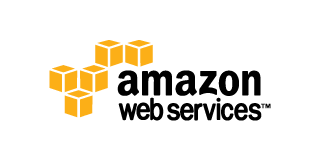
AWS SysOps Associate Exam Notes

  
For more information on AWS, visit [aws.amazon.com](https://aws.amazon.com/)

Description

Notes and information that were collected while studying and prepping for the AWS SysOps Associate Exam.

| **Topic** | **Answer** |
| --- | --- |
| Exam Time: | 80 Minutes |
| No. Questions: | 60 Questions |
| Question Types: | Scenario and Multiple Choice |
| Passing Score: | ~ 70% |
| Validity Period: | 2 years |
| Renewal Exam: | 1/2 price off |

Monitoring:

Monitoring is accomplished through the usage of CloudWatch, which is a service to monitor your AWS resources as well as the applications that you run on AWS.

**CloudWatch Monitoring:**

* Can monitor EC2 instances, Autoscaling Groups, ELBs, Route53 Health Checks, EBS Volumes, Storage Gateways, CloudFront, DynamoDB, ElastiCache nodes, RDS instances, EMR Job Flows, Redshift. SNS topics, SQS Queues, OpsWorks, CloudWatch Logs, Estimated charges on your AWS bill, and custom metrics | logs generated by your applications and services.
* EC2 will by default monitor your instances @5 minute intervals
* EC2 instances can monitor your instances @1 minute intervals if the 'detailed monitoring' option is set on the instance
* By default CloudWatch will monitor CPU, Network, Disk, and Status Checks
* RAM utilization is a custom metric and must be added manually to EC2 instances in order to be tracked.
* 2 types of Status Checks:
  + System Status Checks (Physical Host):
    - Checks the underlying physical host
    - Checks for loss of network connectivity
    - Checks for loss of system power
    - Checks for software issues on the physical host
    - Checks for hardware issues on the physical host
    - Best way to resolve issues is to stop the instance and start it again (will switch physical hosts)
  + Instance Status Checks
    - Checks the VM itself
    - Checks for failed system status checks
    - Checks for mis-configured networking or startup configs
    - Checks for exhausted memory
    - Checks for corrupted file systems
    - Checks for an incompatible kernel
    - Best way to troubleshoot is rebooting the instance or modifying the instance OS
* By default CloudWatch metrics are stored for 2 weeks
* Can retrieve data that is longer than 2 weeks using the GetMetricStatistics API endpoint, or by using third party tools
* Can retrieve data from any terminated EC2 or ELB instance for up to 2 weeks after its termination
* Many default metrics for many default services are 1 min, but it can be 3-5 minutes depending on the service
* Custom metrics have a minimum 1 minute granularity
* Alarms can be created to monitor any CloudWatch metric in your account
* Alarms can include EC2, CPU, ELB, Latency, or even changes on your AWS bill
* Within the alarm, actions can be set, triggering things like lambda functions, or SNS notifications if the alarm threshold is reached

[**Configuring custom metrics:**](https://aws.amazon.com/code/8720044071969977)

* In order to allow custom metrics to be written to CloudWatch, you must assign a CloudWatch full access role to the EC2 instance using the custom metrics.
* RAM utilization for example must be set up as a custom metric
* yum install -y perl-Switch perl-DateTime perl-Sys-Syslog perl-LWP-Protocol-https
* mkdir /CloudWatch && cd /CloudWatch
* wget http://aws-cloudwatch.s3.amazonaws.com/downloads/CloudWatchMonitoringScripts-1.2.1.zip
* unzip CloudWatchMonitoringScripts-v1.2.1.zip
* rm -fr CloudWatchMonitoringScripts-v1.2.1.zip
* cd aws-scripts-mon
* ./mon-put-instance-data.pl --mem-util --verify --verbose (dry run no data will be sent to CloudWatch)
* ./mon-put-instance-data.pl --mem-util --mem-used --mem-avail (set this up on 1/5 minute cron job)
* Set Cron job to run regulary (*/5 \**\* \* ec2-user /CloudWatch/mon-put-instance-data.pl --mem-util --mem-used)

**Monitoring EBS:**

* 4 Types of EBS Storage, General Purpose (SSD) - gp2, Provisioned IOPS (SSD) - io1, Throughput Optimized (HDD) - st1, and Cold (HDD) - sc1
* Throughput Optimized HDDs (ST1) and Cold HDDs (SC1), both CAN NOT BE USED AS BOOT VOLUMES!
* Throughput Optimized HDDs (ST1) and Cold HDDs (SC1), both are not available in the drop list if the volume is the root volume. Adding an additional volume will allow these option types to become present in the drop list.
* GP2 volumes have a base of 3 IOPS per GiB of volume size
* Maximum volume size is 16 TB
* Maximum IOPS size of 10K IOPS total (after which you need to move to provisioned IOPS storage tier)
* Can burst performance on the volume up to 3K IOPS
* Bursting uses I/O credits
* Each volume receives an initial I/O credit balance of 5.4 million I/O credits
* This is enough to sustain the max burst performance of 3K IOPS for 30 minutes (3K being the MAX iOPS available, including your standard 3 IOPS per GB. You can not burst an additional 3K to your standard, only burst up to a max of 3K)
* If you need more than 3K IOPS then you need to increase the volume size accordingly via the 3 IOPS per GB rule
* When not going over provisioned IO level (bursting) you earn credits back
* Don't need to know the calculation to replenish the credit balance
* New volumes no longer require pre-warming, they receive their maximum performance the moment that they are available and do not require initialization / pre-warming.
* When restoring a volume from snapshots, the first time you access the storage block, you can see a 5 to 50 % loss of IOPS due the volume either needing to be wiped clean or instantiated from a snapshot
* Performance is restored after the data is accessed once
* To avoid the performance hit, volumes can be pre-warmed
* For a new volume, you should write to all blocks before using the volume
* For a volume that has been restored from a snapshot, you should read all blocks that have data before using the volume
* Instructions for pre-warming volumes can be found [here](http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-initialize.html)
* EBS CloudWatch Metrics:
  + VolumeReadBytes
  + VolumeWriteBytes
    - Provides info on the I/O operations in a specified period of time
    - The SUM statistic reports the total number of bytes transferred during the period
    - The AVG statistic reports the average size of each I/O operation during the period
    - The SampleCount statistic reports the total number of I/O operations during the period
    - The Minimum and Maximum statistics are not relevant for this metric
    - Data is only reported to CloudWatch when the volume is active
    - If the volume is idle, no data is reported to CloudWatch
  + VolumeReadOps
  + VOlumeWriteOps
    - The total number of I/O operations in a specified period of time
    - To calculate the AVG IOPS for the period, divide the total operations in the period by the number of seconds in that period
  + VolumeTotalReadTime
  + VolumeTotalWriteTime
    - The total number of seconds spent by all operations that completed in a specified period of time
    - If multiple requests are submitted a the same time, the total could be greater than the length of the period
  + VolumeIdleTime
    - The total number of seconds in a specified period of time when no read or write operations were submitted
  + VolumeQueueLength
    - Then number of read and write operation requests waiting to be completed in a specified period of time
    - If the count is high, it would be a good indicator to up the volume size to get more IOPS available via the 3 IOPS per GiB rule
  + VolumeThroughputPercentage
    - Used with Provisioned IOPS (SSD) volumes only
    - The percentage of IOPS delivered of the total IOPS provisioned for an EBS volume
    - Provisioned IOPS SSD volumes deliver within 10% of the provisioned IPS performance 99.9% of the time over a given year
    - During a write, if there are no other pending I/O requests in a minute, the metric value will be 100%
    - A volume's I/O performance may become degraded temporarily due to an action that was taken (such as creating a snapshot of a volume during peak usage, or running the volume on a non-EBS-optimized instance, or accessing data on the volume for the first time, if the volume wasn't pre-warmed)
  + VolumeConsumedReadWriteOps
    - Used with Provisioned IOPS (SSD) volumes only
    - The total amount of read and write operations (normalized to 256K capacity units) consumed in a specified period of time
    - I/O operations that are smaller than 256K each count as 1 consumed IOPS
    - I/O operations that are larger than 256K are counted in 256K capacity units
* VolumeQueueLength can come up frequently, know what it is
* Volume Status Checks:
  + OK:
    - I/O Enabled status:
      * Enabled (I/O Enabled or I/O Auto-Enabled)
    - I/O Performance Status:
      * Only available for Provisioned IOPS (IO1) volumes
      * Normal (Volume performance is as expected)
  + Warning:
    - I/O Enabled status:
      * Enabled (I/O Enabled or I/O Auto-Enabled)
    - I/O Performance Status:
      * Only available for Provisioned IOPS (IO1) volumes
      * Degraded (Volume performance is below expectations)
  + Impaired:
    - I/O Enabled status:
      * Enabled (I/O Enabled or I/O Auto-Enabled)
      * Disabled (volume is off-line and pending recovery, or is waiting for the user to enable I/O)
    - I/O Performance Status:
      * Only available for Provisioned IOPS (IO1) volumes
      * Stalled (Volume performance is severely impacted)
  + Insufficient Data:
    - I/O Enabled status:
      * Enabled (I/O Enabled or I/O Auto-Enabled)
      * Insufficient Data
    - I/O Performance Status:
      * Only available for Provisioned IOPS (IO1) volumes
      * Insufficient Data
* Degraded, Severely Degraded = Warning
* Stalled or Not Available = Impaired
* If your EBS volume is attached to a current generation EC2 instance type, you can increase its size, change its volume type, or adjust its IOPS performance without detaching it
* These changes can be applied to detached volumes as well
* From the console (Volumes Console, Not EC2 Console), or from the API, Volumes can be modifed.
* When modifying a volume, you can monitor the progress of the modification. If the size of the volume was modified, be sure to extend the volumes file system to take advantage of the increased capacity.

