**Cloud Computing**

1. **On – Demand Self Service** – It can provision capabilities as needed without requiring human interaction
2. **Broad Network Access** – Capabilities are available over the network and accessed through standard mechanisms.
3. **Resource Pooling** – There is a sense of location independence, no control or knowledge over the exact location of the resources. Resources are pooled to serve multiple consumers using a multi-tenant model.
4. **Rapid Elasticity** – Capabilities can be elastically provisioned and released to scale rapidly outward and inward with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited.
5. **Measured Service** – Resource usage can be monitored, controlled, reported and billed.

**Public vs Private vs Multi vs Hybrid Cloud**

Public Cloud – AWS, Azure, Google

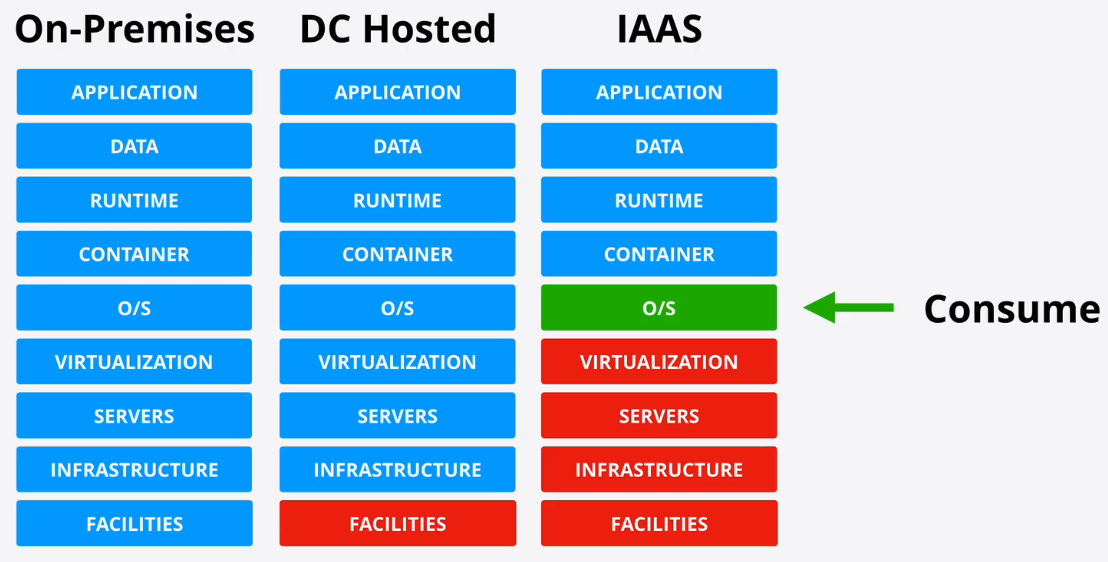
Private Cloud – AWS Outposts, Azure Stack, Anthos

Multi Cloud

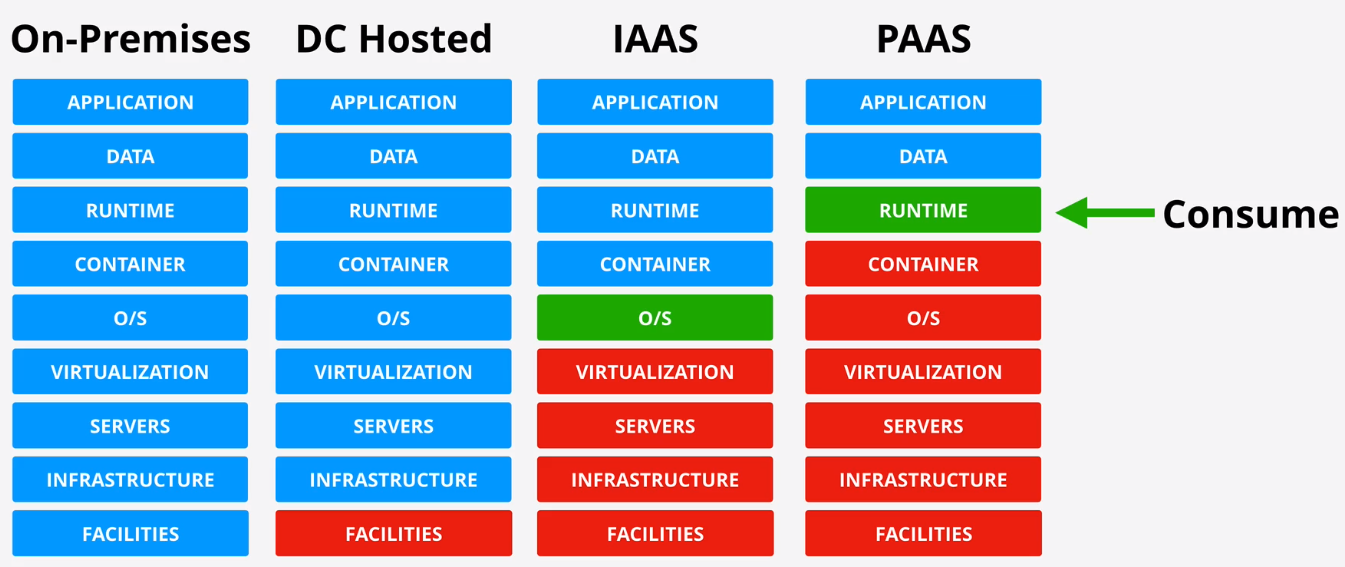
Hybrid Cloud

**Cloud Service Model**

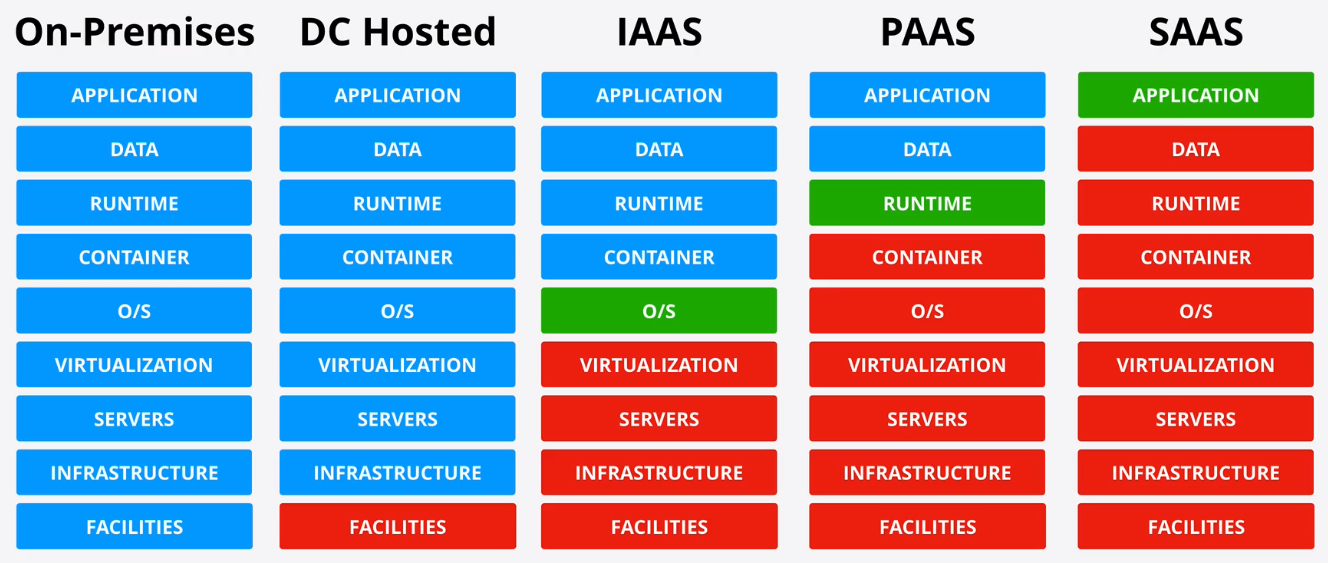
**Infrastructure As A Service (IaaS)**



**Platform As A Service (PaaS)**



**Software As A Service (SaaS)**



Role Switching

Set AWS configure for A4L-MASTER

aws configure --profile A4L-MASTER

Test the configuration

aws s3 ls --profile A4L-MASTER

Set the following in the aws config for Role Switching

[profile A4L-PROD]

role\_arn = arn:aws:iam::226694703277:role/OrganizationAccountAccessRole

source\_profile = A4L-MASTER

[profile A4L-DEV]

role\_arn = arn:aws:iam::103835601551:role/OrganizationAccountAccessRole

source\_profile = A4L-MASTER

Test the Role Switching

aws s3 ls --profile A4L-PROD

aws s3 ls --profile A4L-DEV

**Security Token Service (STS)**

- Generates temproary credentials (sts:AssumeRole)

- If an Identity assume any IAM role then it internally uses the STS by assuming the role

- Role Switch also uses the STS by assuming the role

- It expire and don't belong to the identity

- Limited access

- Used to access AWS resources

- Requested by an Identity (AWS such as IAM role or External such as Facebook, Google etc)

- If an identity isn't allowed to assume a role based on Trust Policy then AssumeRole fails

- STS generates temproary credentials which can access AWS resources until expiration. They

authorize access based on the permission policy.

