**Other AWS Services**

**Polly:** a service to convert text to speech (mp3)

**Alexa:** Amazon voice service

**Cloudformation:** is a service that helps you model and set up your Amazon Web Services resources (EC2, DBs) using templates so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. It is something like Dockerfile!

**IAM (identity access management)**

* Manage users and their level of access to AWS console.
* Grant shared access to aws console
* Granular permissions
* Using Active Directory/Facebook/Linkedin to login
* Multifactor authentication (MFA - device or virtual based)
* Provide temporary access for users/devices
* Support PCI DSS compliance
* IAM is set on global region
* AWS access types: **programmatic access** and **management console**
* **AdministratorAccess** policy is as powerful as root user.
* Using AWS command line needs **Secret access key** not the user’s password
* IAM’s roles define how AWS’s services connect together
* To set Billing alarm:
  + 1. Enable receive billing alerts in preferences
  + 2. Cloudwatch -> Billig -> Create Alarm
* IAM consists of: Users, Groups, Roles, Policy Documents (json documents)
* New created users in IAM have NO permissions but they are assigned access key and secret keys

**S3**

## S3:

* object based
* file size limit 0 to 5TB
* bucket name is unique globally
* Links are like https://s3-eu-west-2.amazonaws.com/bucketname/filename.jpg
* read & write consistency for new objects (immediate effect)
* eventual consistency for update & delete (take time)
* objects are stored as key value; key is name value is data
* Other keys: version id, metadata, sub resources (access control list, torrent)
* availability: built for 99.99 guarantee 99.9
* durability: guarantee 99.999999999 (11\*9)
* ssl support
* versioning
* lifecycle management
* designed to sustain loss of 2 facilities
* buckets are private by default
* s3 can be configured to have access log stored in a bucket or another aws acc
* s3 can be used to host static website
* S3 Transfer Acceleration helps to transfer faster, will upload to s3 edge, then will sync using aws backbone network
* Write to S3 - HTTP 200 code for successful write
* Can upload file to s3 faster by enabling multipart upload
* <https://aws.amazon.com/s3/faqs/>

## S3 IA (infrequently access)

* rapid access
* lower fee than normal s3
* charge per retrieval
* less availability , 99.9 compare to 99.99
* min object size 128KB

## S3 RRS ( reduced redundancy storage)

* less durability compared to S3 99.99 < 99.999999999
* same availability with normal S3
* Less concurrent facility , (1 compare to 2)
* good for stuff like thumbnails that can be generated again

## Glacier

* very cheap
* takes 3-5 hour to retrieve
* $0.01 per gigabyte

## \*AWS Import/Export

* It accelerates moving large files into and out of AWS using portable storage devices for transpor
* It uses AWS internal network to bypass internet for high speed
* It is faster and more cost effective than upgrading network connectivity
* Pricing models:
  + Per-device fee
  + A data load time charge (per data-loading-hour)
  + Possible return shipping charges

**Versioning**

* Stores all versions (write & delete)
* Versioning is a great backup tool
* once enable, can't remove versioning, only suspend
* can use MFA for deletes
* can apply lifecycle rule

**Cross Region Replication**

* must have versioning enabled on both src & dst buckets to enable CRR
* two buckets in one region is not allowed for CRR
* existing objects in src buckets won’t replicate right after enabling. only will replicate after next update
* By default permissions will be copied from one region to the other one
* after 1st update it will replicate all previous versions of an object
* s3 does NOT support "multiple" region replication (chain or clustered); only from 1 region to another
* delete from src -> delete from dst (automatically created delete markers are replicated to the other side region)
* deleting individual versions or delete markers manually will not be replicated
* restore from a backup from versioning from src -> NOT restore on dst

## 

## **Lifecycle Management**

“Lifecycle rules manage cost by archive and moving objects to S3IA or Glacier and then remove them after some time. Lifecycle can manage all new and old version of objects”

* LC can apply to the whole bucket, or with a prefix
* LC can apply on current version, previous versions & incomplete multipart uploads
* LC is not based on versioning, without versioning is also possible to enable LC
* s3 -> s3 IA -> Glacier -> Expire (permanently delete)
* [s3 IA] only object >128Kb & 30days after creation date
* [Glacier] after 30 days after s3IA; objects will be charged for min 90 days because Glacier is designed to keep object for at least 90 days.

**CloudFront - CDN**

“It is a service that users get files from the nearest datacenter (edge locations)”

**Terminologies:**

1. **Edge locations:** where content are cached; edge locations are not AWS regions and AZ
2. **Origin:** src of files, can be S3 buckets, EC2, ELB (elastic load balancer), Route53, non-aws origin files (on-premises files) are also able to work with cloudfront
3. **Distribution:** a name give to CDN consisting a collection of edge locations
4. **Web distribution:** typically used for websites

* Cloudfront can be used for entire website including dynamic/static/streams/interactive contents
* objects cached for TTL or default cached by 24 hours
* edge locations can have PUT as well as READ (they’re not readonly)
* cache can be cleared, but will be charged

**Security & Encryption**

* buckets are private by default
* Setup access control list with **bucket policies** can change which applies to whole bucket
* Setup access control list with **Access Control List** can change, which can applies to objects in bucket too
* can set access logs to see who decrypt which file & when
* **In transit encryption:** for upload objects to s3 is SSl/TLS or https
* **Server side encryption (3 ways):**
  + S3 Managed Keys (SSE-S3)
  + Aws key management service, Managed keys (SSE-KMS),
  + Server side encryption with customer provided keys (SSE-C)
* **Client side encryption:** encrypt first on client system then upload object

**Snowball**

* for migration purposes
* old way: import/export disk service: to send over an external hdd to aws for transferring data
* **standard snowball:** onboard storage, petabyte scale, have multiple layer of security, can be track where it is, aws erase it clean after transfer
* **snowball edge:** onboard storage+compute capacity like small aws cloud,
* **snowmobile:** it's a truck, for exabyte of data transfer