**Monitoring RDS:**

* 2 types of monitoring:
  + Monitor by metrics (CloudWatch monitoring):
    - Per-Database Metrics
    - By Database Class
    - By Database Engine
    - Across All Databases
  + Monitor by events (RDS monitoring):
    - Located in Events tab
    - Events of everything that has happened with your instance
    - Can set event subscriptions which work like SNS topics
    - Events like fail-overs can be a notifying event using subscriptions
    - Available RDS Metrics:
      * BinLogDiskUsage
      * The amount of disk space occupied by binary logs on the master. Applies to MySQL read replicas
      * Units: Bytes
      * Burst Balance
      * The percent of General Purpose SSD (gp2) burst-bucket I/O credits available
      * Units: Percent
      * CPUUtilization
      * The percentage of CPU utilization.
      * Units: Percent
      * CPUCreditUsage (T2 Instances)
      * The number of CPU credits consumed by the instance
      * One CPU credit equals one vCPU running at 100% utilization for one minute or an equivalent combination of vCPUs, utilization, and time
      * Example: one vCPU running at 50% utilization for two minutes or two vCPUs running at 25% utilization for two minutes
      * Units: Count
      * CPUCreditBalance (T2 Instances)
      * The number of CPU credits available for the instance to burst beyond its base CPU utilization
      * Credits are stored in the credit balance after they are earned and removed from the credit balance after they expire
      * Credits expire 24 hours after they are earned
      * CPU credit metrics are available only at a 5 minute frequency
      * Units: Count
      * DatabaseConnections
      * The number of database connections in use
      * Units: Count
      * DiskQueueDepth
      * The number of outstanding IOs (read/write requests) waiting to access the disk
      * Units: Count
      * FreeableMemory
      * The amount of available random access memory
      * Units: Bytes
      * FreeStorageSpace
      * The amount of available storage space
      * Units: Bytes
      * MaximumUsedTransactionIDs
      * The maximum transaction ID that has been used. Applies to PostgreSQL
      * Units: Count
      * ReplicaLag (Seconds)
      * The amount of time a Read Replica DB instance lags behind the source DB instance. Applies to MySQL, MariaDB, and PostgreSQL Read Replicas
      * Units: Seconds
      * ReplicationSlotDiskUsage
      * The disk space used by replication slot files. Applies to PostgreSQL
      * Units: Megabytes
      * OldestReplicationSlotLag
      * The lagging size of the replica lagging the most in terms of WAL data received. Applies to PostgreSQL
      * Units: Megabytes
      * TransactionLogsDiskUsage
      * The disk space used by transaction logs. Applies to PostgreSQL
      * Units: Megabytes
      * TransactionLogsGeneration
      * The size of transaction logs generated per second. Applies to PostgreSQL
      * Units: Megabytes/second
      * SwapUsage
      * The amount of swap space used on the DB instance
      * Units: Bytes
      * ReadIOPS
      * The average number of disk I/O operations per second
      * Units: Count/Second
      * WriteIOPS
      * The average number of disk I/O operations per second
      * Units: Count/Second
      * ReadLatency
      * The average amount of time taken per disk I/O operation
      * Units: Seconds
      * WriteLatency
      * The average amount of time taken per disk I/O operation
      * Units: Seconds
      * ReadThroughput
      * The average number of bytes read from disk per second
      * Units: Bytes/Second
      * WriteThroughput
      * The average number of bytes written to disk per second
      * Units: Bytes/Second
      * NetworkReceiveThroughput
      * The incoming (Receive) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication
      * Units: Bytes/second
      * NetworkTransmitThroughput
      * The outgoing (Transmit) network traffic on the DB instance, including both customer database traffic and Amazon RDS traffic used for monitoring and replication
      * Units: Bytes/second
    - Have general idea of what each of the RDS metrics do
    - DatabaseConnections, DiskQueueDepth, FreeStorageSpace, ReplicaLag (Seconds), ReadIOPS, WriteIOPS, ReadLatency, WriteLatency are all important ones to know

**Monitoring ELB:**

* Monitored every 60 seconds provided there is traffic
* Only reports when requests are flowing through the LB
* If there are no requests or data for a given metric, the metric will not be reported to CloudWatch
* If there are requests flowing through the LB, ELB will measure and send metrics for that LB in 60 second intervals
* Available Metrics:
  + HealthyHostCount:
    - The count of the number of healthy instances in each AZ
    - Hosts are declared healthy if they meet the threshold for the number or consecutive health checks that are successful
    - Hosts that have failed more health checks then the value of the unhealthy threshold are considered unhealthy
    - If cross-zone is enabled, the count of the number of healthy instances is calculated for all AZs
    - Preferred Statistic: Average
  + UnHealthyHostCount:
    - The count of the number of unhealthy instances in each AZ
    - Hosts that have failed more health cheeks than the value of the unhealthy threshold are considered unhealthy
    - If cross-zone is enabled, the count of the number of unhealthy instances is calculated for all AZs
    - Instances may become unhealthy due to connectivity issues, health checks returning non-200 responses (in the case of HTTP or HTTPS health checks), or timeouts when performing the health check
    - Preferred Statistic: Average
  + RequestCount:
    - The count of the number of completed requests that were received and routed to the back end instances
    - Preferred Statistic: Sum
  + Latency:
    - Measures the time elapsed in seconds after the request leaves the load balancer until the response is received
    - Preferred Statistic: Average
  + HTTPCode\_ELB\_4XX
    - The count of the number of HTTP 4XX client error codes generated by the load balancer when the listener is configured to use HTTP or HTTPS protocols. Client errors are generated when a request is malformed or is incomplete
    - Preferred Statistic: Sum
  + HTTPCode\_ELB\_5XX
    - The count of the number or HTTP 5XX server error codes generated by the load balancer when the listener is configured to use HTTP or HTTPS protocols
    - This metric does not include any responses generated by back end instances
    - The metric is reported if there are no back-end instances that are healthy or registered to the load balancer, or if the request rate exceeds the capacity of the instances or the load balancers
    - Preferred Statistic: Sum
  + HTTPCode\_Backend\_2XX:
  + HTTPCode\_Backend\_3XX:
  + HTTPCode\_Backend\_4XX:
  + HTTPCode\_Backend\_5XX:
    - The count of the number of HTTP response codes generated by back-end instances
    - Metric does not include any response codes generated by the load balancer
    - The 2XX class status codes represent successful actions
    - The 3XX class status codes indicate that the user agent requires action
    - The 4XX class status code represents client errors
    - The 5XX class status code represents back-end server errors
    - Preferred Statistic: Sum
  + BackendConnectionErrors:
    - The count of the number of connections that were not successfully established between the LB and the registered instances
    - The LB will retry when there are connection errors, so the count can exceed the request rate
    - Preferred Statistic: Sum
  + SurgeQueueLength:
    - A count of the total number of requests that are pending submission to a registered instance
    - Preferred Statistic: Max
  + SpilloverCount:
    - A count of the total number of requests that were rejected due to the queue being full
    - Preferred Statistic: Sum
* Have an idea of what each metric does
* Important metrics to note are SurgeQueueLength & SpilloverCount

