**AWS Certified Solutions Architect – Associate  
  
Recommended Knowledge and Experience**

* Hands-on experience using compute, networking, storage, and database AWS services
* Hands-on experience with AWS deployment and management services
* Ability to identify and define technical requirements for an AWS-based application
* Ability to identify which AWS services meet a given technical requirement
* Knowledge of recommended best practices for building secure and reliable applications on the AWS platform
* An understanding of the basic architectural principles of building on the AWS Cloud
* An understanding of the AWS global infrastructure
* An understanding of network technologies as they relate to AWS
* An understanding of security features and tools that AWS provides and how they relate to traditional services

Security

|  |  |
| --- | --- |
| API use | * Better to use AWS roles within AWS than access/secret keys |
| Users | * Least Privilege - Always give users the minimum amount of access required |
| Groups | * Users with shared permissions * Assign policies to groups * Assign users to groups, they will automatically inherit permissions |
| Roles | * Assigned to AWS resources * Users and applications assume roles * Grant permissions to entities: ex: user, app, service (ex: Ec2, Lambda) * Prevents having to use access/secret key * Controlled by policies * Can be attached/detached to running Ec2 instances without having to stop them |
| Policies | * JSON document defining permissions * Attach to user/group/role * Includes allow/deny, action and resources |
| Access key/secret access key | * Can only view secret access key once * If lose, must regenerate * Use if connecting from outside AWS |
| Policy Generator | * Creates JSON * Type: SQS, SNS, S3, VPC Endpoint, IAM Policy * Statements: Allow/Deny, Principal’s ARN, AWS Service, Actions, ARN of resource * Principal can be \* * Resource can be arn or arn/\* for all within |
| ARN | * Amazon Resource Name * Ex: arn:aws:iam::accountId:user/name |
| Encryption types | * In transit (SSL/TLS) * At rest (keys) * Client side encryption |
| CloudHSM | * Hardware security module (for keys) |

KMS

|  |  |
| --- | --- |
| General | * Key Management Service * Create/control data encryption keys * Multi-tenant. * Best practice; user who manages keys can’t encrypt/decrypt * Can encrypt up to 4 kilobytes (4096 bytes) of arbitrary data such as an RSA key, a database password, or other sensitive information |
| Keys | * Encryption keys are regional. Must decrypt in region encrypted in * Customer master key – generated by Amazon or AWS provided. Cannot be exported. |
| API Calls | * Aws kms encrypt * Aws kms decrypt * Aws kms re-encrypt – re-encrypt with new master key and delete original encrypted file * Aws kms enable-key-rotation – rotates the key yearly |
| Envelope encryption | * Customer Master Key used to decrypt the data key (envelope key) * Envelope key is used to decrypt the data |
| Systems Manager Parameter Store | * Store confidential information * Create key/value parameter as string/list * Encrypt with KMS * Available from EC2, CloudFormation, Lambda |

**IAM - Identity and Access Management**

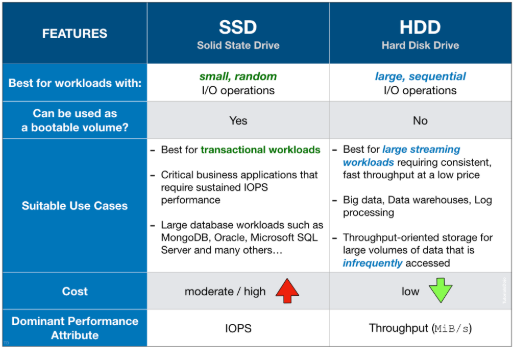
|  |  |
| --- | --- |
| Overview | * Manage users and their access to AWS Console * IAM is global across all regions and consists of users, groups, and roles. Roles can be applied to both users and AWS services (EC2 instances, Lambda, etc). * Pass role to EC2 via instance profile * Supports PCI DSS compliance (Payment Card Industry) * Root account has full admin access upon account creation * Root account is email address that you used to register your account * Not region specific, can be shared between all regions * By default when you create a user, they have NO permissions to do anything * Recommended that root account is not used for login, and should be secured with Multi-factor Authentication (MFA) * Can create Access Keys/ Secret Access Keys to allow IAM users (or service accounts) to be used with AWS CLI or API calls * Policies can be applied to users, groups and roles |
| Web Identity Federation | * Authenticate with Amazon/Facebook/Google * Trade web token/auth code for temporary AWS credentials |
| Cognito | * Identity Broker * Provides Web Identity Federation * Can also use SAML * Synchronizes user data across devices using SNS * Good for mobile apps * No custom code * User pools - user directories to manage sign up/sign in. Generate JSON Web Tokens (JWT) * Identity pools – unique identity/temporary credentials * To configure; needs user pool, app client and domain |
| Policy types | * Managed Policies – created by AWS. * Customer Managed Policies – only within your account. More granular than built in. Use for pre-defined common use cases * Inline Policies – embedded in user/group/role. Only use if need to ensure will not be reused. |
| Policy Example | {  "Version":"2012-10-17",  "Statement":[  {  "Effect":"Allow",  "Action":[  "s3:PutObject",  "s3:GetObject",  "s3:GetObjectVersion",  "s3:DeleteObject",  "s3:DeleteObjectVersion"  ],  "Resource":"arn:aws:s3:::examplebucket/${aws:username}/\*"  }  ]  }  **Notes**: You can attach the following policy to user Alice to allow her specific Amazon S3 permissions on the examplebucket/Alice folder |
| AssumeRoleWithWebIdentity | * From STS (Security Token Service) * API to return temporary security creds * Key for creds: Includes AssumedRoleUser ARN and AssumedRoleID (not IAM role) * Creds include session token, access key, secret access key * Temporary creds default to an hour * API call STS assume-role-with-web-identity - enable a user authenticated by Amazon/Facebook/Google to access your web application hosted in AWS |
| Cross account access | * Same creds to use multiple accounts in console. Don’t need to re-login * Can grant specify policies from one account to another |