**Storage Gateway**

“AWS Storage Gateway is a storage service that enables your on-premises applications to seamlessly use storage in the AWS Cloud. This service also enables you to securely store data to the AWS cloud for scalable and cost-effective storage”

* It is either installed on the local machine in the datacenter using VMWare ESXi/Hyper-v or it is also able to install on EC2 instance
* Four different types of storage gateways
  + **File Gateway** : use NFS mount points, (non-executable files, flat files only, pdf, images, video, etc) stored directly on S3, can use versioning, lifecycle management and cross-region replication
  + **Volume Gateway**: use iSCSi, it’s like virtual disks used to store data in block based storage (data such as Os, DB, etc), written data can be asynchronously backuped to aws, good for unreliable internet connections.
    - **Stored Volumes**: entire dataset is stored locally on site & is async backed up to S3. this provides low-latency access to the data while provides off-site backup
    - **Cached Volumes**: Entire dataset is stored on S3 & most frequently accessed data is cached on site. It leads to need less local storage while still provide low-latency for local applications
  + **Tape Gateway (VTL):** virtual tapes, use for backup and uses popular backup applications eg NetBackup, Backup Exec, Veam, etc.
* \* pricing components:
  + Gateway usage (per gateway per month)
  + Snapshot storage usage (per GB per month)
  + Volume storage usage (per GB per month)
  + Data transfer out (per GB per month)

**S3 transfer acceleration**

“S3 transfer acceleration use Cloudfront edge network to accelerate upload files to s3 instead of directly pushing to s3 bucket”

* Can use a distinct url to upload to and edge location
* Url is like this: **bucket-name.s3-accelerate.amazonaws.com**

**Static website using S3**

* Url is like this: bucket-bane.s3-website.eu-west-2.amazonaws.com
* Can be used without worrying about load balancing/server load and …
* Serverless web service while it scale automatically, very cheap
* It can’t be used for dynamic files like PHP/.net

**EC2**

“Amazon Elastic Compute Cloud is a service that provides resizable compute capacity in the cloud. It allows quick scale capacity to fit in requirements”

**EC2 pricing options:**

## on demand:

* + fixed rate by hour, low cost, flexible, no payment up-front, no long-term, no commitment
  + applications that can't be interrupted or being test & dev
  + Good for app that has unpredictable workloads that can’t be terminated

## reserved:

* + Reserved capacity
  + discount, for 1-3 year reserve
  + Good for app that’ve got steady state and predictable usage
  + Applications that require reserved capacity
  + up-front payment is possible for discount

## spot:

* + bit how much you want.
  + if bit price>market price (spot) let you use.
  + Good for applications that’ve got flexible start/end time
  + Applications that need urgent large amount computing capacity
  + If the spot instance is terminated by EC2, user will not be charged for a partial hour of usage. But, if user terminate the instance, user will be charged for any hour in which the instance ran. (if user terminate the instance, you pay for the hour. If aws terminates, it is free)

## dedicated host:

* + physical ec2 payable by hour
  + useful for regulatory requirement that virtualisation is not possible (Governments)
  + great for licensing apps that doesn’t allow to be on shared cloud
  + 1-3 years reserve get discount

## EC2 Instance Types:

Acronym: **DR MC GIFT PX**

* + D2: density: (storage optimized, file server, data warehouse, hadoop)
  + R4: Ram (memory optimized)
  + M4: Main (general purpose)
  + C4: cpu (compute)
  + G2: Graphic(3d, video encoding..)
  + I2: iops (high speed storage, nosql db, db warehousing)
  + F1: FPGA: change hardware, hardware acceleration for code
  + T2: low cost general purpose (web server, small db)
  + P2: Picture/graphic (Machine learning bitcoin mining)
  + X1: extreme memory optimized (apache spark)

## EBS (Elastic Block Storage)

* Attachable disks - block based storage
* Good for FileSystem, DB storage or install OS
* Automatically replicated on different components within 1 AZ (availability zone)
* Can’t mount 1 EBS to multiple EC2 instance
* Types:
  + **General purpose SSD (GP2)**
    - Balanced price & performance
    - Ratio 3 iops / GB
    - Up to 10K iops
    - Good for root volume
  + **Provisioned IOPS SSD (IO1)**
    - For i/o intensive apps like large DB or NoSQL db
    - Use only if more 10,000 iops is required
    - Up to 20K iops / volume
  + **Throughput Optimized HDD (ST1)** 
    - Magnetic disk
    - Big data
    - Data warehouse
    - Log processing
    - Can NOT be a boot volume
    - Good for sequential writes
  + **Cold HDD (SC1)**
    - Magnetic disk
    - Low cost for infrequently accessed workload
    - File server
    - Can NOT be boot vol
  + **Magnetic HDD (Standard)**
    - The only magnetic storage that can be used for boot vol
    - Lowest cost per gig (cheapest) of all bootable EBS
    - Ideal for infrequent access & application with low cost storage

## ETC…

* By default Termination protection for EC2 is off
* By default if EC2 instance terminates, EBS vol will get deleted as well
* By default root volume can not be encrypted, but there are some ways to encrypt that, but not directly though
* Additional Vols can be encrypted
* The ssh private key has to have chmod 400
* Ssh ec2-user@ip -i certificate.pem
* Status check
  + System status check: to check if aws infrastructure is reachable in terms of network and underlying hypervisor
  + Instance status check: to check if instance accepting network traffic
* puttykeygen: convert pem file to ppk files

## Security Group (SG)

* It is a virtual firewall in aws
* 1 instance can have multiple security groups
* controls traffic in/out to instances
* All inbound traffic is blocked by default, we just open everything we need (only allow rules are available no deny option)
* All outbound traffic is allowed by default
* Can attach 1 SG to multiple EC2 and vice versa
* Any modification to SG applies immediately
* SG is **STATEFUL**: Whatever rules exists for inbound, will be considered as outbound as well, does not need to add specifically; for eg: opening port 80 for inbound allows for outbound as well.
* Can’t block/deny IP using SG; need to use Network Access Control List
* We can specify allow rules, but no deny rules with SG