**Monitoring Elasticache:**

* Consists of 2 different engines:
  + Memcached
  + Redis
* When it comes to monitoring cache engines, there are 4 monitoring points:
  + CPU Utilization
    - Memcached:
    - Multi-threaded
    - Handles loads of up to 90% CPU utilization
    - If > 90% CPU utilization add more nodes to the cluster
    - Redis:
    - Single-threaded
    - Take 90% and / number of cores to determine scale point
    - Will not have to calculate Redis CPU utilization in exam
  + Swap Usage
  + Memcached:
    - Should be around 0 most of the time and should not exceed 50MB
    - If 50MB is exceeded, you should increase the memecached\_connections\_overhead parameter
    - memecached\_connections\_overhead defines the amount of memory to be reserved for Memcached connections and other misc. overhead
  + Redis
    - No SwapUsage metric, instead use reserved-memory
  + The amount of the Swap file that is used.
  + Swap file is the amount of disk storage space reserved on disk if your computer runs out of RAM
  + Typically the size of the swap file = the amount of RAM available
  + Evictions
  + Memcached:
    - No recommended setting
    - Choose a threshold based off your application
    - Scale up (increase the memory of existing nodes) or Scale out (add more nodes) to avoid evictions
  + Redis:
    - No recommended setting
    - Choose a threshold based off your application
    - Only scale out (add read replicas) to avoid evictions
  + Like clowns stuffed in a car, There is a finite number of empty seats that slowly fill up. Eventually the car is full and if more seats are needed, then an Eviction will occur
  + Evictions occur when a new item is added and an old item must be removed due to lack of free space on the system
  + Concurrent Connections
  + No recommended setting
  + Choose a threshold based off your application
  + If there is a large and sustained spike in the number of concurrent connections, this can either mean a large traffic spike or your application is not releasing connections efficiently

Organizations & Consolidated Billing:

AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage

**Consolidated Billing:**

* Have a single payer account
* Have multiple linked accounts that all roll up to the payer account for billing purposes
* Payer account is independent and cannot access resources of any linked account
* Linked accounts are also independent and cannot access resources in any of the other linked accounts, or the payer account
* Currently there is a limit of 20 linked accounts for consolidated billing, unless a limit increase is requested
* Advantages include a single bill per AWS account, easy way to track charges and allocate costs, and volume pricing discount availability
* Always enable MFA on root account, always use strong and complex passwords on the root account, Payer account should be used for billing purposes only, do not deploy resources in the payer account
* When monitoring is enabled on the payer account, the billing data for all linked accounts is also included
* You can still create billing alerts per individual accounts as well
* CloudTrail is per AWS account and is enabled per region
* CloudTrail can be aggregated in to a single bucket in the payer account
* Consolidated billing allows you to get volume discounts on all of your accounts
* Unused reserved instances for EC2 are applied across the group

**Cost Optimization:**

* 3 different instance types:
* Spot
  + Allow you to name your own price for EC2 capacity
  + You bid on spare EC2 instances and these will run automatically whenever your bid exceeds the current spot price
  + Spot price varies in real time based on supply and demand
  + If the spot price goes above your bid price after the instances are provisioned, the instances will be automatically terminated
* Reserved Instances
  + Provide you with up to 75% discount as compared to on-demand pricing
  + You are assured that your RI will always be available for the OS and AZ in which you purchased it
  + For applications that have steady state needs, RIs provide significant savings compared to using on-demand instances
  + RI's and On-demand instances perform identically.
* On Demand
  + Pay for compute capacity by the hour with no long term commitments or upfront payments
  + Increase or decrease your compute capacity depending on the demands of your application and only pay the specified hourly rate for the instances that you use
  + EC2 always strives to have enough capacity to meet customer needs, but during high periods of high demand, it is possible that you may not be able to launch specific instance types in specific AZs

Elasticity and Scalability:

Elasticity is focused around being able to scale your infrastructure up, and down automatically based on traffic, where Scalability is focused on scaling your infrastrucure out more permanentlty. - Elasticity - Allows you to stretch out and retract your infrastructure based on demand - Pay for only what you need - Used during a short time period, such as hours or days - EC2: - Increase instance sizes as required using RIs - DynamoDB - Increase additional IOPS for additional spikes in traffic, then decrease IOPS after the spike - RDS - Not elastic, can't scale RDS based on demand

* Scalability
* Used to talk about building out the infrastructure to meet your demands long term
* Used over a longer time period such as weeks, days months and years
* EC2:
  + Increase the number of EC2 instances based on Autoscaling
* DynamoDB
  + Unlimited amount of storage
* RDS
  + Increase instance size from small to medium
* Scale UP vs Scale Out
* Scale Up
  + Increase the number of CPUs, RMA, or the amount of storage
  + EC2: increase the instance type from say a T1.micro to T2.small or T2.medium, etc..
  + If questions appear to be network related, then its probably a scale up answer
* Scale Out
  + Add more resources such as web servers
  + EC2: add additional EC2 instances and Autoscaling
  + If questions appear to be in relation to not having enough resources, then its probably a scale out answer.

RDS Multi Availability Zones & Failover:

Multi AZ Deployment:

* Multi AZ deployments for MySQL, PostgreSQL, and Oracle engines utilize synchronous physical replication to keep data on the standby up to date with the primary
* Multi AZ deployments for the MSSQL Server engine uses synchronous logical replication to achieve the same result, employing SQL Server native mirroring technology
* Both approaches safeguard your data in the event of a DB instance failure or loss of an AZ
* Failovers are handled via DNS moves from a primary to secondary instances on the backend
* During a failover your connection URL string does not change
* High Availability:
* Backups are taken from secondary which avoids I/) suspension to the primary
* Restores are taken from the secondary which avoids I/o suspensions to the primary
* You can force a failover from one AZ to another by rebooting your instance. This can be done from the AWS management console or by using the RebootDBInstance API Call
* RDS Multi AZ failover is NOT A SCALING SOLUTION
* Read Replicas are used to scale only

Read Replicas:

* MySQL, PostgreSQL, MariaDB
* Amazon uses these engines native asynchronous replication to update the read replica
* Aurora
* Not yet covered by the exam
* Not using Asynchronous replication
* Uses an SSD backed virtualized storage layer purpose built for DB workloads
* Aurora replicas share the same underlying storage as the source instance
* Using same storage lowers cost and avoids the need to copy data to the replica nodes over the network
* Make it easy to take advantage of supported engines built in replication functionality
* Used to elastically scale out beyond the capacity constraints of a single DB instance for read heavy workloads
* You can create a read replica within a few clicks in the AWS management console
* Can also create a read replica with the CreateDBInstanceReadReplica API call
* Once the read replica is created, database updates on the source DB instance will be replicated using a supported engines native asynchronous replication
* Can create multiple read replicas for a given source DB instance and distribute your applications read traffic among them
* Can have up to 5 read replicas for any 1 primary db instance
* When to use Read Replicas:
* Scaling beyond the compute or I/O capacity of a single DB instance for read heavy database workloads
* The access read traffic can be directed to one or more read replicas
* Serving read traffic while the source DB instance is unavailable
* If your source DB instance cannot take I/O requests, you can direct read traffic to your read replicas
* Commonly used for business reporting or data warehousing scenarios
* Reports or data warehousing queries are typically ran against read replicas instead of the primary production DB instance
* Creating a read replica
* AWS takes a snapshot of your database
* If Multi AZ is not enabled, the snapshot will be of the primary database and can cause brief I/O suspension for around 1 minute
* If Multi AZ is enabled, then snapshots will be taken of the secondary database and there will be no performance impact on your primary db
* Connecting to a read replica
* Read replicas have a new DNS a record created that should be used to directly access the read replica when the read replica is created
* You can promote a read replica to its own standalone db.
* Doing this will break the replication link between the primary and secondary db
* Read Replica Tips:
* Can have up to 5 read replicas for MySQL, PostgreSQL, and MariaDB
* Can have read replicas in different regions for all engines
* Replication is asynchronous only not synchronous
* Read replicas can be built off Multi AZ DBs
* Read replicas themselves cannot be Multi AZ currently
* Can have read replicas of read replicas, but beware of latency
* DB snapshots and automated backups cannot be taken of read replicas
* Key metric is ReplicaLag
* Know differences between read replicas and Multi AZ RDS instances

RDS Multi AZ and Read Replicas:

* When you delete an RDS database, the default action will be set to take a final snapshot, this can be over-ridden with a drop down on the delete screen in the AWS management console
* DB Instance Identifiers which are created during the creation of an RDS instance must be unique withing the same AWS account for each RDS instance created
* Instance Identifier is the name of the database instance, not the database name, which is specified on a separate view when creating in the AWS management console
* Can now Enable IAM DB authentication, which will allow you to control authentication into DB instances via IAM users and groups
* Cannot create a read replica initially because no snapshot is present, must take a snapshot in order to create a read replica
* Multi AZ can be turned on from a standalone. This can be done via a snapshot and restore, or you can modify an existing RDS instance, and change the Multi AZ Deployment drop option from No to Yes
* Modifying the database will take the database off line while it applies its modifications
* Modifying or changing an existing db will not change the DNS record
* Creating a new instance from an existing snapshot will provision a new DNS record
* In order to create a read replica of a read replica, Database backups must be turned on (or a snapshot must exist)
* RDS Tips
* Know the difference between read replicas and Multi AZ (scale out vs DR)
* If you can't create a read replica you most likely have disabled db backups, change it and turn it on
* you can create read replica of read replica's in multiple regions
* you can modify the DB itself or create a new database from a snapshot
* Endpoints DO NOT CHNAGE if you modify a db, they will change if you create a new d from a snap or if you create a read replica
* You can manually fail over a multi AZ DB from one AZ to another by rebooting it

Connectivity and Troubleshooting:

**Connectivity:**

* Bastion hosts serve as a more secure way to connect to your VPC and AWS Infrastructure components
* Bastion hosts act as a gateway between you and your EC2 instances
* Bastion hosts help reduce attack vectors on your infrastructure and means that you only have to harden 1-2 EC2 instances as opposed to the entire fleet
* 1 subnet = 1 AZ, a single subnet cannot span more than 1 Availability Zone
* Bastion hosts are used by allowing SSH/RDP connections directly to the hosts, and only those hosts are allowed to SSH/RDP into the rest of your EC2 instances

**High Availability Troubleshooting:**

* Things to look for if your instances are not launching into an Autoscaling group:
* Associated Key Pair does not exist
* Security Group does not exist
* Autoscaling config is not working correctly
* Autoscaling group not found
* Instance type specified is not supported in the AZ
* Invalid EBS device mapping
* Autoscaling service is not enabled on your account
* Attempting to attach EBS block device to an instance store AMI

Elastic Load Balancers:

**Root Access:**

* The following services still allow root access to the hosts provisioned by the corresponding services:
* Elastic Beanstalk
* Elastic MapReduce
* OpsWorks
* EC2
* ECS

**ELB Configurations:**

* You can use ELBs or Elastic Load Balancers to load balance across different AZ's within the same region, but not to different regions or different VPC's themselves
* An ELB is different than a NAT
* Can have 2 types of ELBs:
* External ELB's with External DNS names
* Internal ELB's with Internal DNS names
* Health checks can be configured to check backend services via protocols such as HTTP/HTTPS
* Health check intervals are calculated by multiplying the Health Check Interval x Healthy or Unhealthy Threshold value.
* In the example that the HC Interval is 30 sec, and the Threshold is set to 2, after 2 30 second cycles or 1 minute the host will be marked unhealthy
* Supports Sticky Sessions:
* Not enabled by default
* By default the ELB routes each request independently to the application with the smallest amount of load
* The sticky session feature (session affinity) enables the ELB to lock a user down to a specific web server (EC2 instance)
* All requests at that point from the user during the session are always sent to the same server
* To manage sessions, determine how long your ELB should consistently route the user's request to the same application server
* 2 Types of session stickiness:
  + Duration based:
  + Most commonly used
  + The ELB creates a session cookie
  + When the ELB receives a request, it checks to see if this cookie is present in the request.
  + If the cookie is present, then the request is sent to the server specified in the cookie.
  + If the cookie is not present, the ELB chooses a backend server based on the existing load balancing algorithm and adds a new cookie to the response
  + The stickiness policy config defines the cookie expiration, which establishes the duration of validity for each cookie
  + The cookie is automatically updated after the duration expires.
  + If the backend sever fails or becomes unhealthy, the ELB stops routing requests to it and instead chooses a new instance based on the selected algorithm
  + In the event of failure, the request is routed to the new instance as if there is no cookie and the session is no longer sticky
  + Application controlled:
  + The ELB uses a special cookie to associate the session with the original server that handled the request, but follows the lifetime of the application generated cookie corresponding to the cookie name specified in the policy configuration
  + The ELB only inserts a new stickiness cookie if the application response includes a new application cookie
  + The ELB stickiness cookie does not update with each request
  + If the ELB stickiness cookie is explicitly removed or expires, the session stops being sticky until a new application cookie is issued
  + If an application instance fails or becomes unhealthy, the ELB stops routing requests to that instance, and instead chooses a new healthy instance based on the existing load balancing algorithm
  + The ELB will treat the session as now stuck to the new healthy instance and continue routing requests to that instance even if the failed instance comes back online
  + It is up to the new application instance whether and how to respond to a session which it has not previously seen
* ELB Metrics:
  + HealthyHostCount - The number of healthy instances in each AZ. Hosts are declared healthy if they meet the threshold for the number of consecutive health checks that are successful. Hosts that have failed more health checks then the value of the unhealthy threshold are considered unhealthy. If cross-zone is enabled, the count of the number of healthy instances is calculated for all AZ's. The preferred statistic is the average
  + UnHealthyHostCount - The count of the the number of unhealthy instances in each AZ. Hosts that have failed more health checks then the value of the unhealthy threshold are considered unhealthy. If cross-zone is enabled the count of the number of unhealthy instances is calculated for all AZ's. Instances may become unhealthy due to connectivity issues, health checks returning non 200 responses (in the case of HTTP or HTTPS health checks), or timeouts when performing the health check. The preferred statistic is the average
  + RequestCount - The count of the number of completed requests that were received and routed to the back end instances. The preferred statistic is the sum
  + Latency - Measures the time elapsed in seconds after the request leaves the ELB until the response is received. The Preferred statistic is the average
  + HTTPCode\_ELB\_4XX - The count of the number of 4XX client error codes generated by the load balancer when the listener is configured to use HTTP or HTTPS. Client errors are generated when a request is malformed or incomplete. The preferred statistic is the sum
  + HTTPCode\_ELB\_5XX - The count of the number of HTTP 5XX server error codes generated by the load balancer when the listener is configured to use HTTP or HTTPS. This metric does not include any responses generated by back end instances. The metric is reported if there are no back end instances that are healthy or registered to the load balancer, or if the request rate exceeds the capacity of the instances or the load balancers. The preferred statistic is sum
  + HTTPCode\_Backed\_2/3/4/5XX - The count of the number of HTTP response codes generated by back end instances. This metric does not include any response codes generated by the load balancer. The 2XX class status codes represent successful actions. The 3XX status codes indicate that the user agent requires action. The 4XX class status codes represent client errors, and the 5XX class status codes represents back end instance errors. The preferred statistic is sum
  + BackendConnectionErrors - The count of the number of connections that were not successfully established between the load balancer and the registered instances. Because the load balancer will retry when there are connection errors, this count can exceed the request rate. The preferred statistic is sum
  + SurgeQueueLength - The count of the total number of requests that are pending submission to a registered instance. The preferred statistic is max
  + SpilloverCount - A count of the total number of requests that were rejected due to the queue being full. The preferred statistic is sum
  + Pay attention to SurgeQueueLength and SpilloverCount
* Pre Warming ELB's:
  + AWS can pre-configure the ELB to have the appropriate level of capacity based on expected traffic
  + Used in scenarios such as when flash traffic is expected, or in the case where a load test cannot be configured to gradually increase traffic
  + This can be done by contacting AWS prior to the expected event. You will need to know the following:
  + The start and end date of the expected flash traffic
  + The expected request rate per second
  + The total size of the typical request/response that you will be sending/receiving

