message is not received again by the distributed consumers as the original consumer will delete the message once it has processed. The visibility timeout of the few problematic messages can be altered using ChangeMessageVisibility.

VisibilityTimeout *ChangeVisibility*

- default is 30 seconds - increase is task takes > 30 seconds to complete.
- Max timeout is 12 hours.
- Max retention period - 14 days
- Max size 256KB

SNS

- scalable and highly available notification service.
- Push-based, no polling.
- $0.50 per 1 million
- pub/sub - users subscribe
- notifications delivered to clients using push
- can be customized by protocol type

SQS + SNS to fan out messages distributed to many queues.

SES

Email only service - for marketing teams. Incoming and outgoing emails.

- status updates, shipping notifications, order status
-

Elastic Beanstalk

- you upload code, EB handles deployment and everything else.
- Scales your app up and down
- You maintain admin control
- Supports Java, PHP, Ruby, NodeJS
- App Servers: Tomcat, Passenger, Puma, IIS

Updating:

- All at once - updates take all instances down while updates get applied to all intances. You need to roll all back. Not great for prod.
- Rolling - deploys new version in batches. Good for high performance systems.
- Rolling with batch - launches an additional batch of instances. can't afford downtime - maintains full capacity.
- immutable - deploys new version to fresh group of instances in their own new autoscaling group. preferred for mission critical. Roll backs just entail killing the unwanted instances.

Speed up deployments by bundling dependencies with the code during the build output phase.

Configuring

YAML or JSON file has configs. Needs to end in `myconfig.config` and in `.ebextensions` folder in top-level directory of app.

Coupling with RDS

- In dev OK, not in prod. Ties lifecycle together.
- Any resources created as part of your .ebextensions is part of your CloudFormation template and will get deleted if the environment is terminated. Resources that need to persist environments deletions must be created externally

Kinesis

Streaming data

1. Streams - data producers feed KStreams. Data stored in shards, data is passed to consumer (EC2). The COnsumers then pass that to other AWS services. SHARDS
    1. To reduce overhead and increase throughput, the application produce records in batches
    2. `ProvisionedThroughputException`: use batch messages and ExponentialBackoff to reoslve and keep cost down.
    3. Kinesis Data Streams segregates the data records belonging to a stream into multiple shards. It uses the partition key that is associated with each data record to determine which shard a given data record belongs to. Partition keys are Unicode strings with a maximum length limit of 256 bytes.A stream is composed of one or more shards, each of which provides a fixed unit of capacity. Each shard can support up to 5 transactions per second for reads, up to a maximum total data read rate of 2 MB per second and up to 1,000 records per second for writes, up to a maximum total data write rate of 1 MB per second (including partition keys). The data capacity of your stream is a function of the number of shards that you specify for the stream. The total capacity of the stream is the sum of the capacities of its shards.
2. Firehose - analyzing data. real-time analytics for BI tools.easiest way to load streaming data into data stores and analytics tools.
    1. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk.
3. Analytics - SQL type queries off collected data