## Upgrading EBS Volume types - HDD to SSD and visa versa

“If we want to upgrade an EBS volume, e.g, magnetic to SSD: first we need to take a snapshot from the magnetic volume, then from that snapshot we are able to create a new volume with different type”

* EBS vol can be changed on the fly (except for magnetic standard)
* It is recommended to stop the instance before taking snapshot for consistency
* Change volume type: snapshop -> create a new vol
* If vol is changed on the fly need to wait 6h for a new change
* EBS vol can only scale up
* Vol and instances should be in the same AZ

**Commands to attach a volume to a Linux machine**

**#**lsblk

**#**mkfs -t ext4 /dev/xvdb #wipe the block storage

**#**mount /dev/xvdb /mountdirectory

#file -s /dev/xvdb #to check if the volume is empty or no, if it prints “data” only, the vol is empty

## Volume & Snapshots (SS) & RAID

* Vol should be in same AZ with EC2 to be able to attach
* RAID:
  + RAID 0
    - Striped disks together
    - No redundancy
    - Good performance
    - If one disk fails, all data is gone
  + RAID 1 : exact copy
    - Mirrored
    - Redundancy
    - Capacity is half of the total disks
  + RAID 5
    - 3 disk or more
    - Good for reads
    - Bad for write
    - Recover lost disk from checksum
    - Aws does NOT recommend RAID 5 on EBS.
  + RAID 10: mix raid 0 + 1
    - stripted
    - Mirrored
    - Good redundancy
    - Good performance
* Most of the time is raid 0 & 10 on aws
* Raid is good for getting more disk I/O than max
* Taking snapshot of raid is problematic (not storing cache); can be solved by either:
  + Freeze File system
  + Unmount RAID, take SS, mount again
  + Shutdown EC2, take SS, turn on
* Restore EC2 from encrypted SS will make root device of EC2 to be encrypted
* Snapshot of an encrypted Vol are encrypted automatically & vice versa (as above)
* Snapshots can be shared with other AWS acc / public/ sell on store; only if are not encrypted
* SS stores on s3
* SS can be copied to different regions
* SS are incremental: only changes compare to last snapshot are moved to s3
* SS excludes data in cache of apps & os

## Amazon Machine Image (AMI)

“AMIs are snapshots from root volume of EC2 instances”

* AMI can be choosed from Region/AZ; OS; arch (32/64); launch permissions; storage for root
* Root device type of AMI images are either **EBS** or **instance store**
  + For EBS: root device is created from EBS snapshot/images
  + *EBS backed volumes are persistent, means they are independently continue until the time of EC2 instance*
  + *EBS volumes can be detached and reattached to other EC2 instances*
  + *Instance store backed volumes are NOT persistent (ephemeral), means can NOT detach from one EC2 instance and attach to another, they exist only for the life of that instance*
  + *For Instance Store: root device is created from template in S3, so it doesn’t have any volume in EBS page*
  + *EBS volumes can be stopped; data will persist; used for store data for long term*
  + *Instance store volumes can NOT be stopped; if we stop, data will be wiped; shouldn’t be used for long-term data storage*
* Instance store volumes also called Ephemeral storage
* Instances created from Instance store type can NOT be stopped; if underlying host fails, you will lose data
* ^^ not true for EBS vols
* Reboot both is possible; data won’t get lost
* By default both ROOT volumes will be deleted on termination, however we can ask aws to keep root device for EBS vol only
* AMI’s are regional, however we can copy one AMI from one region to another one using console, CLI or EC2 API

## Elastic Load Balancer (ELB)

* Two types of load balancers:
  + **Application load balancer:**
    - Works on L7 of OSI Model (application layer)
    - Prefered for HTTP/HTTPS
    - It uses a clever routing application load balancer
    - Can route traffic to one or more instances on EC2 or VPC
  + **Classic load balancer:**
    - Works on L4 (transport layer) - TCP/SSL
    - It is good for database load sharing
    - By default, a Classic Load Balancer routes each request independently to the registered instance with the smallest load. However, you can use the **sticky session** feature, which enables the load balancer to bind a user's session to a specific instance. This ensures that all requests from the user during the session are sent to the same instance:
      * Duration-Based Session Stickiness
      * Application-Controlled Session Stickiness
* ELB is integrated with health check, if health check fails, status of ELB will be changed to **OutOfService** or **InService**
* Each subnet is equal to a availability zone
* ELB have their own dns, aws never give an IP for ELB
* The following versions of the SSL protocol are supported:
  + TLS 1.2
  + TLS 1.1
  + TLS 1.0
  + SSL 3.0

## CloudWatch

* Default time interval for standard monitoring = 5min while detail monitoring = 1min
* **Cloudwatch dashboard:** We can create customized dashboard with different widgets
* Default per instance metrics in Cloudwatch
  + CPU related metrics
  + Disk related metrics
  + Network related metrics
  + Status check (instance level and host level)
* RAM usage is NOT in default metrics, if required, need to enable in custome view
* **Cloudwatch alarms:** We can create alarms based on metrics when particular threshold are hit
* **Cloudwatch events:** response to state changes in AWS resources. E.g, calling a lambda function to change a DNS record.
* **Cloudwatch logs:** aggregate, monitor, store log files. monitor EC2 instances in application layer like http response codes, kernel logs and other log files

**Exam tip:**

Cloudtrail vs Cloudwatch:

**Cloudwatch**: is for performance monitoring, likes CPU Utilization

**CloudTrail:** is for auditing, monitor entire cloud like created users, created roles, what people do with aws account.