VPC – Virtual Private Cloud

|  |  |
| --- | --- |
| Overview | * A virtual private cloud (VPC) is a virtual network dedicated to your AWS account. * Default Security Group, Access control List, and Route Table are created automatically |
| Egress-Only Internet Gateway | * Horizontally scaled, redundant, and highly available VPC component that allows outbound communication over IPv6 from instances in your VPC to the Internet, and prevents the Internet from initiating an IPv6 connection with your instances. |
| VPN | * **Virtual private gateway** - is the VPN concentrator on the Amazon side of the Site-to-Site VPN connection * **Customer gateway device** - is a physical device or software application on your side of the Site-to-Site VPN connection |

EC2 - Elastic Compute Cloud

|  |  |
| --- | --- |
| Overview | * Elastic Compute Cloud provides re-sizable compute capacity in the cloud (Virtual Machines). * Reduces the time required to obtain and boot new server instances to minutes allowing you to quickly scale capacity, both up and down, as your computing requirements change. * EC2 meta data service - helpful when you're writing scripts to run from your instance. For example, you can access the local IP address of your instance from instance metadata |
| Pricing Models | * **On Demand:**   + Pay fixed rate by the hour(or by the second) with no commitment   + Users that want the low cost and flexibility of EC2   + Apps with short term, spiky or unpredictable workloads that cannot be interrupted   + Apps being developed or tested on EC2 for the first time * **Reserved Instances:**   + Provide capacity reservation and offer significant discount on the hourly charge for an instance (1-3 year terms)   + Applications have steady state, or predictable usage   + Apps that require reserved capacity   + Users able to make upfront payments to reduce their total computing costs even further. * **Spot:**   + Bid whatever price you want for instance capacity by the hour   + When your bid price is greater than or equal to the spot price, your instance will boot   + When the spot price is greater than your bid price, your instance will terminate with an hours’ notice.   + Applications have flexible start and end times   + Apps that are only feasible at very low compute prices   + Users with urgent computing needs for large amounts of additional capacity   + If the spot instance is terminated by Amazon EC2, you will not be changed for a partial hour of usage   + If you terminate the instance yourself you WILL be charged for any partial hours of usage. * **Dedicated Hosts:**   + Physical EC2 server dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses. Usefull for compliance requirements when you can’t use multi-tenant virtualization * **Scheduled Reserved Instances**:   + Scheduled instances allow you to reserve capacity on a recurring basis with a daily, weekly, or monthly schedule over the course of a one-year term |
| EBS Storage | * Stored in specific availability zone * Automatically replicated within zone * EBS backed instances can be stopped, you will NOT lose any data * EBS volumes can be detached and reattached to other EC2 instances 5 Types of available EBS volumes can be provisioned and attached to an EC2 instance:   + General Purpose SSD (GP2):     - General Purpose up to 10K IOPS     - 99.999% availability     - Ratio of 3 IOPS per GB with up to 10K IOPS and ability to burst     - Up to 3K IOPS for short periods for volumes under 1GB   + Provisioned IOPS SSD (I01)     - Designed for I/O intensive applications such as large relational or No-SQL DBs.     - Use if need more than 10K IOPS   + Magnetic (Throughput Optimized HDD)     - Low Cost HDD volume for frequently accessed, throughput intensive workloads   + Magnetic (Cold HDD)     - Lowest cost HDD volume for less frequently accessed workloads     - Handle large, sequential I/O operations   + Magnetic (Standard)     - Lowest cost per GB     - Ideal for workloads where data is accessed infrequently and apps where the lowest cost storage is important.     - Ideal for fileservers     - Previous generation, can be bootable |
| EBS Encryption | * Creating volume from encrypted snapshot is encrypted * Creating volume from unencrypted snapshot is unencrypted * If copy an unencrypted snapshot to create a new snapshot, can encrypt it when creating the copy. Then can make AMI of it to have encrypted root device. * You can encrypt root device volume by first taking a copy of that snapshot, and then create an encrypted snaphost. You can then make an AMI of this snapshot to have an encrypted root device volume. |
| Placement Groups | * ***Cluster*** – packs instances close together inside an Availability Zone. This strategy enables workloads to achieve the low-latency network performance necessary for tightly-coupled node-to-node communication that is typical of HPC applications. * ***Partition*** – spreads your instances across logical partitions such that groups of instances in one partition do not share the underlying hardware with groups of instances in different partitions. Ex: Hadoop, Cassandra, and Kafka. * ***Spread*** – strictly places a small group of instances across distinct underlying hardware to reduce correlated failures. It can span multiple Availability Zones in the same Region. You can have a maximum of seven running instances per Availability Zone per group |
| Tenancy | * ***default*** - Your instance runs on shared hardware. * ***dedicated*** - Your instance runs on single-tenant hardware. * ***host*** - Your instance runs on a Dedicated Host, which is an isolated server with configurations that you can control  It’s not possible to change the tenancy of an instance from default to dedicated or host after you've launched it. It’s not possible to change the tenancy of an instance from dedicated or host to default after you've launched it. |