- Another sts:AssumeRole is required when the credentials expire

Revoking IAM Role Temproary Security Credentials

- Temproary credentials can't be cancelled or manually expired

- When Credentials are leak and attacker can use the temproary credentials to access AWS resources

- We can't change the Trust Policy because that is only used during assuming the role.

- Changing the persmission would impact all the users.

- In order to solve this we can revoke all the existing session. It updates the permission policy

with a AWSRevokeOlderSessions inline DENY for any session older than now.

# ON EC2

curl http://169.254.169.254/latest/meta-data/iam/security-credentials/

get the role name

curl http://169.254.169.254/latest/meta-data/iam/security-credentials/REPLACE\_ME

SET AWS\_ACCESS\_KEY\_ID=AKID

SET AWS\_SECRET\_ACCESS\_KEY=SAK

SET AWS\_SESSION\_TOKEN=TOKEN

aws s3 ls

aws ec2 describe-instances --region us-east-1

**Identity and Access Management**

* Root user of the account has all the permission.
* IAM let’s you create three different type of identity objects
  + **User** – Identities which represent humans or applications that need access to your account
  + **Group** – Collection of related users e.g. development team, finance or HR
  + **Role** – Can be used by AWS Services, or for granting external access to your account
* **Policy** – It is a set of rules that ALLOW or DENY to AWS Services. Policies can be attached to User, Group or Role.
* IAM has three basic responsibilities
  + **Manage Identities** – An Identity Provider (IDP)
  + **Authenticate**
  + **Authorize**
* No Cost service
* Global Service / Global resilience
* Allow or Deny its identities on its AWS account
* No direct control on external accounts or users
* Identity Federation and MFA

**Elastic Compute Cloud (EC2)**

* IAAS – Provides Virtual Machines => Instances, EC2 instances are virtual machines
* EC2 instances run on EC2 Hosts
* Shared Hosts or Dedicated Hosts
* AZ Resilient, Hosts = 1 AZ – AZ fails, Host fails, Instance fails
* Private Service by default – Uses VPC networking
* Running -> Stopped -> Terminated
* Why EC2
  + Traditional OS + Application Compute
  + Long Running Compute
  + Server style applications
  + Either burst or steady state load
  + Monolithic application stacks
  + Migrated application workloads or Disaster Recovery

**EC2 Instance Types**

**EC2 Categories**

* + **General Purpose** – Default – Diverse Workloads, equal resource ratio
  + **Compute Optimized** – Media Processing, HPC, Scientific Modelling, Gaming, Machine Learning
  + **Memory Optimized** – Processing large in-memory datasets, some database workloads
  + **Accelerated Computing** – Hardware GPU, Field Programmable Gate Arrays (FPGA)
  + **Storage Optimized** – Sequential and Random IO – scale-out transactional databases, data warehousing, Elasticsearch, Analytics workloads

<https://aws.amazon.com/ec2/instance-types/>

<https://ec2instances.info/>

**Storage**

* **Direct (local) attached Storage** – Storage on the EC2 Host
* **Network attached Storage** – Volumes delivered over the network (EBS)
* **Ephemeral Storage** - Temporary Storage
* **Persistent Storage** – Permanent Storage – lives on past the lifetime of the instance

**Block Storage** – Volume presented to OS as a collection of blocks…no structure provided. **Mountable. Bootable**

**File Storage** – Presented as a file share…hast structure. **Mountable. Not Bootable**

**Object Storage** – Collection of objects, flat. **Not Mountable. Not Bootable**.

**Elastic Block Store (EBS)**

* Allocate block storage (Volumes) to instances
* Volume = 1 AZ, but HA/Resilient in that AZ
* Different physical storage types available (SSD/HDD)
* Varying levels of performance (IOPS, Throughput)
* Billed as GB/m (some have IOPS component)
* EBS MAX 80k IOPS (Instance), 64k (Vol) (io1)
* MAX 2375 MB/s (Instance), 1000 MiB/s (Vol) (io1)

**Volume Types**

* **General Purpose SSD (gp2)**
  + Default, most workloads. System Boot Volumes. Development and test environments.
  + Default, Balance of price and performance for normal workloads
  + 3 IOPS/GB (Min 100, Max 16000)
  + 1 GB to 16 TB
* **Provisioned IOPS SSD (io1)**
  + DB Workloads eg: MongoDB, Cassandra, MS SQL, Oracle
  + High performance SSD, low latency, high throughput workloads
  + 50:1 IOPS to GB vs 3:1 gp2
  + 4 GB to 16 TB
* **Throughput Optimized HDD** **(st1)**
  + Great Throughput Great Value
  + 500GB – 16TB
  + No Boot Volume
  + Streaming data into the Harddisk, whenever you need storage
  + Frequent access throughput intensive workloads such as Dataware houses, bigdata, log processing
* **Cold HDD (sc1)**
  + Great Throughput Great Value
  + 500GB – 16TB
  + No Boot Volume

**Instance Store Volume**

* Block Storage Devices
* Physically connected to one EC2 Host
* Instances on that host can access them
* Highest storage performance in AWS
* Included in instance price
* Can be attached only at launch
* Data Lost on instance move, resize or hardware failure
* Temporary

**EBS Snapshots**

* Snapshots are incremental volume copies to S3
* The first is a full copy of data on the volume
* Future snaps are incremental
* Volumes can be created (restored) from snapshot
* Snapshots can be copied to another AZ and region
* Fast Snapshot Restore (FSR) – Immediate restore
* Up to 50 snaps per region

**EBS Encryption**

* Accounts can be set to encrypt EBS Volume by default – default CMK
* Otherwise choose a CMK to use
* Each Volume uses 1 unique DEK
* Snapshots and future volumes use the same DEK
* Can’t change a volume to NOT be encrypted
* OS isn’t aware of the encryption ..no performance loss

**Network Interface and DNS Architecture**

* Every EC2 instance start with at least one primary ENI (Elastic Network Interface)
* Secondary ENI in different subnet can also be attached to EC2 provided it’s in same AZ
* Secondary ENI can be detached and assigned to another EC2 instance
* Different Security Groups – multiple interfaces
* OS doesn’t see public IPv4
* Network Interface has Mac Address 00:0d:83:b1:c0:8e
* Network Interface has Primary IPv4 Private IP
* Network Interface has 0 or more secondary IPs
* Network Interface has 0 or 1 Public IPv4 address which dynamic. Stop and Start changes the IP
* Public DNS = Private IP in VPC, public IP everywhere else
* Network Interface has 1 elastic IP per private IPv4 address
* Network Interface has 0 or more IPv6 addresses
* Network Interface has Security Groups
* Network Interface has Source/Destination check

**Amazon Machine Images (AMI)**

* AMI’s can be used to launch EC2 instance
* AWS or community provided
* Marketplace (can include commercial software)
* Regional ..unique ID, only works in that region
* Permissions (Public, Your Account, Specific Accounts)
* AMI Baking - You can configure an EC2 with something additional and can create AMI out of it to launch EC2
* An AMI can’t be edited..launch instance, update configuration and make a new AMI
* Can be copied between regions (includes its snapshots)

**EC2 Pricing Models**

* On-Demand Instances
  + Hourly rate
  + Default Pricing Model
  + New or uncertain application requirement
  + Short-term, spiky, or unpredictable workloads
* Spot Instances
  + 90% off vs On-Demand
  + You can specify a maximum price you will pay
  + If spot prices goes above yours – instances terminate
  + Applications that have flexible start and end times
  + Apps which only make sense at low cost
  + Apps which can tolerate failure
* Reserved Instances
  + Up to 75% off vs On-Demand – for a commitment
  + 1-3 years, all upfront, partial or no upfront
  + Reserved in region, or AZ with capacity reservation
  + Scheduled Reservations
  + Known steady state usage
  + Can’t handle disruption and need low cost
* Dedicated Hosts

**Instance Status and Auto Recovery**

**EC2 Instance Metadata**

* EC2 Service provide data to instances
* Accessible inside ALL instances
* <http://169.254.169.254/latest/meta-data>
* Environment
* Networking
* Authentication
* User-Data
* Not Authenticated or Encrypted
* <https://aws.amazon.com/code/ec2-instance-metadata-query-tool/>