Backups and Disaster Recovery:

**Disaster Recovery:**

* DR is about preparing for and recovering from a disaster
* Any event that has a negative impact on a company's business continuity or finances could be termed a disaster
* Disasters include software, hardware, network failures, power outages, physical damage to buildings like fire or flooding, human error, etc..
* Traditional approaches involve an N+1 approach and have different levels of off-site duplication of data and/or infrastructure
* Advantages of using AWS for DR
* Only minimum hardware is required for data replication
* Allows you flexibility depending on what your disaster is and how to recover from it
* Open cost model (pay as you use) rather than heavy investment upfront
* Scaling is quick and easy
* Automate infrastructure for DR deployments
* AWS storage Gateways:
* Gateway cached volumes store primary data and cache most recently used data locally
* Gateway stored volumes store entire datasets on site and asynchronously replicate data back to S3
* Gateway virtual tape libraries store your virtual tapes in either S3 or Glacier
* RTO vs RPO
* RTO or Recovery Time Objective is the length of time from which you can recover from a disaster
* RTO is measured from when the disaster first occurred to when you have fully recovered from it
* RPO or Recovery Point Objective is the amount of data your organization is prepared to lose in the event of a disaster
* RPO examples are only allowing 1 day of email loss, or 5 hours of transaction records lost, 24 hours of backups, etc..
* Typically the lower RTO and RPO thresholds that are set, the more costly the solution will be

**DR Strategies:**

* Pilot Light:
* The term used to describe a DR scenario in which a minimal version of the environment is always running in the cloud
* Similar to a backup and restore scenario. AWS can maintain a pilot light by configuring and running the most critical core elements of your system in AWS. When the time comes for recovery, the environment can quickly provision a full scale production environment around the critical core via auto-scaling and other measures.
* Typically includes Databases, which could be replicated to RDS or EC2 instances along with any other critical core components
* Rest of your infrastructure can be set up using pre-configured AMI's and Cloudformation
* For networking, use pre-allocated EIP's and associate them with your instances when invoking DR, or use pre-allocated ENI's with pre-allocated MAC addresses for applications with special licensing requirements
* Use ELBs to distribute traffic to multiple instances, and update DNS records to point at your EC2 instances or point to the ELB's using CNAME records
* Warm Standby:
* Used to describe a DR scenario in which a scaled down version of a fully functional environment is always running in the cloud
* Extends the pilot light elements and decreases the recovery time because some services are always running
* By identifying business critical systems, you can fully duplicate those systems on AWS and have them always on
* Critical components can be running on minimum sized instances. The scenario is not scaled to handle production load, but is fully functional
* Can be used for non production work, such as testing, QA and internal use
* In a disaster, the system can be scaled horizontally or vertically quickly to handle production load
* In AWS you can simply add more instances to the environment or by resizing the small capacity servers to run on larger instance types
* Horizontal scaling is preferred over vertical scaling
* To set up Warm Standby:
  + Set up EC2 instances to replicate or mirror data
  + Create and maintain AMIs
  + Run your application using a minimal footprint of EC2 instances or AWS infrastructure
  + Patch and update software and configuration files in line with your environment
  + Increase the size of the EC2 fleet in service with the load balancer (horizontal scaling)
  + Start applications on larger EC2 instance types as needed (vertical scaling)
  + Either manually change the DNS records or use Route53's automated health checks so that all traffic is routed to the AWS environment
  + Consider using auto scaling to right size the fleet or accommodate the increased load
  + Add resilience or scale up your database
* Multi-Site:
* Runs in AWS as well as your existing on-site infrastructure in an active active configuration
* The data replication method that you employ will be determined by the recovery point that you choose
* You can use Route53 to root traffic to both sites either symmetrically or asymmetrically
* In an on-site disaster recovery situation, you can adjust the DNS weighting and send all traffic to AWS servers
* The capacity of the AWS service can be rapidly increased to handle the full production load
* You can use EC2 Auto scaling to automate the process
* May need some application logic to detect the failure of the primary database service and cut over to the parallel database service running in AWS
* To set up Multi-site Standby:
* Set up AWS environment to duplicate your production environment
* Set up DNS weighting or a similar traffic routing technology, to distribute incoming requests to both sites
* Configure automatic failover to re-route traffic away from the affected site
* Have application logic for failover to use the local AWS database servers for all queries
* Move from DR Site back to primary site:
  + Establish reverse mirroring / replication from the DR site back to the primary site
  + Wait for primary site to catch up to DR site
  + Freeze data changes to the DR site
  + Re-Point users back to the primary site
  + UnFreeze the changes

**Backups:**

* Traditionally data is backed up to tape and sent off site regularly
* Using the tape method can take a long time to restore your system in the event of a disaster
* S3 is an ideal destination for backup data that might need to be restored quickly
* Transferring data to and from S3 is typically done through the network and therefor accessible from any location
* Can use AWS import/export to transfer very large data sets by shipping storage devices directly to AWS
* For longer term data storage where retrieval times of several hours are adequate, Glacier can be leveraged
* Glacier has the same durability model as S3, and can be used in conjunction with S3 to produce a tiered backup solution
* Select an appropriate tool or method to backup your data to AWS
* Ensure you have an appropriate retention policy for your data
* Ensure that appropriate security measures are in place for the data including encryption and access policies
* Regularly test the recovery and restoration of the data and applicable systems
* Moving from DR back to Primary Site:
* Freeze data changes to the DR site
* Take backup
* Restore the backup to the primary site
* Re-point users to the primary site
* Unfreeze changes