EBS Types  


Storage Gateway

|  |  |
| --- | --- |
| Overview | * Hybrid cloud storage service that connects your existing on-premises environments with the AWS Cloud |
| Types | * ***File Gateway*** – store files as objects in Amazon S3 using the industry-standard NFS and SMB file protocols * ***Tape Gateway*** - industry-standard iSCSI-based virtual tape library (VTL) * ***Volume Gateway*** - applications block storage volumes using the iSCSI protocol. Data written to these volumes can be asynchronously backed, and stored in the cloud as EBS snapshots |

ELB – Elastic Load Balancers

|  |  |
| --- | --- |
| Overview | * 3 Types of Elastic load balancers |
| Application Load Balancer | * Layer 7 (app layer) * Can make decisions based on content, advanced routing * HTTP/HTTPS traffic * Session Affinity – Bind user session to specific EC2 |
| Network Load Balancer | * Layer 4 * TCP traffic * Extreme performance/low latency * Assumes static IP addresses * Most expensive |
| Classic Load Balancer | * Legacy; no longer recommend [but on exam] * Can use layer 4 or 7 (X-Forwarded-For and sticky sessions) |
| X-Forwarded-For | * Original (end user public IP) * Load balancer converts public IPv4 address to private IP |
| HTTP Codes | * 200 - The request has succeeded * 3xx - Redirection * 4xx - Client Error (404 not found) * 5xx - Server Error * 504 – Gateway Timeout, means application not responding within idle timeout |

RDS – Relational Database Service

|  |  |
| --- | --- |
| Overview | * Relational Database Service * OLTP – Online Transaction Process, fast writes * You can stop an instance for up to 7 days at a time, if Single AZ |
| Database Types | * SQL Server, Oracle, MySQL, PostGres, Aurora, MariaDB |
| Aurora | * Compatible with MySQL, PostGres * Endpoints   + ***Cluster endpoint*** (or *writer endpoint*) for an Aurora DB cluster connects to the current primary DB instance for that DB cluster. This endpoint is the only one that can perform write operations   + ***Reader endpoint*** for an Aurora DB cluster provides load-balancing support for read-only connections to the DB cluster. Use the reader endpoint for read operations, such as queries   + ***Custom endpoint*** for an Aurora cluster represents a set of DB instances that you choose. When you connect to the endpoint, Aurora performs load balancing   + ***Instance endpoint*** connects to a specific DB instance within an Aurora cluster. |
| Security | * If EC2 and RDS in different security groups, need to open port 3306 (example) |
| Backups | * Automated backup – 1-35 days. Daily snapshot and transaction logs throughout day. Enabled by default. Get free storage space matching RDS disk space. * Database snapshots – manual. Kept even after delete RDS instance * When restore, get new RDS instance with new DNS endpoint. Can restore to any point in time. |
| Multi AZ | * For disaster recovery only * Synchronously replicated to standby in another availability zone * Automatic failover. Name stays same even though IP changes |
| Read replica | * For performance/scaling, NOT disaster recovery * Up to 5 read replicas * Requires automatic backups to be enabled * Can have read replicas of read replicas. Can increase latency. * Can be in different availability zone or region * Not available for SQL Server or Oracle * Read replica can have Multi-AZ * Can “clone” to be own db and turn off replication * Can encrypt even if source is not encrypted |