**EC2 Placement Group**

Three types of placement group

* **Cluster** – Pack instances close together
* **Spread** – Keep Instances separated
* **Partition** – groups of instances spread apart

**Cluster Placement Groups**

* Can’t span AZs – One AZ Only
* Can span VPC peers – but impacts performance
* Requires a supported instance type
* Use the same type of instance
* Launch at the same time
* 10 Gbps single stream performance
* Use Cases : Performance, fast speeds, low latency

**Spread Placement Groups**

* Provides infrastructure isolation – each instance runs from a different rack
* 7 instances per AZ
* Not supported for Dedicated instances or hosts
* Use case: Small number of critical instances that need to be kept separated from each other

**Partition Placement Groups**

* 7 partitions per AZ
* Instances can be placed in a specific partition
* Or auto placed
* Partition placement groups are not supported for Dedicated hosts
* Great for HDFS, HBase and Cassandra

**EC2 Dedicated Host**

* EC2 hosts dedicated to you
* Specific family e.g. a1, c5, m5
* No instances charges ..you pay for the host
* On-Demand and Reserved Options available
* Host hardware has physical sockets and cores
* AMI limits – RHEL, SUSE Linux and Windows AMIs’ are not supported
* RDS instances are not supported
* Placement Groups are not supported for Dedicated Hosts
* Hosts can be shared with other ORG accounts…RAM

**Load Balancer**

**Application Load Balancer (ALB)**

* ALB is a Layer-7 load balancer – understands HTTP/S
* Scalable and highly available
* Internet facing or internal
* Listens on the outside -> Sends to Target(s) (Groups)
* Hourly rate and LCU Rate (Capacity)
* Cross Zone Load Balancing
* Target => Target Groups which are addressed via rules
* Rules are path based or host based
* Support EC2, ECS, EKS, Lambda, HTTPS, HTTP/2 and Websockets
* ALB can use SNI for multiple SSL certs – host-based rules
* Recommended vs CLB (Legacy)

**Network Load Balancer (NLB)**

* NLB’s are **Layer-4** so it **only understands TCP and UDP**
* **Can’t understand HTTP/S** but are faster 100ms vs 400ms for ALBs
* Rapid Scaling – millions of requests per second
* 1 interface w/ static IP per AZ, can use Elastic IPs (**whitelisting**)
* Can do **SSL Pass through**
* Can load balance non-HTTP/s applications – doesn’t care about anything above TCP and UDP

**SSL Offload**

**Bridging** –

* Default mode. Listener is configured for HTTPS. Connection is terminated on the ELB and needs a certificate for the domain name
* ELB initiates a new SSL connection to backend instance.
* Instances need SSL certificates and the compute required for cryptographic operation

**Passthrough** –

* Listener is configured for TCP. No encryption or decryption happens on NLB. Connection is passed to the backend instance.
* Each instance needs to have the appropriate SSL certificate installed.
* With this architecture there is no certificate exposure to AWS.. all self-managed and secured

**Offload –**

* Listener is configured for HTTPS. Connections are terminated and then backend connection use HTTP.
* ELB to instance connections use HTTP – no certificate or cryptographic requirements.

**Connection Stickiness**

* With no Stickiness connections are distributed across all in-service backend instances. Unless application handles user state this could cause user logoffs shopping cart losses.
* Stickiness generates a cookie which locks the device to a single backend instance for a duration. Unless a server failure or cookie expired.

**Launch Configuration and Launch Templates**

* Allow you to define the configuration of an EC2 instance in advance
* AMI, Instance Type, Storage and Key-pair
* Networking and Security Groups
* User data and IAM role
* Both are NOT editable – defined once. LT has revisions
* LT provide newer feature - including T2/T3 unlimited, Placement Groups, Capacity Reservations, Elastic Graphics

**Auto Scaling Groups**

* **Automatic Scaling** and **Self-Healing** for EC2
* Uses Launch Templates or Configurations
* Has a **Minimum, Desired and Maximum** Size (e.g. 1:2:4)
* Provision or Terminate Instances to keep at the Desired level (between Min/Max)
* Autoscaling Groups are free, only the resources created are billed
* Use **cool downs** to avoid rapid scaling
* Use with ALB’s for elasticity – abstraction
* ASG defines WHEN and WHERE, LT defines WHAT
* Scaling Policies automate based on metrics
  + Manual Scaling
  + Scheduled Scaling
  + Dynamic Scaling
    - Simple – “CPU above 50% +1”, “CPU Below 50 -1”
    - Stepped Scaling – Bigger +/- based on difference
    - Target Tracking – Desired Aggregate CPU = 40% …ASG handle it
* Cooldown Periods

**Simple Storage Service (S3)**

* Global Storage Platform – **regional based/resilient**
* Public service, unlimited data and multi-user
* Movies, Audio, Photos, Test, Large Data sets
* Economical and accessed via UI/CLI/API/HTTP
* Objects and Buckets
* S3 is **private by default**
* S3 can store up to **5TB**
* Bucket names are **globally unique**
* S3 stores all the objects at **root label as flat structure**. There is no file structure.
* Buckets – **100 soft limit per account**, 1000 hard limit per account
* Key = Name and value = data
* S3 is an **object store** – not file or block
* You **can’t mount** an S3 bucket
* Great for large scale data storage, distribution or upload
* Great for offload
* Input and/or Output to many AWS products

**S3 Bucket Policies**

* A form of **resource policy**
* Like identity policies, but attached to a bucket
* Resource perspective permissions
* **ALLOW/DENY same or different AWS accounts**
* **ALLOW/DENY anonymous principals**

**Access Control List (ACLs)**

* **ACLSs on objects and bucket**
* A Sub resource
* **Legacy**
* **Inflexible and Simple** permissions

**Block Public Access -** Blocks the anonymous principal accessing the bucket

**S3 Static Website Hosting**

* Normal access is via AWS APIs
* This feature allows access via HTTP – e.g. Blogs
* Index and Error documents are set
* Website Endpoint is created
* Custom Domain via R53 – BUCKETNAME MATTERS

**Object Versioning**

* By default, S3 Object versioning is disabled, **once enabled you can’t disable it again**.
* It **can be suspended and enabled again**.
* Versioning lets you store multiple versions of objects within a bucket.
* Operations which would modify objects generate a new version.
* **Space is consumed by ALL versions** and you are billed for ALL version
* **Only way to 0 costs is to delete the bucket**
* When you delete the object, it doesn’t get delete in real instead **Delete Marker** is created.
* When you **delete the Delete Marker**, you can **retrieve the original objects**.
* In order to delete object, you have to **select the actual version** of the object and **then delete it**.

**MFA Delete**

* Enabled in versioning configuration
* MFA is required to change bucket versioning state
* MFA is required to delete versions
* Serial number (MFA) + codes passed with API calls

**Single Put Upload**

* Single data stream to S3
* **Stream fails – upload fails**
* **Requires full restart**
* Speed and Reliability = limit of 1 Stream
* **Any upload up to 5GB**

**Multipart Upload**

* Data is broken up
* **Min data size 100MB** for multipart
* **10000 max parts, 5MB -> 5MB**
* Last part can be smaller than 5MB
* Parts can fail, and be restarted
* Transfer rate = speed of all parts

**S3 Transfer Acceleration** - Data is transferred to the **Edge Locations** using AWS network.

**Encryption**

Concepts

* Plain Text
* Algorithm
* Key
* Cipher Text

**Encryption At Rest**

**Encryption in Transist**

**Symmetric Encryption**

**Asymmetric Encryption**

**Signing**

**Steganography –** It allows you to embed the data in another data such as image.

**Key Management Service (KMS)**

* Regional and Public Service
* Create, Store and Manage Keys
* **Symmetric and Asymmetric Keys**
* **Cryptographic operations** (encrypt, decrypt)
* **Key never leaves KMS** – Provides FIPS 140-2 (L2)

**Customer Master Key**

* **CMK is logical** – **ID, date, policy, desc & state**
* Backed by physical key material
* Generate or Imported
* CMKs can be used for upto **4KB** of data
* CMKs are **isolated to a region and never leave**
* There are 2 types of CMKs – **AWS Managed or Customer Managed CMKs**
* **Customer managed keys are more configurable**
* **CMKs support rotation**
* You can **also create aliases**
* Every CMK **has one Key Policies** (Resource)

**Data Encryption Keys (DEK)**

* GenerateDataKey works on > 4KB
* **Data Encryption Key is created using the Customer Master Key**
* You get the plain text version key
* And get the Cipher text version
* Encrypt data using plain text key
* Then **Discard the key**
* Store encrypted key with data