**Services with automated backups:**

* RDS
* Need InnoDB (translated Engine)
* Performance hit if Multi-AZ is not enabled
* If you delete an instance, then ALL automated backups are deleted
* Manual DB snapshots will NOT be deleted
* All backups stored on S3
* When you do a restore, you can change the engine type (SQL standard to SQL Enterprise for example), provided you have enough space
* Elasticache (Redis Only)
* Available for Redis Cache Cluster only
* The entire cluster is snapshotted
* Snapshots WILL degrade performance
* Set the snapshot window during the least busy part of the day
* All snapshots stored on S3
* Redshift
* By default Redshift enables automated backups of your data warehouse cluster with a 1 day retention
* Redshift only backs up data that has changed, so most snapshots only use up a small amount of backup storage
* All snapshots stored on S3
* EC2 does NOT have automated backups (Can take snapshots, but they are not automated)
* No automated backups
* Backups degrade your performance, schedule these times during off peak hours
* Can create automated backups using either the CLI interface or Python
* Snapshots are Incremental:
  + Snapshots only store incremental changes since the last snapshot
  + Only charged for incremental storage
  + Each snapshot still contains the base snapshot data
* All snapshots are stored on S3

EC2 & EBS:

**EC2:**

* When EC2 was first launched all AMI's were backed by Instance store or Ephemeral storage
* Ephemeral storage is non-persist or temporary storage
* When an instance is shut down, even if turned back up, the the contents of the instance store, or ephemeral storage will be gone, and unaccessible
* Stopping and restarting an instance moves the instance to another host, hence the lost data
* EC2 eventually got the ability to attach EBS or Elastic Block Storage which allows for data persistence
* There is NO way to flag data preservation on ephemeral storage, if the instance restarts, or the host experiences issues, you can incur data loss
* 2 types of Volumes
* Root Volume:
  + This is where your operating system is installed
  + Can either be EBS or Ephemeral
  + Max size is 10GB
  + EBS root device volume can be up to 1 or 2TB depending on OS
  + Delete on Terminate is the default value
* Additional Volumes:
  + This can be your D:, E:, F: / dev/sdb, /dev/sdc, /dev/sdd etc..
  + Delete on Terminate is NOT the default value, additional volumes WILL persist after the instance is terminated and must be manually deleted

**EBS:**

* Allows users to have data persistence
* EBS volumes can be detached from an instance and attached to other instances without data loss
* EBS volumes can only be attached to a single instance at a time
* EBS root volumes are terminated/deleted by default when the EC2 instance is terminated
* Termination/Deletion default behavior can be stopped by un-selecting the "Delete on Termination" option when creating the instance or by setting the deleteontermination flag to false using the command line at boot time
* Non root EBS volumes attached to the instance are preserved if you delete the instance
* Boot time is quicker using EBS, typically less than 1 minute, where Instance store volumes are generally less than 5 minutes
* Must manually delete additional EBS volumes when an instance is terminated. Failure to do so will hold a storage charge for unattached non deleted volumes

**Snapshots:**

* Exist on S3, you do not have access to the snapshots directly, but on the backend they are stored on S3
* Snapshots are point in time copies of volumes
* Snapshots are incremental, only the blocks that have changed since your last snapshot are moved to S3
* The first snapshot takes some time to create as its a full snapshot of the volume
* To create a snapshot for EBS volumes that serve as root devices you should stop the instance before taking the snapshot
* You can take a snapshot while the instance is running
* You can create AMI's from both volumes and snapshots
* You can change EBS volume sizes on the fly, including changing the size and storage type
* Volumes will ALWAYS be in the same AZ as the EC2 instance
* To move an EC2 volume from one AZ/Region to another, take a snapshot or an image of it and then copy it to the new AZ/Region
* Snapshots of encrypted volumes are encrypted automatically
* Volumes restored from encrypted snapshots are encrypted automatically
* You can share snapshots, but only if they are unencrypted
* Shared snapshots can be shared with other AWS accounts or made public

Opsworks:

* Cloud based applications usually require a group of related resources that must be created and managed collectively
* The collection of instances is called a stack
* Opsworks provides a simple and straight forward way to create and manage stacks and their associated resources
* Opsworks is an application management service that helps automate operational tasks like code deployment, software configurations, package installations, database setups, and server scaling using Chef
* Provides the flexibility to define your application architecture and resource configuration and handles the provisioning and management of your AWS resources for you
* Includes automation to scale your application based on time or load, monitoring to help troubleshot and take automated actions based on the state of your resources
* Handles permissions and policy management to make management of multi-user environments easier
* Chef turns infrastructure into code
* Can automate how you build, deploy, and manage your infrastructure
* Allows infrastructure to become as versionalble, testable, and repeatable as application code
* Chef server stores your recipes as well as other configuration data
* The chef client is installed on each server, instance, container or networking device that you manage referred to as nodes
* The client periodically polls the Chef server for the latest policy and state of the network, if anything is out of date, the client brings it up to date
* Opsworks provides a GUI to deploy and configure your infrastructure quickly
* Consists of 2 elements, Stacks and Layers
* A stack is a container or group of resources such as ELBs, EC2 instances, RDS instances, etc
* A layer exists within a stack and consists of things like a web application layer
* Think of a stack as a virtual data center
* Each function is a different layer, you can wrap up the full configuration of a component within a layer such as PHP, Apache, etc..
* Need 1 or more layer per stack
* An instance must be assigned to at least 1 layer
* Which Chef layers run, are determined by the layer the instance belongs to
* There are pre-configured layers that will auto provision things such as Applications, Databases, Load balancing, or Caching
* if you select an existing ELB to be used in a layer, Opsworks will remove any currently registered instances and then manages the ELB for you. If you use the ELB console to modify the configuration, the changes will NOT be permanent

Security:

**Shared Security Model:**

* AWS Responsibilities:
* Securing the underlying infrastructure that supports the cloud
* Protecting the global infrastructure that runs all of the services offered on AWS
* All hardware, software, networking, and facilities that run AWS services
* Security configuration of its products and services that are considered managed services
  + DynamoDB
  + RDS
  + Redshift
  + EMR
  + Workspaces
  + Workmail
  + etc...
* Patching of managed service nodes
* Antivirus for managed service nodes
* Storage device decommissioning, with prevention of customer data exposure
* AWS uses techniques detailed in DoD 5220.22-M or NIST 800-88 to destroy data as part of its decommissioning process
* All decommissioned magnetic storage devices are degaussed and physically destroyed in accordance with industry standard practices
* AWS corporate network is completely segregated from the AWS production network by means of complex network security devices
* AWS provides protection against DDOS, Man in the Middle attacks, Ip Spoofing, Port Scanning and Packet Sniffing by other tenants
* Different instances run on the same physical hardware and are isolated from each other via the Xen hypervisor
* AWS has firewalls that reside in the hypervisor layer, between the physical network interface and the instances virtual interfaces
* All network packets must pass through the firewall layer, ensuring that no instance has access to any other instance other than what is intended. Instance traffic to other instances is treated the same as public internet traffic
* Customer instances have no access to raw disk devices, but are presented instead with virtual disks
* AWS proprietary disk virtualization automatically resets each block of storage used by customers so that one customers data is never unintentionally exposed to another
* Memory allocated to guests is scrubbed or set to 0 by the hypervisor when it is unallocated from a guest
* Unallocated memory is NEVER returned to the pool of free memory until the memory scrubbing process is complete
* AWS Service compliance
  + SOC 1/SSAE 16/ISAE 3402 (formerly SAS 70 Type II)
  + SOC2
  + SOC3
  + FISMA
  + DIACAP
  + FedRAMP
  + PCI DSS Level 1
  + ISO 27001
  + ISO 9001
  + ITAR
  + FIPS 140-2
  + HIPPA
  + Cloud Security Alliance (CSA)
  + Motion Picture Association of America (MPAA)
* AWS provides their annual certifications and compliance reports
* User Responsibilities:
* Anything that is put on the cloud or connects to the cloud
* IAAS (Infrastructure as a service) components require the user to perform all security configuration and management tasks
  + EC2
  + VPC
  + S3
* Account management and user access
* MFA implementation
* Communication to services using SSL/TLS
* Logging of API/User activity via CloudTrail
* Protecting data transmission via HTTPS using SSL
* Obtaining permission from AWS to perform penetration testing and or port scanning against your AWS Nodes
* All vulnerability scans, port scans, and penetration testing requests MUST be submitted in advance and approved by AWS
* When requesting and granted permission for port scanning, scans must be limited to your own instances
* Unauthorized port scans are a violation of the AWS Acceptable Use Policy
* Checking Trusted Advisor (TA) recommendations for potential cost savings, system performance improvements, and potential security gaps
* TA can provide alerts on security misconfiguration, such as open ports, public access to S3 buckets, user logging activities, lack of MFA, and more
* Guest operating system on non-managed services such as EC2 are under the full control of the user
* AWS does NOT have any login or access rights to your instances guest operating system
* Configuration of EC2 firewall. The inbound firewall configured on each EC2 instances is set by default in deny-all mode
* Users are fully responsible for explicitly opening the ports needed to allow inbound traffic to their instances
* User is responsible for using AWS's provided encryption options to encrypt EBS volumes and their snapshots with AES-256 bit encryption
  + Encryption occurs on the servers that host the EC2 instances and EBS storage
  + EBS Encryption is only available on EC2's bigger instance types such as the M, C, R, and G instance families
* SSL Termination on ELBs
* Anything put on AWS assets including the raw data compliance