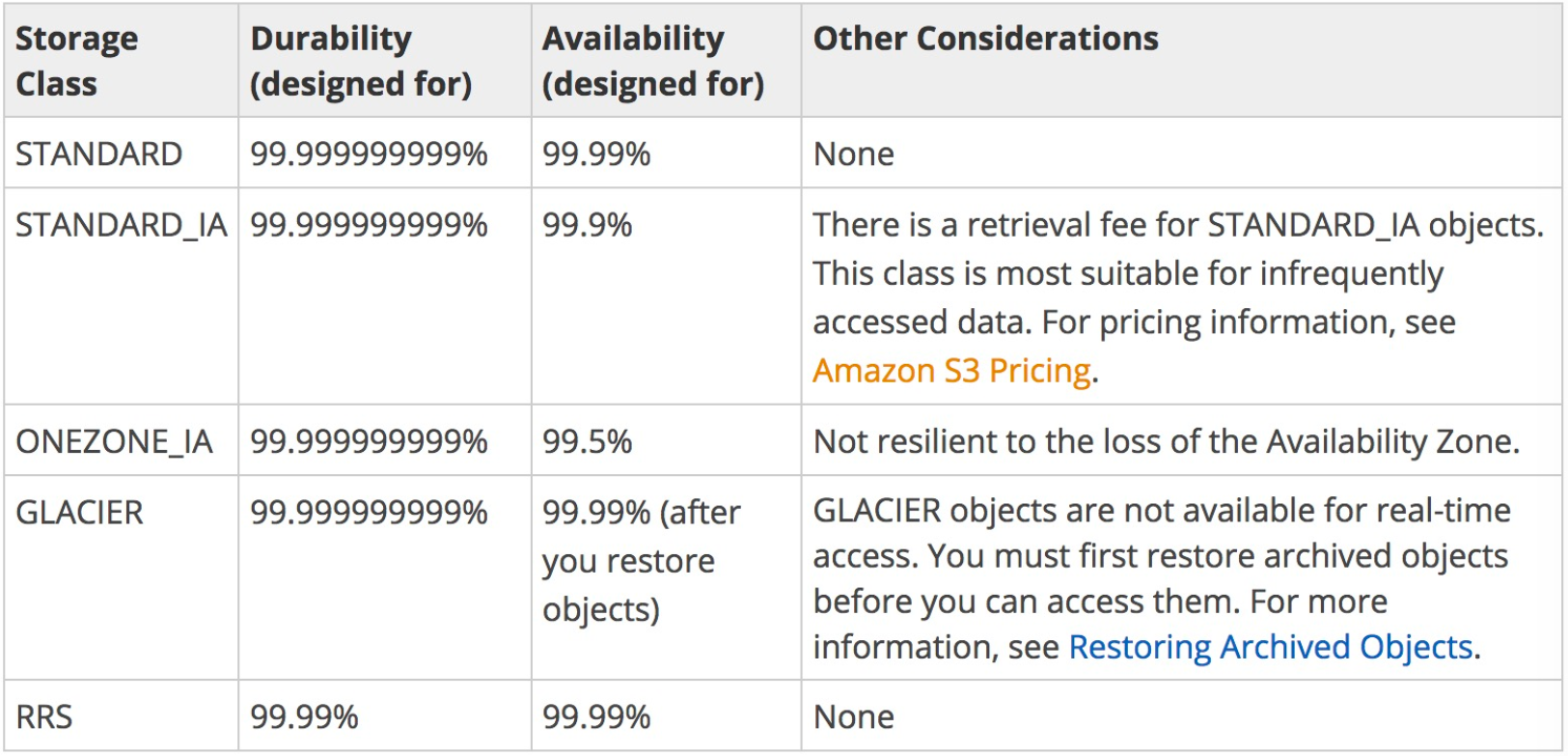
Elasticache

|  |  |
| --- | --- |
| Overview | * In memory cache in cloud, used to cache frequently accessed information |
| Types | * Supports Memcached (memory object caching with no persistence) and Redis (in memory key- value store, works with Multi-AZ and maser/slave replication. Manages like RDS) * Use Memcached for: object caching, simple, large cache nodes with threads, scale horizontally (scale out) * Use Redis for: advanced data types(lists/hashes) , sorted/ranking (ex: leaderboard), persistence, failover(Multi AZ), pub/sub [unless data warehousing, then RedShift] |
| Caching Strategy | * Lazy Loading – loads only when needed. Returns null if not found. Data can be stale if changed after placed in cache and before TTL expires * Write through – updates cache when data changes. Write penalty because updates even if not needed and updates even if not read. (no stale data) |