**S3 Encryption**

* Buckets are not encrypted, objects are
* There are 2 types of encryption Client Side and Server side encryption
* **Client-side encryption** happens at client side before it even leave for transit to S3
* **Server-side encryption** 
  + Server-Side Encryption with Customer-Provided Keys (**SSE-C**)
  + Server-Side Encryption with Amazon S3-Managed Keys (**SSE-S3**)
  + Server-Side Encryption with Customer Master Keys (CMKs) stored in AWS Key Management Service (**SSE-KMS**)

Bucket Default Encryption – Use header x-amz-server-side-encryption (AES256 or aws:kms)

**S3 Object Storage Class**

**S3 Standard**

* It is a **default** storage class for S3.
* Use S3 Standard for **frequently accessed data** which is **important and non-replaceable**
* When objects are stored a **HTTP 200 OK** response is provided by S3 endpoint
* S3 standard has a **millisecond first byte latency** and objects can be made publicly available
* Objects are **replicated across at least 3 AZs** in the AWS region
* 99.9999999% durability for 10000000 objects, 1 object loss per 10000 years
* Replication over 3 AZ’s and Content MD5 Checksums and cyclic redundancy checks are used to detect and fix any data corruption

**S3 Standard IA (Infrequent access)**

* Should be used for **long lived data which is important** but where **access is infrequent**
* S3 standard **IA has a per GB data retrieval fee**, overall cost increases with frequent data access
* S3 standard IA has a **minimum duration charge of 30 days** – objects can be stored for less but the minimum billing always applies
* It has a minimum capacity charge of **128KB** per object
* Objects are replicated across at least 3 AZs in the AWS region
* 99.9999999% durability for 10000000 objects, 1 object loss per 10000 years
* Replication over 3 AZ’s and Content MD5 Checksums and cyclic redundancy checks are used to detect and fix any data corruption

**S3 One Zone IA**

* Should be used for **long lived data which is not critical** and **replaceable and infrequently accessed**
* Does not provide the multi-AZ resilience model of Standard or Standard IA, instead **only one AZ is used in the zone**.
* One Zone IA has a per GB data retrieval fee, overall cost increases with frequent data access
* One Zone IA has a **minimum duration charge of 30 days** – objects can be stored for less but the minimum billing always applies
* It has a minimum capacity charge of 128KB per object

**S3 Glacier**

* **Archival data where** **frequent or real time access isn’t needed**. **Minutes-hours retrieval**
* Objects access requires a retrieval process
* Data in Glacier Is retrieved to S3 Standard IA temporarily
  + Expedited (1-5 minutes)
  + Standard (3-5 hours)
  + Bulk (5-12 hours)
  + Faster = More Expensive
* First byte latency = **minutes or hours**
* 99.9999999% durability for 10000000 objects, 1 object loss per 10000 years
* Objects are replicated across at least 3 AZs in the AWS region
* Replication over 3 AZ’s and Content MD5 Checksums and cyclic redundancy checks are used to detect and fix any data corruption

**S3 Glacier Deep Archive**

* **Archival data that rarely if ever needs to be accessed** – **hours or days retrieval**. E.g Legal or Regulation data storage
* Objects access requires a retrieval process
* Data in Glacier Deep Archive Is retrieved to S3 Standard IA temporarily
  + Standard (12 hours)
  + Bulk (48 hours)
* First byte latency = **hours or days**
* 99.9999999% durability for 10000000 objects, 1 object loss per 10000 years
* Objects are replicated across at least 3 AZs in the AWS region
* Replication over 3 AZ’s and Content MD5 Checksums and cyclic redundancy checks are used to detect and fix any data corruption

**S3 Intelligent -Tiering**

* It is **automatically managed between Frequent Access and Infrequent Access**
* If you don’t know how your access patterns are or constantly changing

**S3 Lifecycle Configuration**

* A lifecycle configuration is a set of rules
* Rules consist of actions
* On a bucket or groups of object
* Transition Actions
* Expiration Actions

**S3 Replication**

* **Cross-Region Replication (CRR)**
* **Same-Region Replication (SRR)**
* Buckets can be in same or different aws account
* **Replication configuration is applied to source bucket**
* All objects or a subset
* Storage Class – default is to maintain
* Ownership – default is the source account
* **Replication Time Control (RTC)**
* Not retroactive & **Versioning needs to be ON**
* **One-way replication** Source to Destination
* Can replicate Unencrypted, SSE-S3 and SSE-KMS with extra config
* Source bucket owner needs permissions to objects
* **NO system events, Glacier or Glacier Deep Archive**
* **NO Deletes**
* Why Replication
  + SRR – Log Aggregation
  + SRR – PROD and Test Sync
  + SRR – Resilience with strict sovereignty
  + CRR- Global Resilience Improvements
  + CRR – Latency Reduction

**S3 Presigned URLs**

* You **can create a URL for an object you have no access too**
* When using the URL, the permissions match the identity which generated it
* **Access denied could mean the generating ID never had access** or doesn’t know
* **Don’t generate with a role**…URL stops working when temporary credentials expire. **Always generate the presigned url using identity.**

**S3 Select and Glacier Select**

* S3 can store objects upto 5TB
* You often want to retrieve the entire object
* Retrieving a 5TB object..takes time, uses 5 TB
* Filtering at the client side doesn’t reduce this
* **S3/Glacier select let you use SQL-Like statements**
* **To select part of the object, pre-filtered by S3**
* CSV, JSON, Parquet, BZIP2 compression for CSV and JSON

**Virtual Private Cloud (VPC)**

* VPC is a virtual network inside AWS
* **A VPC is within 1 account and 1 region**
* Private and isolated unless you decide otherwise
* Two types – Default VPC and Custom VPC
* **One per region** – can be removed and recreated
* Default VPC CIDR is always 172.31.0.0/16
* /20 subnet in each AZ in the region
* Internet Gateway, Security Group and NACL
* Subnets assign public IPv4 address

**Classless Inter-Domain Routing (CIDR)**

**0.0.0.0/0** means **ALL IP addresses**

**10.0.0.0/8** means **10.ANYTHING** – **Class A – 16 million IPs**

**10.0.0.0/16** means **10.0.ANYTHING** **– Class B – 65536 IPs**

**10.0.0.0/24** means **10.0.0.0-255 – Class C – 256 IPs**

You can divide 10.0.0.0/24 just by increasing the prefix to /25 which results into 10.0.0.0-127 and 10.0.0.127-255

**1.3.3.7/32** the 32 means **1 IP address**

**VPC Sizing and Structure**

**VPC Considerations**

* What size should be the VPC be
* Are there any Networks we can’t use
* What IPs are already being used for other VPC’s, Cloud, On-premises, Partners and Vendors
* Try to predict future
* VPC Structure – How many Tiers and Availability Zones (Resiliency)
* **VPC can have minimum /28 (16 IPs), maximum /16 (65456 IPs)**
* Personal preference for the 10.x.y.z range
* Avoid common ranges – avoid future issues
* To determine IP ranges, Reserve 2+ networks per region being used per account

**VPC Sizing Scenario (**Example Animals4Life**)**

**IP Ranges to Avoid**

192.168.10.0/24 (192.168.10.0 -> 192.168.10.255)

10.0.0.0/16 (AWS) (10.0.0.0 -> 10.0.255.255)

172.31.0.0/16 (Azure) (172.31.0.0 -> 172.31.255.255)

192.168.15.0/24 (London) (192.168.15.0 -> 192.168.15.255)

192.168.20.0/24 (New York) (192.168.20.0 -> 192.168.20.255)

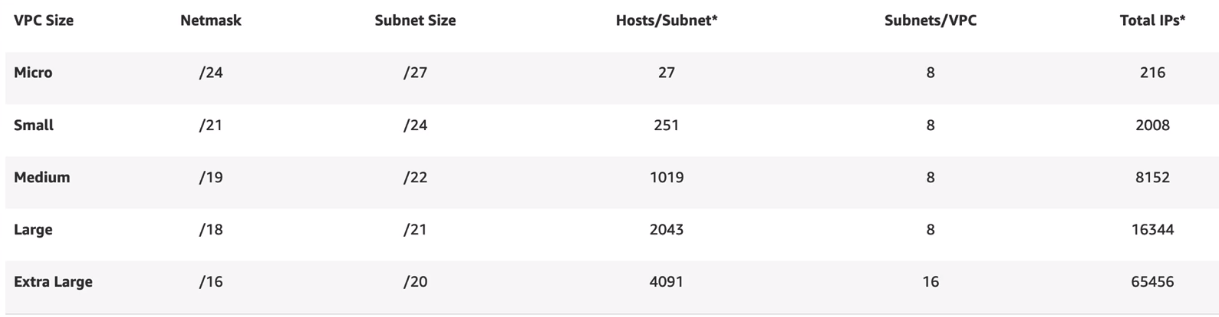
192.168.25.0/24 (London) (192.168.25.0 -> 192.168.25.255)

10.128.0.0/9 (GCP) (10.128.0.0 -> 10.255.255.255)

Regions in our scenario 3 US, Europe, Australia 5 x 2 = 10

Assume 4 accounts = 10 x 4 = 40 IP ranges

**VPC Sizing**



**How many subnets will you need? How many IPs total? How many per subnet?**

How many AZs your VPC would need. It depends on the region also. By default, I start with 3 AZ and 1 Spare so minimum 4. That means at least split the VPC in **4 smaller networks**. So, if we started with /16 now we have four /18

We also have tiers (web, app, db, spare). If you use 1 AZ, then each tier would need its own subnet that mean 4 subnets. Since we are shooting for 4 AZs. So, each tier would have its own subnet in each AZ. So total **16 subnets**. So, if we started with /16 now we have sixteen **/20**.