## AWS CLI Tools

* Aws CLI is preinstalled on Amazon AMI
* It needs a user with programmatic access (access key + secret key)

|  |
| --- |
| aws configure //set access key & secret key  aws s3 ls //list all s3 buckets aws [service] help vi ~/.aws/credentials //location of keys aws ec2 describe-instances //describe all instances **in** EC2 aws ec2 terminate-instances --instance\_ID //terminate an instance aws s3 cp --recursive s3://bucketname /home/user/downloads aws s3 cp --recursive s3://bucketname /home/user/downloads --region region\_name //it is good to use region flag because cli does not let us to work with other regions  aws s3 sync // it will sync (not copy as duplicate) contents of a directory  Aws s3 sync --delete //it delete if one file in src location was deleted |

## IAM Roles

Using access & secret key for aws cli tools raise a security issue. Hence the credentials are stored as plain text files, they may be compromised or be hacked. In other case, we may need to change the credentials over 100 servers. The solution is to use IAM Roles and define that role to the EC2 instances to have direct access to the resources without needing to set the keys manually.

It is also now available to assign IAM Roles to an existing EC2 instance.

* Roles are more secure than keys
* Roles are much easier to manage
* Roles can be assigned to instances AFTER it has been created by command line or aws console
* Changing to an assigned role will take immediate effect

## Bootstrap scripts

* Bash scripts to run when instance is created
* It should be start with shebang. eg. #!/bin/bash
* Should be defined when instance is created -> advanced details -> user data

Example:

|  |
| --- |
| #!/bin/bash  yum update -y yum install httpd -y service httpd start  chkconfig httpd on  aws s3 cp s3://mybucket/index.html /var/www/html |

## Instance Metadata

* Used to get information about an instance, like getting public/private ip addresses, CPU cores, RAM and …
* Curl <http://169.254.169.254/latest/meta-data> //it should be ran inside that instance
* *Key things to remember is that it’s an instance's Meta Data, not user data*

## Launch configuration and Auto Scaling groups

* Before creating auto scaling group we should create launch configuration
* To have high availability, it is better to set auto scaling group in different AZ
* It can remove or add new instances automatically by using alarms (eg. CPU Usage > 90%)

## EC2 Placement Group

* It is a group of instance in one AZ that got 10Gb network
* It is recommended for low latency needed and high throughput apps like Grid Computing
* It can NOT span in different AZ
* Only certain types of instances are allowed for PG like:
  + Compute optimized
  + GPU
  + Memory optimized
  + Storage optimized
* AWS recommend homogenous (instance with similar CPU/RAM/Disk size) instance in PG
* We can’t merge a placement group
* Can’t move an existing instance to a PG. can create AMI and the launch in a PG.

## EFS (Amazon Elastic File System)

“It is a file storage used for EC2 instances”

* Using EFS storage capacity is elastic, growing and shrinking is automatic
* Can mount one EFS to two or more instances what EBS won’t allow
* Support Network File System (NFSv4)
* Only pay for capacity that we use - no need to define the size
* Data is stored in multiple AZ within a region
* Read after write consistency
* Support thousands of concurrent NFS connections
* EFS is block storage not object storage
* Instances and EFS should be on the same Security Group?!
* We can use EFS and mount it to several instances as /var/www/ mount point to create a robust high availability system with similar files as centralized repository
* It is also possible to restrict a user to a file or directory

## Lambda

“It is a compute service that we can upload our code or have lambda functions triggered by some events. In this case we shouldn’t be worried about OS, scaling, …”

* Usage:
  + Use as **event-driven compute service**: AWS lambda will run when an event happens. Like uploading file to S3 or inserting a record to amazon database.
  + Use as **compute server:**  upload only codes and get the response by calling Amazon API or AWS SDKs
* Supported languages: Nodejs, Python, Java, C#
* Pricing:
  + **Number of requests** ( first million requests are free)
  + **Duration** needed for code execution (amount of RAM allocated)
* No server is required.
* Continues scaling and it is very cheap
* To invoke a lambda function from different domain or other hosts using AWS API Gateway we need to enable **CORS** on API Gateway

**Route 53**

## DNS:

* The last part in a domain name is top-level-domain (.com, .net) that are hold by IANA
* **SOA Record (Source of Authority)**
  + The name of the server that supplied the data for the zone
  + The administrator of the zone (owner’s email address)
  + Current version of data
  + Some interval time for checking for updates, failures, refreshes or expires
  + Default number of TTL on records
* **NS Record**
  + Name server record to direct traffic to the DNS server that has DNS records
* **A Record (address record)**
  + Used to resolve name to IP
* **TTL Record**
  + The time that a record is allowed to be cached. The lower TTL, the faster update propagate
* **CNAME Record (canonical record)**
  + To resolve one domain name to other. Eg, m.site.com and mobile.site.com
* **PTR Record**
  + To point an IP to a name (reverse of A record)
* **Alias Record**
  + Only aws has this record type
  + Records to map resources like ELB or S3 to a domain name
  + Difference between CNAME and Alias is that a CNAME can’t be used for empty (naked, @, or nothing.site.com) domain names. It should be an IP which ELB’s don’t have any IP addresses so they have to be used using DNS names.
  + It is recommended to always choose an Alias record over CNAME in the exam!
  + Alias record is free of charge.

## Route53 Routing Policies

* Simple
  + The default routing policy when a new record is created
  + It is the most common policy used when only one server or resource is available
  + Pointing a.site.com to one IP address only
* Weighted
  + It let us to split traffic based on different weights assigned.
  + Eg, 10% of traffic to one instance and 90% to another instance
  + Weight value 0 - 255. Setting 0 will disable weights
* Latency
  + Allows to route traffic based on lowest network latency (which region response faster)
  + It selects the latency resource record that gives the lowest response time to a user
* Failover
  + It is used when a passive/active (primary/secondary) set-up is required.
  + Route53 will monitor the health of the primary site using a health check
  + It is good to detect a failover
  + The primary dns record requires a health check
* Geolocation
  + Let us to choose from which geographic location the traffic should be sent to the users
  + For example, to route Europe traffic to an instance configured with European Language or prices displayed