**IAM Policies:**

* Each IAM Policy must contain the Resource property
* Policies consist of 3 main components, Action, Resource, and Effect
* Effect - Whether the policy allows or denies access
* Action – The list of actions that are allowed or denied by the policy
* Resource – The list of resources on which the actions can occur
* Condition (Optional) – The circumstances under which the policy grants permission
* Roles are more secure than programmatic access, and should always be used as the first resort where possible
* All IAM users should have MFA (Multi-Factor Authentication) enabled

{

"Version": "2012-10-17",

"Statement": {

"Effect": "Allow",

"Action": "s3:ListBucket",

"Resource": "arn:aws:s3:::example\_bucket"

}

}

This sample policy would allow a ListBucket Request to be performed on the example\_bucket S3 bucket for example.

**STS (Security Token Service):**

* Grants users limited and temporary access to AWS resources
* Users can come from 3 different sources:
* Federation (Active Directory):
  + Uses Security Assertion Markup Language (SAML)
  + Grants temporary access based off hte users AD credentials
  + Does not need to be an IAM user
  + Single sign on allows users to log into the AWS console without assigning IAM credentials
* Federation with Mobile Apps:
  + Use Facebook, Amazon, Google, or other OpenID providers to log in
* Cross Account Access:
  + Lets users from one AWS account access to resources in another AWS account
* Federation - Combining or joining a list of users in one domain with a list of users in another domain (Active Directory -> IAM for example)
* Identity Broker - A service that allows you to take an identity from Domain A and join it (federate it) to Domain B
* Identity Store - Services like Active Directory, Facebook, Google, Amazon, etc..
* Identities - A user of a service like Amazon, Facebook, Google, etc..
* Steps of Authentication:
* User enters username/password
* Application calls an Identity Broker. The broker is passed the username/password
* The Identity Broker uses the organizations centralized authentication to validate the identity of the user (Think Active Directory)
* The Identity Broker then calls the new GetFederationToken function using IAM credentials. The call must include an IAM policy and duration (1-36 hours), along with a policy that specifies the permissions to be granted to the temporary security credentials
* STS confirms that the policy of the user making the call gives permission to create new tokens and then returns 4 values
  + Access Key
  + Secret Access Key
  + Token
  + Duration of token
* Identity Broker returns the temporary security credentials to the requesting application
* The requesting application uses the temporary security credentials and token to make requests to Amazon
* Amazon uses IAM to verify that the credentials allow the requested operation on the given service using the given key
* IAM provides the service with a allowed action to perform the requested operation
* Steps in Simplicity:
* Develop an Identity Broker to communicate with LDAP and AWS STS
* Identity Broker should always authenticate with LDAP first, then the STS service
* Application gets temporary access to AWS resources

Route53:

**DNS:**

* DNS or Domain Name System is used to convert human friendly domain names into IP addresses
* 2 types of IP addresses:
* IPv4
  + 32 bit address
  + 4 billion different addresses (4,294,967,296)
* IPv6
  + Created to solve depletion issue of IPv4 address space
  + 128 bit address
  + 340 undecillion addresses (340,282,366,920,938,463,463,374,607,431,768,211,456)
* Top Level Domains: Signified by the last word in a domain name
* .com
* .edu
* .gov
* .net
* .io
* etc
* Controlled by the Internet Assigned Numbers Authority (IANA)
* Stored in a root zone database which is a database of all available TLDs (Top Level Domains)
* Database can be found at <http://www.iana.org/domains/root/db>
* Domain Names:
* All names in a given domain name have to be unique
* DNS registrars are authority's that can assign domain names directly under one or more TLD's
* Domains are registered with InterNIC, as service of ICANN, which enforces uniqueness of domain names across the internet
* Each domain name becomes registered in a central database known as the WhoIS database
* Popular domain registrars include godaddy.com, namecheap.com, Route53 etc..
* SOA (Start of Authority) Records store information about:
* Name of the server that supplied the data for the zone
* Administrator of the zone
* Current version of the data file
* Number of seconds a secondary name server should wait before checking for updates
* Number of seconds a secondary name server should wait before retrying a failed domain transfer
* Maximum number of seconds that a secondary name server can use data before it must either be refreshed or expired
* Default number of seconds for the TTL (Time to Live) on resource records
* DNS Record Types:
* NS or Name Server Records are used by TLD's to direct traffic to the content DNS server which contains the authoritative DNS records
* A or Address records are used by a computer to translate the name of the domain to an IP address
* CNAMES or Canonical Names can be used to resolve one domain name to another
  + CNAME's can't be used for naked domain names (zone apex). As such awsdocs.com must be either an A record or an Alias record
* Alias records are used to map resource record sets in your hosted zone to ELBs, CloudFront Distributions, or S3 Buckets that are configured as websites
  + Alias records work like CNAME records in that you can map one DNS name to another target DNS name
  + Alias records can save time because Route53 automatically recognizes changes in the record set that the alias resource record set refers to
  + You are NOT charged for requests to Alias records, you ARE charged for requests to CNAMES, so using Alias records is cheaper
* TTL or Time to Live is the length that a DNS record is cached on either the resolving server or the users local PC. The lower the TTL, the faster changes to DNS records take to propagate throughout the internet
* ELBs do not have a pre-defined IPv4 address, DNS names are used for ELB resolution
* Understand the difference between an Alias Record and a CNAME
* Always use an Alias record over a CNAME where possible, as it's cheaper and faster

**Route53 Routing Policies:**

* Simple
* Default routing policy when you create a new record set
* Most commonly used when you have a single resource that performs a given function for your domain
* Example would be a single web server that serves content for a single domain name
* Weighted
* Use to route traffic to multiple resources in proportions that you specify
* Split traffic based on different weights assigned within the record set
* Example would be sending 10% of user traffic to US-East-1 and the other 90% to US-East-2
* Latency
* Use when you have resources in multiple locations and you want to route traffic to the resource that provides the least latency
* Route traffic based on lowest network latency for your end users, such as sending requests to the region that will give the user the fasted response time
* Create a resource record set for EC2 or ELB resources in each region that hosts your content. When Route53 receives a request for your content, it selects the latency resource record for the region that gives the user the lowest latency
* Failover
* Use when you want to configure active-passive failover
* Example would be when you want your primary site to be in US-East-1, and a DR site in US-West-1
* Route53 will monitor the health of our primary site using a health check
* Health checks are not automatic and must be configured by the user
* GeoLocation
* Use when you want to route traffic based on the location of your users
* Example would be ensuring that EU customers get routed to servers residing in the EU, and ensuring US customers get routed to servers residing in the US

VPCs and Direct Connect:

**VPC's:**

Lets you provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network that you define. You have complete control over your virtual networking, IP ranges, creation of subnets and configuration of route tables and network gateways.