**VPC Proposal**

* Animals4life could become a huge global entity
* Use the 10.16 -> 10.127 range (avoiding google)
* Start at 10.16 (US1), 10.32 (US2), 10.48 (US3), 10.64 (EU), 10.80 (Australia) – each AWS account has 1/4th
* Eachaccount in each region get four /16 ranges that means 4 VPC per account
* /16 per VPC – 3 AZs (+1), 3 Tiers (+1) – 16 subnets
* /16 split into 16 subnets = /20 per subnet (4091 IPs)



**Custom VPC**

* **VPCs are regionally isolated and regionally resilient** service. VPCs are created in a region and operates from all of the AZs.
* Allows you to create isolated network. Can have multiple VPCs within a region.
* **Nothing IN or OUT without explicit configuration**.
* Default VPC created by AWS using the one subnet per AZ and requiring no config from admin.
* Flexible configuration – Simple or Multi-tier
* Hybrid Networking – Other Cloud and On-premises
* Default or Dedicated Tenancy
* Can use IPv4 Private CIDR Blocks & Public IPs
* **1 primary private IPv4 CIDR block**
* **Min/28 (16 IPs) and Max /16 (65536 IPs)**
* Optional secondary IPv4 Blocks
* **Optional single assigned IPv6 /56 CIDR block**
* Provided by Route53
* VPC Base IP+2 Address
* **enableDnsHostnames** – give instance DNS Names
* **enableDnsSupport** – enables DNS resolution in VPC

**VPC Subnets**

* AZ Resilient
* **A subnetwork of VPC within a particular AZ**
* **1 Subnet in 1 AZ**. It can’t be in more than 1 AZs.
* **But 1 AZ can have multiple subnets**
* **IPv4 CIDR is a subset of the VPC CIDR**
* Can’t overlap with other subnets
* Optional IPv6 CIDR (/64 subset of the /56 VPC – space for 256)
* Subnets can communicate with each other in the VPC
* **5 IPs are reserved within a Subnet**
* Network Address (10.16.16.0)
* Network + 1 (10.16.16.1) – VPC Router
* Network + 2 (10.16.16.2) – Reserved DNS
* Network +3 (10.16.16.3) – Reserved Future Use
* Broadcast Address (10.16.16.255) – Last IP in Subnet

<https://www.site24x7.com/tools/ipv4-subnetcalculator.html>

**VPC Router**

* **Every VPC has a VPC Router** – Highly available
* In every subnet **‘network+1’** address
* **Routes traffic between subnets**.
* **Controlled by “Route Tables”** each subnet has one
* A **VPC has a “Main” route table** – **subnet default**
* If you create route table and associate with subnet, then “Main” route table is disassociated with that subnet
* **A subnet can be associated with one route table at any time, but route table can be associated with many subnets**
* Route table Higher Prefix = More Specific = Higher Priority

**Internet Gateway (IGW)**

* **Region resilient gateway** attached to VPC
* VPC can be attached to **only one Internet Gateway**
* Runs from within the AWS Public Zone
* **Gateways traffic between the** **VPC and the Internet or AWS Public Zone (S3, SQS, SNS etc)**
* Managed – AWS handles performance

**Bastion Host / Jumpbox**

* **An ec2 instance in a public subnet**
* Incoming management connection arrive there
* Then access internal VPC resources
* **Often the only way IN to a VPC**

**Network Access Control List (NACL)**

* **NACL** is like the **firewall around the VPC subnet**
* **All VPCs are created with a default NACL** and **this default NACL is associated with all the subnets within the VPC**
* NACLs are used **when a traffic enters or leaves the subnet**
* NACLS are consist of INBOUND and OUTBOUND rules
* **Highest priority rule is the rule with lowest number**
* NACL can **explicitly ALLOW and DENY traffic**
* NACL default rule with number 100 **by default ALLOW all the traffic**
* **Stateless** – **Initiation and Response seen as different**
* It supports only IPs/Networks, Ports and Protocols – no logical resources
* NACL can’t be assigned to AWS resources but only subnets
* One subnet can be associated with only one NACL at a time

**Security Groups (SG)**

* Security groups are **assigned to the Network Interface**
* **SGs are stateful** – Traffic and Response – same rule
* SGs can filter based on **AWS logical resources**. Resources, other **SGs and even themselves**
* **SGs are by default DENY**
* **It can’t explicitly DENY but can ALLOW**

**SGs vs NACL**

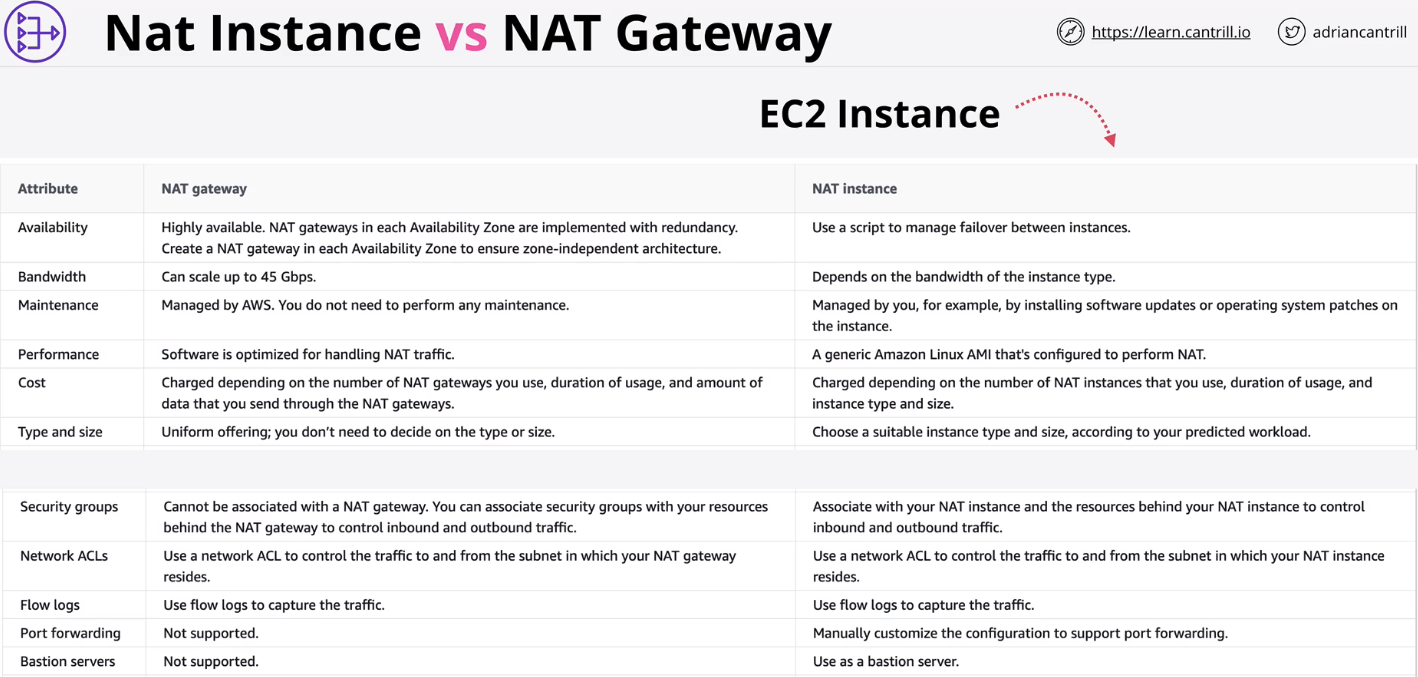
* NACLs on subnet for any products which don’t work with SGs e.g. NAT Gateways
* NACLs when adding explicit DENY (bad IPs, bad actors)
* SGs as the default almost everywhere