**Database on AWS**

* **RDS:** Relational database service
  + SQL Server (1433 TCP)
  + Oracle
  + Mysql Server (3306 TCP)
    - maximum number of read replicas can you have for your MySQL database: **5**
  + PostgresSQL
  + Aurora (Amazon proprietary database)
  + MariaDB
* **DynamoDB:** NoSQL database
* **ElasticCache:** In-Memory cache
  + It can improve the performance of web-apps by retrieving frequent data from fast, managed in memory cache
  + it caches the most consistent query from the database, eg. top 10 product
  + ElasticCache supports two caching engine
    - Memcached
    - Redis
* **Redshift:** fast, simple, cost-effective data warehousing
  + Used for business intelligence
  + Used to do queries on large and complex data sets
    - **Online Transaction Processing (OLTP)** - pull a row from the data set (select \* from table where id=1)
    - **Online Analytic Processing (OLAP)** - pulls in large amount of data (sum of sales in one region)
  + Data warehousing database use both database and infrastructure to run managemen queries
  + Usually we make a copy of production database to Redshift to run OLAP query without affecting IOPS on the production database
* **DMS:** database migration service
  + It supports compression, parallel transfer for fast data migration
  + It can convert data set between different database types (Oracle to Mysql)

## RDS backup, Multi-AZ, Replica set

* **DB Backup:** AWS provides two ways for backup
  + **Automated backup:** allow to recover db to any time within “retention period = 1 to 35 day”
    - It is enabled by default
    - Backup are stored in S3 with the free storage same as the size of db
    - After db termination, all automatic backups will be gone
  + **Database snapshot:** it takes daily snapshot and also stores transaction logs for each day. In this case it is possible to recovery to any seconds by choosing latest snapshot and then applying the transactions.
    - They are done manually
    - They are stored even after db deletion
* **Encryption:**
  + it is supported for all types except Aurora
  + Encryption is done using AWS key management service (KMS)
  + If db is encrypted, all snapshots and backups are encrypted, too.
  + Encrypting existing db is not possible, can be done using new db and migration
* **Multi-AZ RDS:**
  + Allows to have an exact copy of production db in another AZ
  + AWS handles the replication automatically
  + In case of db failure or AZ failure, AWS will automatically failover to the standby db. In this case no need to change dns of db, aws changes the ip
  + To switch db to primary or secondary, no administrative interaction is required
  + Is only of **disaster recovery**. It is not used for improving performance. For **performance improvement** Read Replicas is required.
* **Read Replica:**
  + Allows to have read only copy of production database
  + This is asynchronous replication
  + read replica is read only,write is not allowed
  + It is used for very read-heavy (read intensive) databases
  + It is only used for scaling!!! Not used for disaster recovery
  + Automatic backups should be turned on for enabling read-replica
  + Can be up to 5 copy from main db
  + We can have read replicas of read replicas
  + Each replica will have its own DNS endpoint
  + Can NOT have read-replica that have multi-AZ
  + Can create read-replica of multi-az database
* **DynamoDB vs RDS scaling**
  + Dynamodb offers “push button” (automatic) scaling, means scale db without downtime
  + RDS’s scaling is not easy, need to change instance type that leads to downtime

## DynamoDB

* It supports both document and key-value data model
* Its flexible data model and reliable performance make it great for mobile, web, gaming apps
* Stored on SSD storage
* Spread across 3 data centers (built in redundancy)
* If our app can wait 1s after a write, eventual consistent is fine otherwise we have to choose strong consistent reads
* Eventual consistent read (default)
  + Consistency across all copies of data is within 1 second. Read after a write should return updated data (best read performance)
* Strong consistent reads
  + A strongly consistent read returns a results that reflects all writes that received a successful response prior the read
* Pricing
  + Need to calculate how many writes and read is required per seconds
  + It expensive for write
  + For throughput capacity
    - Write throughput $0.0065 per hour for every 10 units
    - Read throughput $0.0065 per hour for every 50 units
  + Storage cost $0.25Gb/month
* To update read capacity no need to change instance size or enable read replica, just need to change a number in dynamodb, without having downtime while we have down time for RDS

## Redshift

* It is a fast and powerful, fully managed, petabyte-scale data warehouse service in the cloud
* $0.25/h with no commitments or upfront cost
* It is used for OLAP transactions
* Currently only available in 1 AZ
* Configuration
  + **Single node**
  + **Multi-Node**
    - **Leader node** (manages connections, queries adn compute nodes)
    - **Compute node** (store data and perform computations), up to 128 nodes

* Columnar data storage:
  + 10 times faster cos data stores as columns instead of rows
  + Column-based systems require fewer I/O
  + Column-based data can be compressed much better than row-based
  + Aws automatically sample the data and select best compression scheme
* Massively Parallel Processing (MPP)
  + Redshift automatically distribute data and queries across all nodes.
* Pricing
  + Compute node hours
  + It won’t charge for leader nodes, only compute nodes will incur charges
  + It also charge for backup and data transfer only within a VPC not outside it
* Security
  + Encrypted in transit using SSL
  + Encrypted at rest using AWS-256
  + It supports key management

## ElasticCache

* Improves performance of web application by retrieving info from memory instead of disk
* Caching improves performance by storing critical data in memory for low-latency access
* Cached info are useful for read-heavy/computationally-intensive apps, gaming, question & answers apps
* Types of ElasticCache
  + **Memcached**
  + **Redis:** supports master/slave replication and multi-AZ

**Exam tips:**

* ElasticCache is good choice for read heavy db that not changing frequently
* Redshift is a good answer if db is under stress and load because of running OLAP transactions