* Virtual data center in the cloud
* Allowed up to 5 VPCs in each AWS region by default. This limit can be increased with a support ticket request
* All subnets in default VPC have an Internet gateway attached
* Multiple IGW's can be created, but only a single IGW can be attached to a VPC.. No exceptions
* Again, You can only have 1 Internet gateway per VPC
* Each EC2 instance has both a public and private IP address
* If you delete the default VPC, the only way to get it back is to submit a support ticket
* This answer is correct for the current iteration of tests, however AWS has now crated a mechanism in the console that allows you to recreate a default VPC
* By default when you create a VPC, a default main routing table automatically gets created as well.
* Subnets are always mapped to a single AZ
* Subnets can not be mapped to multiple AZ's
* /16 is the largest CIDR block available when provisioning an IP space for a VPC
* /28 is the smallest CIDR block available when provisioning an IP space for a VPC
* Amazon uses 3 of the available IP addresses in a newly created subnet
  + x.x.x.0 - Always subnet network address and is never usable
  + x.x.x.1 - Reserved by AWS for the VPC router
  + x.x.x.2 - Reserved by AWS for subnet DNS
  + x.x.x.3 - Reserved by AWS for future use
  + x.x.x.255 - Always subnet broadcast address and is never usable.
* 169.254.169.253 - Amazon DNS
* By default all traffic between subnets is allowed
* By default not all subnets have access to the Internet. Either an Internet Gateway or NAT gateway is required for private subnets
* A security group can stretch across different AZ's
* Security Groups are stateful (Don't need to open inbound and outbound, if inbound is allowed, outbound is auto allowed)
* Network Access Control Lists (NACLs) are stateless (Must define both inbound and outbound rules)
* You can also create Hardware Virtual Private Network (VPN) connection between your corporate data center and your VPC and leverage the AWS cloud as an extension of your corporate data center
* VPC Flow Logs:
* VPC Flow Logs is a feature that enables the user to capture information about the IP traffic going to and from network interfaces in your VPC
* Flow log data is stored using Cloudwatch Logs
* When Flow log data is collected it can be viewed and its data can be retrieved within Cloudwatch
* Flow logs can be created at 3 different levels, VPC, Subnet and Network Interface levels
* Flow logs via Cloudwatch can be configured to stream to services such as Elasticache, or Lambda
* You cannot enable flow logs for VPC's that are peered with your VPC unless the peer VPC is in your account
* You cannot tag a flow log
* After you have created a flow log, you cannot change its configuration, for example you cannot associate a different role with the flow log
* Not all traffic is monitored:
  + Traffic generated by instances when they contact Route53 is not monitored or logged
  + If you use your own DNS server, then all traffic to that DNS server is logged
  + Traffic generated by a Windows instance for Windows license activation is not monitored or logged
  + Traffic to and from the metadata service (169.254.169.254) is not monitored or logged
  + DHCP traffic is not monitored or logged
  + Traffic to the reserved IP address for the default VPC router is not monitored or logged
* Network Address Translation (NAT) Instances:
  + When creating a NAT instance, disable Source/Destination checks on the instance or you could encounter issues
  + NAT instances must be in a public subnet
  + There must be a route out of the private subnet to the NAT instance in order for it to work
  + The amount of traffic that NAT instances support depend on the size of the NAT instance. If bottlenecked, increase the instance size
  + If you are experiencing any sort of bottleneck issues with a NAT instance, then increase the instance size
  + HA can be achieved by using Auto-scaling groups, or multiple subnets in different AZ's with a scripted fail-over procedure
  + NAT instances are always behind a security group
* Network Address Translation (NAT) Gateway:
  + NAT Gateways scale automatically up to 10Gbps
  + There is no need to patch NAT gateways as the AMI is handled by AWS
  + NAT gateways are automatically assigned a public IP address
  + When a new NAT gateway has been created, remember to update your route table
  + No need to assign a security group, NAT gateways are not associated with security groups
  + Preferred in the Enterprise
  + No need to disable Source/Destination checks
  + More secure than a NAT instance
* Network Access Control Lists (NACLS):
  + NACL's are stateless, meaning both inbound and outbound rules must be configured for traditional request/response model
  + Numbered list of rules that are evaluated in order starting at the lowest numbered rule first to determine what traffic is allowed in or out depending on what subnet is associated with the rule
  + The highest rule number is 32766
  + Start with rules starting at 100 so you can insert rules if needed
  + NACL's have separate inbound and outbound rules, and each rule can either allow or deny traffic
  + The Default NACL will allow ALL traffic in and out by default
  + Custom NACL's by default will deny all inbound and outbound traffic until allow rules are added
  + You must assign a NACL to each subnet, if a subnet is not associated with a NACL, it will allow no traffic in or out
  + NACL rules are stateless, established in does not create outbound rule automatically
  + You can only assign a single subnet to a single NACL
  + When you associate a NACL with a subnet, any previous associations are removed
  + You can associate a single NACL with multiple subnets
  + Each subnet in your VPC must be associated with a NACL. If you don't explicitly associate a subnet with an ACL, the subnet automatically gets associated with the default ACL
  + You can block IP addresses using NACLs not Security Groups
* VPC Peering:
  + Connection between two VPCs that enables you to route traffic between them using private IP addresses via a direct network route
  + Instances in either VPC can communicate with each other as if they are within the same network
  + You can create VPC peering connections between your own VPCs or with a VPC in another account within a SINGLE REGION
  + AWS uses existing infrastructure of a VPC to create a VPC peering connection. It is not a gateway nor a VPN, and does not rely on separate hardware
  + There is NO single point of failure for communication nor any bandwidth bottleneck
  + There is no transitive peering between VPC peers (Can't go through 1 VPC to get to another)
  + Hub and spoke configuration model (1 to 1)
  + Be mindful of IPs in each VPC, if multiple VPCs have the same IP blocks, they will not be able to communicate
  + You can peer VPC's with other AWS accounts as well as with other VPCs in the same account
* VPC Endpoints:
* Allows internal resources such as EC2 instances to reach various AWS services without having to traverse the public internet to get to the service
* When you use an endpoint, the source IP address from your instances in your affected subnets for access the AWS service in the same region will use private IP address's instead of public IP address's
* When configuring VPC endpoints, existing connections from your affected subnets to the AWS service that use public IP address's may be dropped

**Direct Connect (DX):**

* DX or Direct Connect makes it easy to establish a dedicated network connection from your premises to AWS
* Using DX, you can establish private connectivity between AWs and your data center, office or collocation environment
* Requires a dedicated line such as MPLS, or other circuit ran from tel-co.
* From this line, you would have a cross connect from your on-premises device direct to AWS data centers
* Using DX, can reduce network costs, increase bandwidth throughput and provide a more consistent network experience then internet based connections
* Lets you establish a dedicated network connection between your network and one of the AWS DX locations
* Uses industry standard 802.1Q VLANs
* Dedicated connections can be partitioned into multiple virtual interfaces
* Same connection can be used to access public resources such as objects stored in S3 using public IP's and private resources such as EC2 instances running in a VPC using private IP's, all while maintaining network separation between the public and private environments
* Virtual interfaces can be reconfigured at any time to meet changing needs
* Offers more bandwidth and a more consistent network experience over using VPN based solutions
* VPC VPN connections utilize IPSec to establish encrypted network connectivity between your intranet and your AWS VPC over the internet
* VPN connections can be configured in minutes and are a good solution if you have an immediate need
* DX does NOT involve the internet, instead, it uses dedicated private network connections between your intranet and AWS VPC