**Network Address Translation (NAT)**

* **A process of giving private resource outgoing only access to the internet**.
* **A set of processes – remapping SRC or DST IPs**
* IP masquerading – **hiding CIDR blocks behind one public IP**
* Gives private CIDR range **outgoing internet access**
* **NAT isn’t required for IPv6**
* **All IPv6 addresses in AWS are publicly routable**
* The internet Gateway works with ALL IPv6 IPS directly
* **NAT Gateways don’t work with IPv6**

**NAT Gateway**

* **Runs from a public subnet**
* **Uses Elastic IPs** (Static IPv4 public)
* AZ resilient Service (HA in that AZ)
* For region resilience – **Nat Gateway in each AZ**
* Route table in for each AZ with that NAT Gateway as target
* Managed, scales to 4 Gbps, $ Duration & Data volume



**Domain Name System (DNS)**

* DNS is a discovery service
* Translates machine into human and vice-versa
* [www.amazon.com](http://www.amazon.com) -> 104.98.34.131
* It’s huge and has to be distributed
* DNS resolver sits on the router or internet to redirect to DNS zone
* DNS zone has the NameServer which has DNS record
* By querying the NameServer we can get IP associated with amazon.com
* **DNS Client** => Your laptop, phone, tablet, PC
* **Resolver** => Software on your device, or a server which queries DNS on you behalf
* **Zone** => A part of the DNS database (e.g. amazon.com)
* **Zonefile** => physical database for a zone
* **Nameserver** => where zonefiles are hosted
* **DNS Root & Root Zone (.com, .org, .in)**
* **Root Hints –** config points at the root servers IPs and addresses
* **DNS Root Server**  - There are 13 DNS Root Server
* **Root Zone** – points at TLD authoritative servers
* **gTLD** – generic top level domain (.com .org)
* **ccTLD** – country-code top level domain (.uk .eu etc)

**Route53**

* **Register Domains**
* **Host Zones .. managed nameservers**
* Global Service …single database
* Globally Resilience

**Hosted Zones**

* A R53 Hosted Zone is a **DNS DB for a domain** e.g: animals4life.org
* **Contains** the **Zone Files** on at least **four managed Name Servers**
* **Public** = Hosted on R53 provided public DNS Servers
* Or **private** ..linked to VPC(s)
* Globally resilient (multiple DNS Servers)
* Created with Domain Registration via R53 or can be created separately
* Stores Host DNS Records (e.g A, AAAA, MX, NS, TXT ..)
* Hosted Zones are what the DNS system references – Authoritative for a domain e.g. Animals4life.org

**DNS Record Types**

Nameserver (NS)

A Record – host name to IPv4

AAAA Record – host name to IPv6

CNAME Record – Host to Host. CNAME can’t point to the IP ..it should only point to the name

MX Record

TXT Record – It can be used in proving the ownership

**Route53 Health Checks**

* Health Checks are separate from, but are used by records
* Health checker located globally
* TCP, HTTP/HTTPS
* Endpoint, Cloud Watch Alarm, Checks of Check
* 30 seconds default

**Route53 Routing Polices**

* Simple
* Failover
* Weighted
* Latency Based
* Geolocation
* Multi Value

**Storage**

**Relational Database Service (RDS)**

* Database-as-a-Service
* DatabaseServer-as-a-Service
* **Managed Database** **Instance**
* By default, **it’s in only AZ**
* Multiple engines MySQL, MariaDB, PostgreSQL, Oracle, Microsoft SQL Server
* RDS is accessed using the **Database CNAME**
* One **EBS Storage** is attached to RDS

**RDS Multi AZ**

* No Free-tier – Extra cost for standby replica
* **Standby** **can’t be directly used**
* **60 – 120 seconds** failover
* **Same region only** (other AZs in the VPC)
* **Backups** taken from Standby (removes performance impact)
* AZ Outage, Primary Failure, Manual failover, Instance type change and software patching

**RDS Backups**

* **RTO – Recovery Time Objective**

- Time between the last backup and the incident

- Amount of maximum data loss

- Influences technical solution and cost

- Generally lower values cost more

* **RPO – Recovery Point Objective**

- Time between the DR event and full recovery

- Influenced by process, staff, tech and documentation

- Generally lower values cost more

* **Automatic Backups** and **Manual Snapshots** – both store the back up **in S3 bucket**
* First Manual Snapshot is FULL and then incremental
* Manual Snapshots don’t expire, you have to delete it manually
* Automatic Backups are configured
* Automatic Backups **write transaction logs to S3** in every 5 minutes
* Automatic backups **retention period is 0 to 35 days**

**RDS Restore**

* **Creates a New RDS instance** – new address
* **Snapshots = single point in time**, creation time
* Automated = any 5-minute point in time
* Backup is restored and transaction logs are replayed to bring DB to desired point in time
* **Restores aren’t fast** – Think about RTO

**Read Replica**

* **5x** direct read replicas per DB instance
* Each providing an additional instance of read performance
* Read-Replicas can have read-replicas – but lag starts to be a problem
* Global performance improvements
* Snapshots and Backups improve RPO
* RTO’s are a problem
* **RR’s offer near zero RPO**
* **RR’s can be promoted quickly** – low RTO
* Failure Only – **watch for data corruption**
* Read only – until promoted
* Globally availability improvements …global resilience

**Amazon Aurora Architecture**

* Aurora architecture is very different from RDS, **it uses a “Cluster”**
* A **single primary instance +0** or more replicas
* No local storage – **uses cluster volume**
* Faster provisioning and improved availability and performance
* It **functions across multiple AZs**
* It has **6 replicas** across multiple AZs
* **Storage synchronize across the multiple AZs** to provide HA, this replication happens at storage level
* Aurora **avoids any data loss** by using the data from other storage replica
* All SSD based – high IOPS, low latency
* **Storage is billed on what’s used**
* High water mark – billed for most used
* Storage which is freed up can be re-used
* Replicas can be added or removed without requiring storage provisioning
* Unlike RDS, **Aurora has multiple endpoints** such as **Cluster Endpoint** and **Reader Endpoint**
* Backups in Aurora work in the same way as RDS
* Restores create a new cluster
* Backtrack can be used which allow in-place rewinds to a previous point in time
* Fast clones make a new database MUCH faster than copying all the data – copy on write

**Costs**

No Free-Tier option

Aurora doesn’t support micro instances

Compute – hourly charge, per second, 10 minute minimum

Storage – GB-month Consumed, IO costs per request

100% DB size in backups are included

**Aurora Serverless**

* Scalable – ACU – **Aurora Capacity Units**
* **Aurora Serverless cluster has a MIN & MAX ACU**
* Cluster **adjusts based on load, can go to 0 and be paused**
* Consumption billing per – second basis
* **Same resilience as Aurora (6 copies across AZs)**
* Aurora Serverless Cluster has a shared Proxy Fleet. Any application trying to access cluster will go via Proxy Fleet which as a connection Broker between client application and Aurora Serverless cluster.
* Use Cases
  + Infrequently used applications
  + New applications
  + Variable workloads
  + Unpredictable workloads
  + Development and test databases
  + Multi-tenant applications

**Aurora Global Database**

* **Cross-Region DR and BC**
* **Global Read Scaling** – low latency performance improvements
* **1 second or less replication** between regions, No Impact on DB performance during replication
* **Secondary regions** can have **16 replicas**
* **Can be promoted to R/W**
* Currently **Max 5 secondary region**

**Aurora Multi-Master**

* **Default Aurora** mode is **Single-Master**
* **One R/w** and **0+ Read only replicas**
* Cluster endpoint is used to write, read endpoint is used for load balanced reads
* Failover takes time – replica promoted to R/W
* In **Multi-Master** mode **all instances are R/W**

**Elastic File System (EFS)**

* EFS is an **implementation of NFSv4**
* EFS Filesystems **can be mounted in Linux**
* **Shared between many EC2** instances
* **Private service, via mount targets inside a VPC**
* Can be accessed from on-premises – VPN or DX
* General Purpose and Max I/O performance modes
* General Purpose = default for 99.9 % of uses
* Bursting and Provisioned Throughput Modes
* Standard and Infrequent Access (IA) Classes, lifecycle policies can be used with classes