**VPC (Virtual private cloud)**

## VPC Overview

* VPC is a logical datacenter
* It lets to create a logical isolated section in aws
* We can create a Hardware Virtual Private Network (VPN), a connection between corporate datacenter and VPN
* We can have multiple vpc in a region
* Private network ranges
  + 10.0.0.0 - 10.255.255.255 (10./8)
  + 172.16.0.0 - 172.31.255.255 (172.16/12)
  + 192.168.0.0 - 192.168.255.255 (192.168/16)
* **Public subnet** = internet access, **Private subnet** = no internet access
* **Very Important:** Each subnet is always map directly to a availability zone, 1 subnet = 1 AZ
* Can NOT span subnet across multiple AZ
* What can do with VPC
  + Launch instance to a deliberate subnet
  + Assign custom IP address
  + Configure route tables between subnets
  + Create internet gateway and attach to each VPC
  + Better security control
  + Instance security group
  + Subnet network access control lists (ACLS)
* Default vs custom VPC
  + All subnets in default vpc have a route to the internet
  + Each EC2 instance has both public and private IP address
  + If we delete default vpc, we need to contact aws to get it back
* VPC Peering
  + It is a connection between two VPCs that enable to route traffic between two network using private IP address
  + We can connect to other account’s VPC or other VPC in the same account
  + *VPC peering can be established only if both are within a* ***single region****.*
  + *VPC peering can be with multiple account but not multiple region.*
  + *AWS use the existing infrastructure for VPC peering. It is not a VPN or a gateway. So there is no single point of failure or bandwidth bottleneck.*
  + **Important:** Peering is in a star configuration, ie 1 central vpc peers with 4 others. **NO TRANSITIVE PEERING!!!.** It means that vpc B can not traverse through vpc A to access to VPC C. Traffic can be flown between VPC B and C using a direct peering between two of them.

**VPC B <-> VPC A <-> VPC C**

**(10.0.0.0/16) <-> (192.168.0.0/24) <-> (172.16.0.0/16)**

* When we create a VPC, a **Route Table**, **Network ACL** and a **Security Group** will be created automatically. It DOES NOT create **subnets** and **internet gateways**
* Only **1** internet gateways can be attached to custom VPC
* Network ACLs and Security Groups can span over multiple AZ but Subnets CANNOT
* **Reserved IP addresses** by aws in each subnet:
  + 10.0.0.0: Network address
  + 10.0.0.1: reserved for VPC router
  + 10.0.0.2: reserved by aws (DNS)
  + 10.0.0.3: reserved for future use
  + 10.0.0.255: network broadcast address
* /24 has only 251 available ip address
* Auto-assigning ip address is not enabled by default for each subnet
* *VPC Peering limitations:*
  + *Can’t create VPC between VPCs that have matching or overlapping CIDR; Two VPCs with subnet 10.0.0.0/16 and 10.0.0.0/24 won’t able to connect to each other because they have network address overlap*
  + Can’t create VPC peering connection between VPCs in different regions
  + VPC peering doesn’t support transitive peering

## NAT

* For giving access to the private subnets we have two options:
  + **NAT instance**
    - using NAT AMI and create instance from
    - **source/destination check** should be disabled for this instance, then it allows that traffic can traverse through it
    - It should reside behind a security group
    - Nat instance must be behind a public security group
    - There should be route to outside (0.0.0.0/0) by using this instance
    - It is not high available but can be done using autoscaling groups and scripts to automate failover
  + **NAT gateway**
    - It should be in the public subnet that has internet access
    - No security group is required
    - No need to patch
    - Need to update route tables manually
* **NAT vs Bastion:** NAT is used to give access to EC2 instances in private subnets (source NAT), Bastion is used to securely connect to EC2 instance in private subnets, SSH or RDP (destination NAT, jump boxes called in AUS)

## Network Access List vs Security Groups

|  |  |
| --- | --- |
| **Security Group** | **Network ACL** |
| Operate at instance level (first layer of defence) | Operate at subnet level (second layer of defence) |
| Allow rules are supported | Both allow/deny rules are accepted |
| Is stateful: return traffic is automatically allowed | Is stateless: return traffic must be explicitly allowed by rules |
| AWS evaluate all rules before allow traffic | AWS process rules in number order |
| Applies to instance | Can be applies to all instance on a subnet |

* VPC comes with a default created ACL that allows all inbound and outbound rules
* Manually created ACL denies all traffic by default
* Each subnet only can be associated to a ACL, if don’t assign it manually, it will be associated to the default ACL
* We can associate an ACL with multiple subnet. When associate an ACL with a subnet, the previous association is removed
* Because ACL is stateless we need to create inbound and outbound rules as well as a rule to allow ephemeral ports 1024-65535
* responses to allowed inbound traffic are subject to the rules for outbound traffic (visa versa)
* ACL is only able to block IP address, security groups are not able to block

## ELB and VPC

* To associate ELB to a VPC, two public subnets is required, cos if one gateway goes down, there is no other gateway to handle traffic

## VPC Flow Log

* It enables to capture IP traffic flow to the vpc
* It needs to create a log group and log stream in Cloudwatch

**Application Services**

## SQS (Simple Queue Service)

* It is a web service that provides message queue system to store messages while waiting for process
* It is the first aws service
* It is both a distributed queue system and a **pulling based** system NOT PUSH based
* It is used for decouple the components of an application that they run independently
* Any component can store messages in a fail-safe queue
* Message size can be up to **256KB**
* Messages can be in the queue for 1 min to 14 days (default is 4 days)
* SQS is like a buffer between producing component and processing component
* It can be configured with auto scaling group that if messages >10 we run another processor instance
* Two types of Queue:
  + **Standard queue(default):** 
    - It guarantee that a message is delivered at least once.
    - Sometimes because of highly-distributed system may more than one copy of data be delivered
    - It tries to deliver messages in order as they are sent but there is no guarantee
  + **Fifo queue:** 
    - It 100% guarantee that data are in FIFO, and then they will processing only once.
    - The order of data are based on entered order
    - Duplicate data are not allowed
* **Visibility Time Out:** after a processor pick a message, that message will be invisible (not deleted) and after visibility timeout will be deleted from the queue. If job is not processed within that time, the message will become visible again for another reader. This could result two time processing.
* Visibility time out maximum is 12 hours
* SQL long polling is a way to retrieve messages from an empty queues.