S3

|  |  |
| --- | --- |
| Overview | * Object storage, key/value pairs * Not for database, OS * Unlimited storage * High availability/disaster recovery built in * 0 bytes to 5TB. Can upload up to 5GB with PUT. Use multi-part upload API for 100MB+ * Files stored in buckets or folders within buckets, no nested buckets * Bucket names must be globally unique * Basic charges for storage, data transfer and requests * Buckets partitioned by key name * Presigned URL:  Object owner can optionally share objects with others by creating a presigned URL, using their own security credentials, to grant time-limited permission to download the objects * Default URL: <https://s3-eu-west-1.amazon.com/celfocusbucket> * Website URL: <http://celfocusbucket.s3-website-ap-northeast-1.amazonaws.com> * S3 Inventory - Amazon S3 inventory is one of the tools Amazon S3 provides to help manage your storage. You can use it to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs. * S3 Select - enables applications to retrieve only a subset of data from an object by using simple SQL expressions |
| Data consistency | * Read after write – available right away. For new objects(PUTS) * Eventual consistency – can take time to propagate. For updating objects (PUTS)/deleting objects (DELETES) |
| Storage Tiers | * S3 – 99.99% availability, 11 9’s durability. Stored across multiple facilities. Designed to sustain loss of two data centers * S3 - IA (Infrequently accessed) – Lower fee, but charged every time access. Min 30 days * S3 – One Zone IA – 99.5% availability. Only in one availability zone. Min 30 days * Reduced Redundancy Storage – 99.99% durability. For data that can be recreated if lost. Not recommended for use. * Glacier – for archiving. Very cheap. For data infrequently accessed. Several hours to retrieve data. Min 90 days * S3 – Intelligent Tiering – 2 tiers. Automatically moves data to most cost-effective tier based on how frequently access. New option. * Set using x-amz-storage-class header |
| Security | * Buckets private by default * Bucket policies at bucket level * Access control lists – at object level * Can log all access to bucket in another bucket |
| Encryption at rest | * AES-256/SSE-S3 – S3 Managed keys - each object gets own key * AES-KMS/SSE-KMS – Key Management Service – additional key to encrypt data’s encryption key. Get audit trail of when key used. * SSE-C – Customer provided keys (set your own encryption keys) * Enable managed keys when creating a bucket. Alternatively, create Bucket policy to deny all PUTS without x-amz-server-side-encryption header * Client Side Encryption - data is encrypted on the client side and is encrypted at rest but to download the file again, the client must decrypt it |
| CORS | * Cross Origin Resource Sharing, enforced by client * Avoids same origin policy problem (which prevents XSS) * Allow a resource in one bucket to access one in another bucket * Configure as XML on bucket being referenced from elsewhere and specify bucket that can access * Gives 403 (forbidden) when not enabled |

S3 – Tiers Matrix



CloudFront

|  |  |
| --- | --- |
| CDN | * Content Delivery Network * Serve static content from closer location around world * Has Viewer protocol policy * User signed URL or signed Cookie for paid content |
| Edge location | * Where content is cached and can be written * More edge locations than availability zones * Not read only, can PUT an object to S3 |
| Origin | * Where content starts – S3, EC2, ELB, Route 53 |
| Distribution | * Web distributions – websites * RTMP (real time messaging protocol) – media streaming |
| S3 Transfer Acceleration | * Uses edge locations to route to S3, for intensive workloads |
| Caching | * Stored for TTL (time to live) * Get charged to clear cache object before TTL |
| Using CloudFront | * URL cloudfront.net * Takes about 15 minutes to propagate initially * Slow first time because caching at edge location |

Lambdas

|  |  |
| --- | --- |
| Overview | * Serverless – Cheap Compute service * Scales continuously with more functions (can’t automatically add memory) * Upload code to create Lambda function * Use cases: event driven compute service, compute service in response to HTTP requests * Max timeout changed 5 minutes to 15 minutes * Max 50MB compressed/250MB uncompressed * Max environment variables – 4KB * Lambdas can call other lambdas * Lambda@Edge is a feature of [Amazon CloudFront](https://aws.amazon.com/cloudfront/) that lets you run code closer to users of your application, which improves performance and reduces latency |
| Languages | * Node.JS, Java, Python, C#, Go, Ruby, PowerShell |
| Debugging | * Debug with XRay |
| Step Functions | * Type of application integration * Graphical console to arrange/visualize components * Automatically triggers next step * Types of steps – sequential, branching, parallel * Coded in JSON using Amazon State Language * Generates Lambda Functions * Logs each step * Differs from SWF (simple workflow service) in that can only have one state definition vs multiple deciders. Also deciders can’t be implemented as lambdas. Use step functions as first choice and SWF if doesn’t meet needs |
| Concurrent Executions | * There is a limit of 1000 concurrent execution per second, can be requested an increase by AWS support * Reserved concurrency guarantees an always available to a critical function |
| VPC Access | * Lambda use VPC information to setup ENI using an IP from Private Subnet * Security Group allow function to access resources |
| Temp Directory | * directory /tmp - This is 512MB of temporary space you can use for your Lambda functions |

API Gateway

|  |  |
| --- | --- |
| Overview | * Managed service * Publish/maintain/monitor/secure APIs * “Front door” for APIs in EC2/Lambda/web app * Exposes HTTPS REST endpoints * Each endpoint has a different target * Can log to CloudWatch * Can configure multiple versions of API |
| To use | * Define API (container) * Define resources (URL Paths) * Chose HTTP verbs, set security, choose targets, set transformations * Free SSL/TLS certs if using Route 53 |
| API caching | * Reduce load/improve latency * Set TTL in seconds |
| Security | * CORS if using multiple domains * Use API key to track/control usage * Can throttle usage |
| Create API | * Has visual editor * Can create from New/existing/example or swagger API * Supports Open API (Open API 2.0, Open API 3.0 and Swagger) * Set HTTP verb * Set integration type (lambda, HTTP, mock, AWS service, VPC link) * Set proxy integration so lambda can see request * Set lambda function name and region * Can upload via copy/paste of zip file |
| Throttling | * 10K requests/second * max 5000 concurrent requests/millisecond across AWS account * 429 Error Code - Too Many Requests error |
| Access control | * IAM roles (sigv4), lambda authorizers, Cognito pools |