**DynamoDB**

* **NoSQL** Public Database-as-a-Service (DBaaS) – Key/Value and Document
* **No self-managed servers or infrastructure**
* Manual/Automatic provisioned performance IN/OUT or On-Demand
* **Highly Resilient ...across AZs** and optionally global
* Really fast …single-digit milliseconds (SSD based)
* Backups, point-in-time recovery, encryption at rest
* Event-Driven integration …do things when data changes
* **A table is a grouping of ITEMS with the same PRIMARY KEY**
* Primary Key can be either **Simple (Partition) or Composite (Partition & Sort)** PRIMARY KEY
* Each item MUST has a unique value for PK and SK
* **Can have none, all, mixture or different attributes**. DynamoDB has no rigid attribute schema
* Items can be **Max of 400 KB**
* Capacity (speed) is set on a table
* **Write 1 WCU = 1KB per second**
* **Read 1 RCU = 4KB per second**
* There are two types of Backups
  + **On-Demand Backup** -> Full copy Table is retained until Removed.
  + **Point-in-time-Recovery (PITR)** -> Not Enabled by default. Continuous record of changes allows replay to any point in the window. 35 days recovery window. 1 second granularity -> New Dynamo DB Table

**DynamoDB Operation**

* **On-Demand Capacity** – Unknown, unpredictable, low admin
* On-Demand – price per million R or W units
* **Provisioned Capacity** – **RCU and WCU set on a per table basis**
* **Every operation consumes at least 1 RCU/WCU**
* 1 RCU is 1 x 4KB read operation per second
* 1 WCU 1 x 1KB write operation per second
* Every table has a RCU and WCU burst pool (300 seconds)

**Query**

* **Query accepts a single PK value and optionally a SK** **or range.**
* Capacity consumed is the size of all returned items.
* Further filtering discards data – capacity is still consumed.
* **Can only query on PK or PK and SK**

**Scan**

* It is least efficient Operation but also very flexible
* **Scan moves through a table consuming the capacity of every ITEM**.
* You have complete control on what data is selected, **any attributes can be used,** and any filters applied but scan consumes capacity for every ITEM scanned through.

**Consistency Model**

* **Eventual Consistency**
  + Cheaper but may not return the most accurate data
* **Strong Consistency**
  + Expensive but always return the accurate data

**DynamoDB Stream**

* Time ordered list of ITEM CHANGES in a table
* 24 hour rolling window
* Enabled on a per table basis
* Records INSERTS, UPDATES and DELETES
* Different view types influence what is in the stream
* There are 4 view types
  + **KEYS\_ONLY**
  + **NEW\_IMAGE**
  + **OLD\_IMAGE**
  + **NEW\_AND\_OLD\_IMAGES**

**Trigger Concepts**

* ITEM changes generate an event
* That event contains the data which changed
* An action is taken using that data
* AWS = Streams + Lambda
* Reporting and Analytics
* Aggregation, Messaging or Notifications

**DynamoDB Indexes**

* Query is the most efficient operation in DynamoDB
* Query can only work on 1 PK value at a time
* And Optionally a single, or range of SK values
* **Indexes** are **alternative views** on table data
  + **Local Secondary Indexes**
    - Different SK (LSI)
    - **LSI is an alternative view for a table**
    - **MUST be created with a table**
    - **5 LSI’s per base table**
    - **Alternative SK on the table**
    - Shares the RCU and WCU with the table
    - Attributes – ALL, KEYS\_ONLY & INCLUDE
  + **Global Secondary Indexes**
    - Different PK and SK (GSI)
    - **Can be created at any time**
    - **Default limit of 20 per base table**
    - **Alternative PK and SK**
    - GSI’s have their own RCU and WCU allocation
    - Attributes – ALL, KEYS\_ONLY & INCLUDE
    - **GSI’s are always eventually consistent**, replication between base and GSI is Asynchronous
    - **Use GSI as default**, LSI only when strong consistency is required

**DynamoDB Global Tables**

* Global tables **provide multi-master cross-region replication**
* **Tables are created in multiple regions and added to the same global table** (becoming replica tables)
* **Last writer wins is** used for conflict resolution
* Reads and Writes can occur to any region
* Generally sub-second replication between regions
* Strongly consistent reads ONLY in the same region as writes

**DynamoDB Accelerator (DAX)**

* Application uses the DAX SDK and makes a single call for the data which is returned by DAX
* DAX either returns the data from its cache or retrieves it from the database and then caches it
* Less complexity for the app developer – tighter integration
* Primary NODE (Writes) and Replicas (Read)
* Nodes are HA .. Primary failure = election
* In-Memory cache – Scaling ..Much faster reads, reduced costs
* Scale UP and Scale OUT (Bigger or More)
* Supports write-through
* DAX deployed within a VPC

**Amazon Athena**

* **Serverless Interactive Querying Service**
* **Ad-hoc queries on data** – pay only data consumed
* Schema-on-read -> table-like translation
* Original data never changed – remains on S3
* **Schema translates data** => relational-like when read
* Output can be sent to other services

**Elastic Cache**

* In-memory database … high performance
* Managed **Redis or Memcached** … as a service
* **Can be used to cache data** – for READ HEAVY workloads with low latency requirements
* Reduces database workloads (expensive)
* **Can be used to store Session Data** (Stateless Server)
* Requires application code changes

|  |  |
| --- | --- |
| **Memcached** | **Redis** |
| Simple Data Structure | Advanced Structures |
| No Replication | Multi AZ |
| Multiple Nodes (Sharding) | Replication (Scale Reads) |
| No backups | Backup and Restore |
| Multi-threaded | Transactions |

**Redshift**

* Petabyte-scale Data warehouse
* OLAP (column based) not OLTP (row/transaction)
* Pay as you use …similar structure to RDS
* Direct Query S3 using Redshift Spectrum
* Direct Query other DBs using federated query
* Integrates with AWS tooling such as Quicksight
* SQL-like interface JDBC/ODBC connections
* Server based (not serverless)
* It is cluster architecture
* One AZ in a VPC – network cost/performance
* Leader Node – Query input, planning and aggregation
* Compute Node – performing queries of data
* Compute Node has Slices which are connected to storage
* Data is replicated to 1 additional node
* Automatic snapshot to S3(8 hours, 5GB) with retention. Manual snapshot to S3 is managed by administrator
* VPC Security, IAM Permissions, KMS at rest Encryption, CW Monitoring
* **Redshift Enhanced VPC Routing** – VPC Networking
* DMS can migrate data into Redshift
* Firehose can stream data into Redshift

**Resiliency and Recovery**

**Serverless**

**Monlithic Architecture**

* Fails Together
* Scales Together
* Bills Together

**Tiered Architecture**

* We can use internal LB so that modules through LB rather than communicating directly
* Can be scaled separately vertically and Horizontally
* But it still with this architecture modules expects each other to respond. Eg: “Upload” expects “Processing” to respond
* Another draw back is modules needs to be running continuously even if there is no workload. Eg: “Processing” needs to be running for “Upload” to function

**Queue Based Architecture**

* Tiers can communicate asynchronously
* It becomes loosely coupled now
* Modules can scale up and down (using ASG) based on the messages in the queue. E.g. “Processing” can scale up and down using ASG based on the messages in Queue

**Microservice Architecture**

* There can be multiple small services.
* Large number of microservices can become complex very quickly.
* If all those services need to communicate with each other then there will be lots of queues.

**Event Driven Architecture**

* **Event Producer** – It can be anything which produces events
* **Event Consumer** – Other services or component waiting for events to Consume.
* **Event Router or Event Bus** – Producer can send the events to router. Router can put that event on bus ..which in turn can be consumed by Consumer
* No constant running or waiting for things
* Producers generate events when something happens (Clicks, errors, criteria met, uploads, actions etc)
* Actions are taken and the system returns to waiting
* Mature event-driven architecture only consumes resources while handling events (serverless)

**AWS Lambda**

* Function-as-a-Service (FaaS)
* Event Driven invocation (execution)
* Lambda function = piece of code in one language
* Lambda function use a runtime (e.g. Python 3.6)
* Runs in a runtime environment
* You are billed only for the duration a function run
* Key component of serverless architecture
* Event-Driven or manual invocation e.g. AWS Service, Customer via API Gateway or even Scheduled
* Currently 15-minute execution limit
* New runtime environment every execution – no persistence
* Execution Role provides permissions
* Load data from other services (e.g. S3)
* Store data to other services (e.g. S3)
* (free tier) 1M free requests per month and 400000GB seconds of compute time per month

**CloudWatchEvents and EventBridge**

* If X happens, or at Y times … do Z
* EventBridge is CloudWatch Events v2
* A default Event Bus for the account
* In CloudWatch Events this is the only bus (implicit)
* EventBridge can have additional event buses
* Event Router route the event to right target
* Rules match incoming events ..or (schedule events)
* Route the events to 1+ targets ..e.g. Lambda