## SWF (Simple Workflow Service)

* It is a web service for managing tasks across distributed components
* Range of use case, media processing, web application back-ends, business process workflows, analytic pipelines
* Tasks in SWF can be performed by codes, web services, human action and scripts
* SQS has a retention period of 14 days, SWF has up to 1 year waiting time
* If in the exam, they asked a task may take over 15 days, SWF is the answer
* SQS is a message-oriented API, SWF presents a task-oriented API
* SWF ensure a task is assigned only once and no duplicate, in SQS we need to handle duplicate messages
* SWF keeps track of task, in SQL we have to implement our own tracking app
* **SWF Actors:**
  + **Workflow Starters:**
    - An application that starts a workflow. E-commerce website or mobile app
  + **Deciders:**
    - Control the flow of tasks in a workflow execution. If a task gets finished, decider decides what to do next
  + **Activity Workers:**
    - Carry out the activity tasks

## SNS (Simple Notification Service)

* It allows to send notifications from the cloud to subscribers or apps
* It can sends push notifications to Apple, Google, Android devices and …
* It is able to send SMS, send message to SQS, trigger Lambda functions.
* It allows to group multiple recipients using a topic. A topic is “access point” for receiving messages
* Flexible message delivery: one topic can support deliveries to multiple endpoints and protocols like IOS, Android, SMS
* SNS messages are stored redundantly across multiple AZ
* Instantaneous push-based delivery (no polling)
* **SNS vs SQS:**
  + Both are messaging service in aws
  + SNS -> push
  + SQS -> polls (pulls)

## Elastic Transcoder

* It is a media transcoder in the cloud
* It converts media files that get played on different devices like smartphones, tablets, …
* Pay base on the minutes and resolution
* *No need to guess about which settings work best on particular devices*

## API Gateway

* It is a manageable service to work with AWS services using a “front-door”
* API Caching is a mechanism to cache endpoint’s response. With caching, the number of calls will be reduced and it improve the latency. The cached data will be stored for a period of TTL in seconds.
* It is a low cost & efficient service. By connecting it to Cloudwatch we can log all requests
* API gateways is low cost and scales automatically
* We can throttle API gateways to prevent attacks
* **CORS (Cross-Origin Resource Sharing)**
  + If we are using JavaScript/Ajax calls in multiple domains, need to enable CORS
  + CORS is a resource sharing mechanism that allow restricted resources (e.g. fonts) to be requested from another domain
  + Error “ Origin policy cannot be read at the remote resource?” -> need to enable CORS

**Kinesis**

**Streaming Data:** The data that is generated continuously by many data sources with small sizes

* Game data that is sent back from user’s experiences
* Social network data
* Geospatial data (Uber)
* IoT sensor data

**Kinesis:**

* Is an aws platform to send streaming data to.
* *Is a fully-managed service for real-time massive data processing.*
* *Can configure many data producers to continuously put data*
* *It can analyze streaming data, and it provides tools to build apps on these data*
* *Is used to consume big data*
* *To process large amount of data in the cloud we have two services:*
  + *Redshift for business intelligence*
  + *Elastic Map Reduce for Big Data Processing*
* *If in the exam, large amounts of streaming data from social media, news, feeds, logs, etc in the cloud -> Kinesis*
* **Kinesis Core Services:**
  + **Kinesis Streams:** 
    - it stores data generated by data producers like mobile apps.
    - By default it keeps data for 24h, but can be 7 days retention.
    - It got many shards
    - Data are stored in Shards, and then these data will be sent to EC2 instances to process or use them
    - Producers -> Shard -> Consumers -> store
  + **Kinesis Firehose:**
    - It does NOT have Shards and its limitation
    - It is fully automated and no need to be worried about data consumers
    - Data can be directly piped to lambda for processing and the output can directly stored in S3/RedShift/Elasticsearch
    - There is not retention period, because as soon as data get into Firehose, it gets processed and won’t be kept in there
  + **Kinesis Analytics**
    - Allows to use SQL queries on the data and the result can be stored in storage or DB

**Overview of AWS white paper**

## Overview of Cloud Computing

* Is on-demand delivery of IT resources via a pay-as-you-go pricing
* It provides simple way of accessing to servers, storage, database and …
* Providers maintain network hardwares as connections
* **6 advantages of cloud:**
  + Trade capital expense for variable expense
  + Benefit from massive economies of scale
  + Stop guessing about capacity
  + Increase speed and agility
  + Stop spending money running and maintaining data centers
  + Go global in minutes
* In each region, there can be 1 or more data centers that are called AZ
* **Security:**
  + Electronic surveillance and multi factor access control systems
  + Staffed 24 x 7 security guards
  + Access is authorized on a “least privilege basis”

## Overview of Security Processes

* AWS offers a shared security model: AWS is responsible for underlying infrastructure security only, user is responsible for anything put on the IAAS
* AWS is responsible for security of the cloud while user should be responsible for security in the cloud
* IAAS such as EC2, VPC, S3
* Amazon is also responsible for its managed services like Dynamodb, RDS, Redshift, Elastic MapReduce, WorkSpaces
* **Storage Decommissioning:** When a storage device has reached end of life, AWS has a technique to prevent customer’s data to be exposed.
* **Network Security:** 
  + Transmission Protection: Secure Socket Layer (SSL) and IPsec/VPN in VPC
  + Amazon Corporate Segregation (Segmentation): AWS production network is separated from Amazon Corporate network (amazon.com)
* **AWS Network Monitoring & Protection:**
  + DDOS protection
  + Man in the middle attack (MITM)
  + IP Spoofing
    - It is a host based firewall infra that does not permit an instance to send traffic with a source IP or MAC to other than its own
    - It leads to prevent ARP poisoning and MITM attack
    - Unauthorized port scans by EC2 customers are a violation of use policy.
    - For conducting vulnerability test/scan we must request a permission from AWS in advance
  + Port Scanning
  + It stops Packet Sniffing by other tenants
* **AWS Credentials:**
  + Password: AWS root account or IAM user account
  + MFA: Multi-factor Authentication
  + Access Keys
  + Key Pairs
  + X.509 certificate: it ensures that the CDN link we sent out to a friend is secured
* **AWS Trusted Advisor:**
  + It inspect the whole AWS environment and gives recommendation for save money, improve system performance or security gaps.
  + It provides alerts on most common security misconfigurations, open ports, allowing S3 public access, not turning on AWS CloudTrail for auditing logs
* **Instance Isolation:**
  + Different instance running on the same physical machine are isolated from each other using **Xen** hypervisor NOT KVM
  + There is RAM isolation as well, memory allocation to guests is scrubbed (set to zero by hypervisor)
  + EC2 customers have to access to raw disk devices, only virtualized disks
  + Each terminated virtual disk will resets automatically to avoid leaking data
* AWS has no right to control and have access to EC2 instances
* In AWS firewall, all ingress traffics are blocked by default while all egress are opened
* **ELB** supports SSL Termination and allow identify the origination of IP address of a client
* **Direct Connect:** can connect to AWS using a cross-connect. It also supports 802.1q VLAN traffics to make network separation between public and private environments