X-ray

|  |  |
| --- | --- |
| Overview | * Collects data about requests * X-ray SDK inside your app. API sends to X-ray * SDK adds interceptors to code to trace HTTP requests, calls to other AWS services and web services * View using X-ray console – shows error rate, traces, timings * Segment Documents - trace segment is a JSON representation of a request that your application serves. * Annotations - simple key-value pairs that are indexed for use with filter expressions. Use annotations to record data that you want to use to group traces in the console |
| Integrations | * Works with Elastic Load Balancer, Lambda, EC2, API Gateway, Elastic Beanstalk and SNS/SQS * For ECS, run X-Ray Daemon on it’s own Docker image(sidecar container) * Use X-Ray SDK and Daemon within EC2 instances, instrument application needed too |

Route 53

|  |  |
| --- | --- |
| Overview | * DNS Service. Map domain names to EC2, load balancer, S3 buckets |
| Create record set | * Maps domain name abc.com to AWS resource * CNAME record - CNAME records can be used to map one name to another (yourapp.3rdparty.com to www.mydomain.com) |
| Record Types | * **Alias** – used to map your record name (example.com – zone apex) to the DNS name for an AWS resource(elb1234.elb.amazonaws.com). Resolvers see the A or AAAA record and the IP address of the AWS resource. * **Cname** - You cannot create a CNAME record for example.com (zone apex), but you can create CNAME records for www.example.com, newproduct.example.com |
| Policy | * **Failover routing** - route traffic to a resource when the resource is healthy or to a different resource when the first resource is unhealthy * **Multivalue answer routing** - Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random * **Geolocation routing** - lets you choose the resources that serve your traffic based on the geographic location of your users, meaning the location that DNS queries originate from * **Geoproximity routing** – route traffic to your resources based on the geographic location of your users and your resources. You can also optionally choose to route more traffic or less to a given resource by specifying a value, known as a *bias*. |

DynamoDB

|  |  |
| --- | --- |
| Overview | * NoSQL database * Fully managed, autoscales, and single digit millisecond latency * Supports key-value and document data models * Stored on SSD * Spread across 3 data centers * Supports conditional writes and optimistic locking with version numbers |
| Consistency Model | * Eventually Consistent Reads – default. Might see stale data but usually less than a second to propogate data * Strongly Consistent Reads. All writes will be available for read |
| Terms | * Tables * Item – single record (MAX 400KB) * Attributes – key/value pairs * Key - name of data * Value - data * Documents – JSON, HTML and XML * Partition – physical storage location |
| Primary key | * Partition key – unique attribute that hashes to partition * Composite key – partition key + sort key. Partition key doesn’t need to be unique but * combined key does |
| Security | * IAM Condition – restricts access by record * Partition key must match user id * Must create new table to encrypt |
| Scan | * Examines all every item in the table, use ProjectionExpression to refine * Looks at everything in table – slower than query * Can add filter to limit results returned * Parallel scans to improve performance * Uses eventually consistent reads when accessing the data in a table |
| Query | * Find Items using only Primary Key attribute * ProjectionExpression – limit attributes returned * KeyCondition – like where clause * Better performance than scan * Must include primary key * Results sorted by sort key (or reverse with ScanIndexForward=false) * Defaults to Eventually Consistent |
| Local Secondary Index | * Must be created when create table, cannot add/remove later * Same partition key as table * Different sort key |
| Global Secondary Index | * Can create when create table or later * Different partition key than main table * Usefull when need to perform many kinds of queries, using a variety of different attributes other than the specified partition key as query criteria |
| Performance | * Can reduce impact by setting smaller page size to avoid throttling. * Can configure parallel scans. Bad if table already under heavy load |
| Capacity Units | * Measure of provisioned throughput * 1 write capacity unit is one 1KB write/second * 1 read capacity unit is 1 strongly consistent read of 4KB/second * 1 read capacity unit is 2 eventually consistent reads of 4KB/second * No fractional capacity units. Round up. |
| On Demand Capacity | * Autoscales based on activity * Don’t need to specify capacity in advance * Pay per request * Provisioned Capacity costs less if predicatable * Can switch one per day. |
| DynamoDB Accelerator (DAX) | * Fully managed, clustered in-memory cache * Up to 10x performance improvement * Microsecond response times * Ideal for read heavy bursty workloads * Writes to cache at same time as db * If not in cache, does eventually consistent get |
| Transactions | * (won’t be on exam before May 2019) * ACID, span tables |
| TTL | * Time to live, measured since 1970 * Expiration time for data * Marked for deletion and deleted within 48 hours * Reduces cost by automatically removing data |
| Streams | * Time ordered sequence of modifications * Guaranteed delivery exactly once * Logs stored 24 hours * Encrypted at rest * Separate endpoint from stream than db * Primary key always stored. Before/after can be stored too * Can trigger events – ex: lambda |
| If too many requests | * ProvisionedThoroughputExceededError * SDK will automatically retry until successful. * Use exponential backoff (applies to most AWS services). SDK does automatically. Jitter adds randomness so don’t all try at same second * Check if request size too big |
| Common APIs | * BatchGetItem, GetItem * BatchWriteItem, PutItem * DeleteItem, UpdateItem * Query, Scan |
| Global Tables | * Specify regions want table available * AWS replicates |

SQS

|  |  |
| --- | --- |
| Overview | * Simple queue service * Pull based * Up to 256 KB per message * Defaults to 4 days max in queue. Can increased to two weeks (14 days). * Can build in autoscaling * First AWS service * Supports dead-letter queues are messages that cannot be processed (consumed) successfully. Dead-letter queues are useful for debugging your application or messaging system because they let you isolate * PurgeQueue - Deletes the messages in a queue specified by the QueueURL parameter. |
| Types | * Standard – default queue, message delivered 1+, order not guaranteed * FIFO – message delivered exactly one in order. Ends with .fifo. Can use message group id to guarantee order within groups, when don’t want overall FIFO. |
| Visibility timeout | * Number seconds message invisible after reader picks it up * Message deleted if job processed, else available for processing again * Default 30 seconds * Maximum 12 hours * ChangeMessageVisibility - API call can you use to extend the length of time to process the jobs |
| Polling types | * Short polling – returns immediately * Long polling – waits for response or timeout. Maximum/default 20 seconds. Saves money when queue typically empty * ***Long VS Short*** - Long polling doesn’t return a response until a message arrives in the queue, reducing your overall cost over time. Short polling WILL return empty responses. |
| Delay Queues | * Postpone delivery of new messages, remaining invisible to consumers * Default is 0, maximum is 900 seconds |
| Large Messages | * Use AWS SDK for Java and SQS Extended Library * Use S3 for messages > 256KB up to 2GB |
| Ordering | * MessageGroupId - The message group ID is the tag that specifies that a message belongs to a specific message group. Messages that belong to the same message group are always processed one by one, in a strict order relative to the message group |
| Duplication | * MessageDeduplicationId - The message deduplication ID is the token used for deduplication of sent messages. If a message with a particular message deduplication ID is sent successfully, any messages sent with the same message deduplication ID are accepted successfully but aren't delivered during the 5-minute deduplication interval |

SNS

|  |  |
| --- | --- |
| Overview | * Simple Notification Service * Push based, pub-sub * Send to topics. Can have multiple subscribers |
| Types | * Devices, SMS, Email, Email JSON, SQS, HTTP, lambda |

SES

|  |  |
| --- | --- |
| Overview | * Simple Email Service * Can deliver to S3 or trigger lambda/SNS * Can use for incoming mail * Doesn’t require subscribing from the user |

Kinesis

|  |  |
| --- | --- |
| Overview | * Receive streaming data |
| Kinesis Streams | * Stores received data or video * Stored for a day; can increase to a week * Stores in shards. Use more to increase read/write in parallel * Send to consumers |
| Kinesis Firehose | * Data analyzed immediately using lambda or forwarded. Not stored locally * Forward data to S3 or ElasticSearch * Can forward from S3 to RedShift |
| Kinesis Analytics | * Run SQL queries from firehose/streams and send results to S3/ElasticSearch/RedShift |
| Kinesis Shards | * Use Auto Scale Group and base scaling decisions on CPU load on consumers |

Elastic Beanstalk

|  |  |
| --- | --- |
| Overview | * Upload code and automatically provision infrastructure * Control AWS resources created * Pay for EC2 and S3 created/used * Automatically scales * Integrated with CloudWatch and Xray * Uses AWS CloudFormation to launch the resources in your environment and propagate configuration changes * Resources that need to persist environments deletions must be created externally |
| Configuring in zip/war | * YAML or JSON format * In folder .ebextensions file \*.config |
| Integrating with RDS | * Launch from Elastic Beanstalk console. Within Elastic Beanstalk environment so deleted when delete app * For Prod, create standalone RDS. Create extra security group in autoscaling group. Add connection information to RDS. |
| Security | * Can set roles on service on instance |
| Supports | * Tomcat, Passenger, Docker, IIS |

Deployment strategies

|  |  |
| --- | --- |
| All at once/in place | * Outage while update all * If update fails, redeploy old version * Don’t use in Prod. * Elastic Beanstalk: All at once * CodeDeploy: In Place deployment - All at Once |
| Rolling | * Deploys in batches. * Less live instances while batches down for update. Can reduce performance * Repeat to rollback * Elastic Beanstalk: Rolling, Rolling with Additional Batch (the later launches a new batch so live instance count unchanged) * Code Deploy: In Place Deployment – One at a Time, Half at a Time |
| Immutable/blue green | * Starts new servers with new code * Maintains full capacity * Rollback is terminating new instances * Elastic Beanstalk: Immutable * Elastic Beanstalk & Code Deploy: Blue/Green * Blue/Green – (blue = active; green = new) Deploys to new environment Keeps old up for rollback. Instances are provisioned with new server in new environment and swap DNS. * Immutable – new autoscaling group on same environment |