## Risk & Compliance

* AWS has developed a strategic business plan for risk identification and controls to mitigate and manage risks.
* AWS re-evaluates the strategic business plan at least biannually (6 months)
* Several industry-specific standards:
  + HIPPA
  + Cloud Security Alliance (CSA)
  + Motion Picture Association of America (MPAA)

## Architecturing for the Cloud

* **Business benefits of cloud:**
  + Zero upfront infrastructure investment
  + Just-in-time Infrastructure (detect overloaded)
  + More efficient resource utilization
  + Usage-based costing (no long term contract)
  + Reduced time to market, CloudFormation (startups fan)
  + Automation - “Scriptable infrastructure”
  + Auto-scaling, Proactive Scaling
  + More Efficient Development lifecycle
  + Improve testability, Disaster recovery
  + “Overflow” the traffic to the cloud, combination of on-premises and cloud infra
* **Design for failure**
  + Be a pessimist (-) when designing architecture, assume things will fail.
  + Always design, implement and deploy for automated recovery from failure
  + Assume hardware will fail, outage happens
  + Think about recovery strategies
* **Decoupling the components**
  + The way building components that have no tight dependency on each other. (like SQS)
  + E.g. if one component is stopped other parts continue to work as not failure is happening.
  + It is good to design components that work asynchronously (black box) to cover essence of one part.
* **Implement Elasticity**
  + Elasticity in the cloud can be implemented in three ways:
    - Proactive Cyclic Scaling: periodic saling that occurs at fixed interval (daily, monthly)
    - Proactive Event-based Scaling: Scaling when we expect a big demand for traffic due to a scheduled business event (new product launch, campaigns)
    - Auto-scaling based on demand: by using monitoring service for auto scale up or down

**Additional Exam Tips**

## OpsWorks

* Orchestration service that use Chef
* Chef consists of recipes to maintain a consistent state
* Look for the term “Chef” or “recipes” or “cook books” and think OpsWorks

## Consolidated Billing (Organization account)

* What is AWS Organizations?
  + Is an account management service to enable consolidate multiple AWS accounts into one organization (mostly used for billing) to manage centrally. It is available in two feature sets:
    - Consolidated billing
    - All features
* It is a paying account
* Currently a limit of 20 linked account
* Advantages:
  + One bill per aws account
  + Very easy to track charges and allocate costs
  + Volume pricing discount
  + Unused reserved instances for EC2 are applied across the group
  + Billing alerts per each account or overal
  + CloudTrail is on per account or per region bass but can consolidate logs to a single s3 bucket in the paying account

## Cross Account Access

* Makes it easier to work productively within a multi-account (or multi-role) AWS environment
* One user and switch to other accounts without having their identities

## Resource Groups and Tagging

* **Tags:**
  + Key value pairs attached to aws resources
  + Metadata (data about data)
  + Tags can be inherited, e.g. autoscaling, CloudFormation, Elastic Beanstalk can create other resources
  + We can use tag editor to update tags for all resource in one page
* **Resource Groups**
  + Make it easy to group resources using assigned tags
  + We can group resources with tags
  + Resource groups contain info like: Region, Name, Health Checks
  + Specific information:
    - For EC2 - public & private IP Addresses
    - For ELB - port configurations
    - For RDS - database engine

## Direct Connect

* It doesn’t involve the internet!
* It is a dedicated communication line between on-premises assets (datacenter) and AWS
* It establishes a private connectivity
* It reduces network costs, increase bandwidth throughput and provide a more consistent network instead of internet-based connections.
* It is available in 10/1Gb and sub 1Gb from AWS direct connect partners
* It uses VLAN trunking 802.1Q
* **Benefits:**
  + Reduce costs when using large volume of traffic
  + Increase reliability
  + Increase bandwidth
* **Direct connect vs VPN**
  + VPN can be configured quickly in a minute. Good if immediately is needed
  + VPN needs low to modest bandwidth for connection
  + Direct connect no need internet. It uses dedicated, private connection between intranet and AWS
  + Direct connect provides private Rack/Cage at direct connect facility
  + Direct connect takes up to 3-4 months, vpn in a minute

## Active Directory Integration

* It is possible to authenticated for aws console using Active directory; it is done through SAML authentication (camel:d)
* It should be alway authenticated first using active directory and the you’ll be assigned the temporary security credential

## Workspaces

* It is a VDI
* Is a cloud-based desktop providing compute resources, storage and softwares
* Users can connect to it using most of the devices (PC, Mac, Android)
* It requires a free amazon workspace client application and credential or integrating using Active directory.
* It doesn’t need to have AWS account for each user
* Windows 7 experience provided by wi 2008
* Users can personalize the workspace and even install softwares because b default they will be given administrator account
* Workspaces are persistent
* All data on D:\ is backed up every 12 hours

Finishhhhhhhhhhhhhhhhhhhhhhhhhhhhhh

## Good to read

<https://acloud.guru/forums/aws-certified-solutions-architect-associate/recent?p=1>

<https://medium.com/@franktran/how-to-succeed-in-aws-certified-solutions-architect-associate-exam-8a30344347f>

<https://acloud.guru/forums/exam-strategies/discussion/-KoLb4ymtyzQnv880l5J/94%25_pass_in_8_days_--_my_tips>