Developer Tools

|  |  |
| --- | --- |
| CodePipeline | * Continuous deployment service to visualize/automate * Workflow of stages/tasks * Automatically configured so commits trigger CloudWatch which triggers CodePipeline * Enable versioning in S3 bucket * Get code from S3, CodeCommit or GitHub * Manual approvals steps fails if not approved in a week |
| OpsWorks | * Manage infrastructure/layers * Supports Chef and Puppet |

CloudFormation

|  |  |
| --- | --- |
| Overview | * Infrastructure as code * Template supports YAML and JSON * Can use to create/rollback/delete entire stack * Store template in S3 * Partial failure rollback entire stack * For each AWS account, export names must be unique within a region * Prevent CloudFormation from deleting your entire stack on failure:   + Set the 'Rollback on failure' radio button to No in the CloudFormation console   + Use the --disable-rollback flag with the AWS CLI |
| Template | * Resources section mandatory * Optional sections: AWSTemplateFormatVersion (must be 2010-09-09), Description, Metadata, Parameters (input when run. For Prod or Test), Conditions (based on env), Mappings (by region), Transform (include external code from s3), Outputs (to browser of another template) |

CloudWatch

|  |  |
| --- | --- |
| Overview | * Monitors performance and other stats * Can install agent for on-prem data |
| Host level metrics | * CPU, Network, Disk, Status Check of EC2 instance |
| Custom metrics | * RAM Utilization, not supported by default * Get data at minimum once a minute even if script runs more often |
| Frequency | * Default – 5 minute intervals * Detailed – 1 minute intervals * High resolution – 1 second intervals |
| Data storage | * Forever unless configure otherwise. * Logs not deleted when EC2/ELB terminated |
| Alarm | * Monitor any metric for sustained state changes * States: OK, ALARM, INSUFFICIENT\_DATA * Criteria: period of time, evaluation period, data points to alarm * High-resolution metric - specify a high-resolution alarm with a period of 10 seconds or 30 seconds, or you can set a regular alarm with a period of any multiple of 60 seconds. |
| Not done by CloudWatch | * CloudTrail logs API calls/activity * Config does state change |

AWS CLI

|  |  |
| --- | --- |
| Overview | * AWS Command Line Interface (CLI) is a unified tool to manage your AWS services * --dry-run option: Checks whether you have the required permissions for the action, without actually making the request |
| Pagination | * To avoid timeout errors related to too many results, use --page-size, example aws s3 list bucket –page-size 100 |

Disaster Recovery

|  |  |
| --- | --- |
| Overview | * A Disaster Recovery Plan (DRP) consists of structured, detailed and documented instructions to recover disrupted systems and networks |
| Types | * **Backup and Restore** - simply backup and restores data as needed. It has the drawback that none of the data is on standby, so it can be time-consuming. * **Pilot Light** - keeps critical applications and data ready to be quickly retrieved in the event of a disaster. * **Warm Standby** - keeps a duplicate version of the organization’s core elements running on standby at all times, so in a disaster, they can use this duplicate to maintain operations. * **Hot Standby** - replicates fully the data and applications in two or more active locations, splitting the traffic between them. In the event of a disaster, is simple to reroute everything to the unaffected region, meaning close to no downtime. |

Other Services

|  |  |
| --- | --- |
| Serverless services | * Lambda, API Gateway, S3, DynamoDB, SNS, SQS, Step Functions, Kinesis, Athena (queries), Tooling |
| ECS | * Elastic Container Service – manages containers |
| ECR | * Elastic Container Registry – image repository |
| CodeStar | * AWS CodeStar enables you to quickly develop, build, and deploy applications on AWS. AWS CodeStar provides a unified user interface, enabling you to easily manage your software development activities in one place |
| Canary Deployment | * 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 |
| Fargate | * Compute engine for Amazon ECS that allows you to run containers without having to manage servers or clusters. With AWS Fargate, you no longer have to provision, configure, and scale clusters of virtual machines to run containers. AWS Fargate eliminates the need to manage a cluster of Amazon EC2 instances |
| IOT Core | * IoT Core allows you to easily connect any number of devices to the cloud and to other devices. AWS IoT Core supports HTTP, WebSockets, and MQTT, a lightweight communication protocol specifically designed to tolerate intermittent connection |
| Snowball | * Petabyte-scale data transport solution that uses secure appliances to transfer large amounts of data into and out of the AWS cloud |
| Glue | * Fully Managed extract, transform and load (ETL) service to prepare and load data to analytics. |
| Global Accelerator | * Improves the availability and performance of your applications with local or global users. It provides static IP addresses that act as a fixed entry point to your application endpoints in a single or multiple AWS Regions, such as your Application Load Balancers, Network Load Balancers or Amazon EC2 instances |

HTTP Error codes

|  |  |
| --- | --- |
| 2xx | * Success |
| 3xx | * Redirection |
| 4xx | * Client error   + 400 – Bad request   + 401 – Unauthorized   + 403 – Forbidden   + 404 – Not found   + 409 – Conflict   + 429 – Too many requests/throttling error (API Gateway and Lambda) |
| 5xx | * Server error   + 500 – Server error   + 502 – Bad gateway   + 503 – Service unavailable   + 504 – Gateway timeout/not responding |