**API Gateway**

* API Gateway is a managed API Endpoint Service
* Create, Publish, Monitor and Secure APIs as a Service
* Billed based on Number of API calls, Data Transfer and additional performance features such as caching
* Can be used directly for serverless architecture
* Or during a architecture evolution

**Serverless Architecture**

* Serverless isn’t one single thing
* You manage few, if any servers – low overhead
* Applications are a **collection of small and specialized functions**
* **Stateless and Ephemeral** environments – duration billing
* **Event-driven** …consumption only when being used
* **FaaS is used** where possible **for compute functionality**
* Managed services are used where possible

**Simple Queue Service (SQS)**

* Public, Fully Managed, Highly Available Queues – Standard or FIFO
* FIFO ensures the ordering, however, there is no such guarantee with Standard
* Standard = at-least-once, FIFO = exactly-once
* Messages up to 256 KB in size – link to large data
* Received messages are hidden (VisibilityTimeout)
* Once the VisibilityTimeout expires then either reappear (retry) or explicitly deleted
* Dead-Letter queues can be used for problem messages
* ASGs can scale and Lambdas invoke based on queue length
* FIFO 3000 messages per second with batching, or up to 300 messages per second without
* Billed based on requests
* 1 request = 1-10 messages up to 64KB total
* Short (immediate) vs Long (waitTimeSeconds) polling
* Encryption at rest (KMS) and in-transit
* Queue Policy…

**Simple Notification Service (SNS)**

* Public AWS Service – network connectivity with public Endpoint
* **Coordinates the sending and delivery of messages**
* **Messages are <= 256 KB** payloads
* SNS Topics are the base entity of SNS – permissions and configuration
* **A Publisher sends message to a TOPIC**
* **TOPICS have Subscribers which receive messages**
* E.g. HTTP(s), Email (JSON), SQS, Mobile Push, SMS Messages and Lambda
* SNS used across AWS for notifications – e.g. CloudWatch & CloudFormation
* **Fanout architecture** in SNS
* Delivery Retries – Reliable Delivery

**Step Functions**

**Problems with Lambda**

* Lambda is FaaS
* 15-minute max execution time
* Can be chained together
* Gets messy at scale
* Runtime Environments are stateless

**Step Function**

* Step Functions are like State Machine
* Serverless workflow …Start-> States -> End
* States are THINGS which occur
* Maximum Duration 1 year
* Standard Workflow and Express Workflow
* Started via API Gateway, IOT Rules, EventBridge, Lambda
* Amazon States Language (ASL) – JSOn Template
* IAM Role is used for permissions

**States**

* + Succeed & Fail
  + Wait
  + Choice
  + Parallel
  + Map
  + Task (Lambda, Batch, DynamoDB, ECS, SNS, SQS, Glue, SageMaker, EMR, Step Functions)

**Kinesis Stream**

* Kinesis is a scalable streaming service
* Producers send data into kinesis stream
* Streams can scale from low to near infinite data rate
* Public service & highly available by design
* Streams store a 24-hour moving window of data – can be increased to 7 days for additional cost
* Multiple consumers access data from that moving window
* Kinesis scale using the Shard architecture
* More shard a kinesis has, more expensive it is and more performance it has
* Designed for Data ingestion, Analytics, Monitoring, App Clicks etc..

**Kinesis Firehose**

**Global Content Delivery and Optimization**

**CloudFront**

* CloudFront is a **global object cache** (CDN)
* Content is **cached in locations** close to customers
* Lower latency and higher throughput
* Load on the content server is decreased
* It can handle **static and dynamic content**
* Integrates with ACM and HTTPS
* Upload direct to origins – No caching

**Origin –** The source location of your content

**Distribution –** The configuration unit of CloudFront

**Edge Location –** Local infrastructure which host a cache of your data

**Regional Edge Cache –** Larger version of an edge location. Provides another layer of caching.

**AWS Certificate Manager (ACM)**

* HTTP – Simple and Insecure
* HTTPS – SSL/TLS layer of Encryption added to HTTP
* Data is encrypted in-transit
* Certificates prove identity
* Signed by a trusted authority
* **Create, renew and deploy certificates** with ACM
* Supported AWS Services only (e.g. CloudFront and ALB)

**Lambda@Edge**

* You can run lightweight Lambda at edge locations
* Adjust data between the Viewer and Origin
* Currently supports Node.js and Python
* Run in the AWS Public Space (Not VPS)
* Layers are not supported
* Different Limits vs Normal Lambda functions
* Use Cases
  + A/B testing – Viewer Request
  + Migration between S3 origins – Origin Request
  + Different objects based on device – Origin Request
  + Content by Country – Origin Request

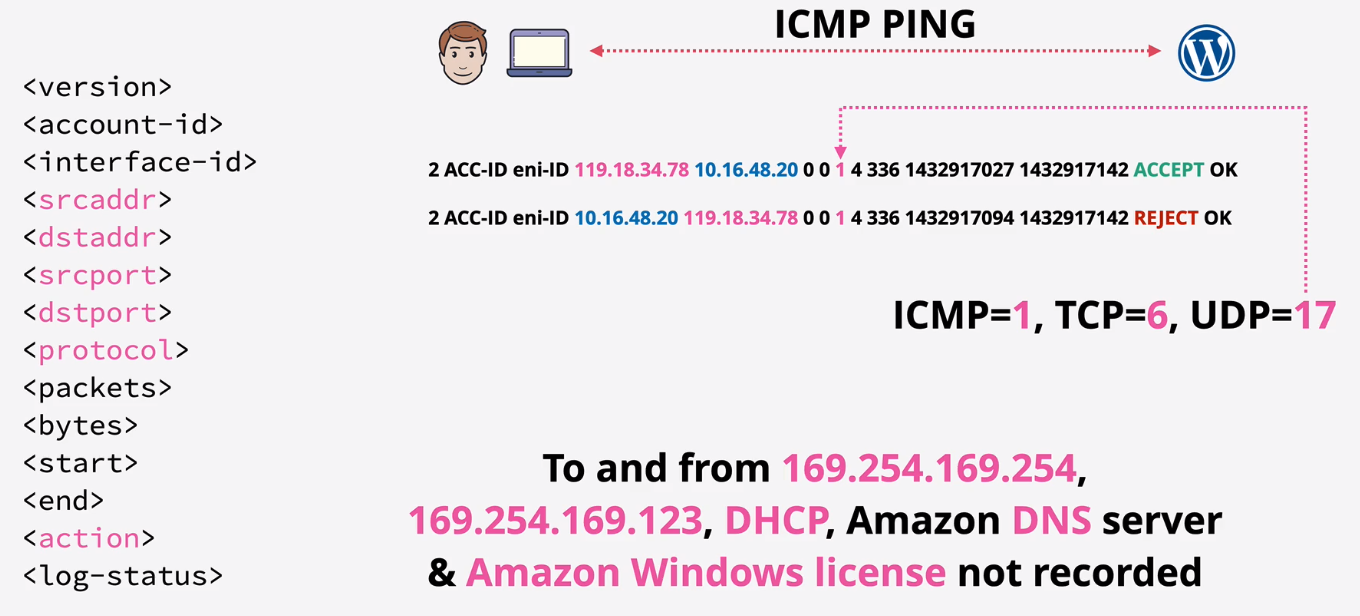
**Global Accelerator**

* Anycast IP’s allow a single IP to be in multiple locations. Routing moves traffic to closest location
* Traffic initially uses public internet and enters a Global Accelerator edge location
* From the edge, data transits globally across the AWS global backbone network. Less hops, directly under AWS control, significantly better performance
* Moves the AWS network closer to customers
* Connections enter at edge using the anycast IPs
* Transit over AWS backbone to 1+ location
* Can be used for Non-HTTP/S (TCP/UDP) – Different from CloudFront

**Advanced VPC**

**VPC Flow Logs**

* Capture packet Metadata …Not packet contents
* Applied to a VPC – All interfaces in that VPC
* Applied to a Subnet – interfaces in that Subnet
* Applied to an interface directly
* VPC Flow Logs are not real-time
* Destination can be S3 or CloudWatch Logs



**Egress-Only Internet Gateway**

* With IPv4 addresses are private or public
* NAT allows private IPs to access public network but without allowing externally initiated connections IN
* With IPv6 all IPs are public
* Internet Gateway (IPv6) allows all IPs IN and OUT
* Egress-Only is outbound-only for IPv6

**Gateway Endpoints**

* Provide private access to S3 and DynamoDB
* Prefix List added to route table => Gateway Endpoint
* Highly Available (HA) across all AZs in a region by default
* Endpoint policy is used to control what it can access
* Regional can’t access cross-region services
* Prevent leaky Buckets – S3 Buckets can be set to private only by allowing access only from a gateway endpoint

**Interface Endpoints**

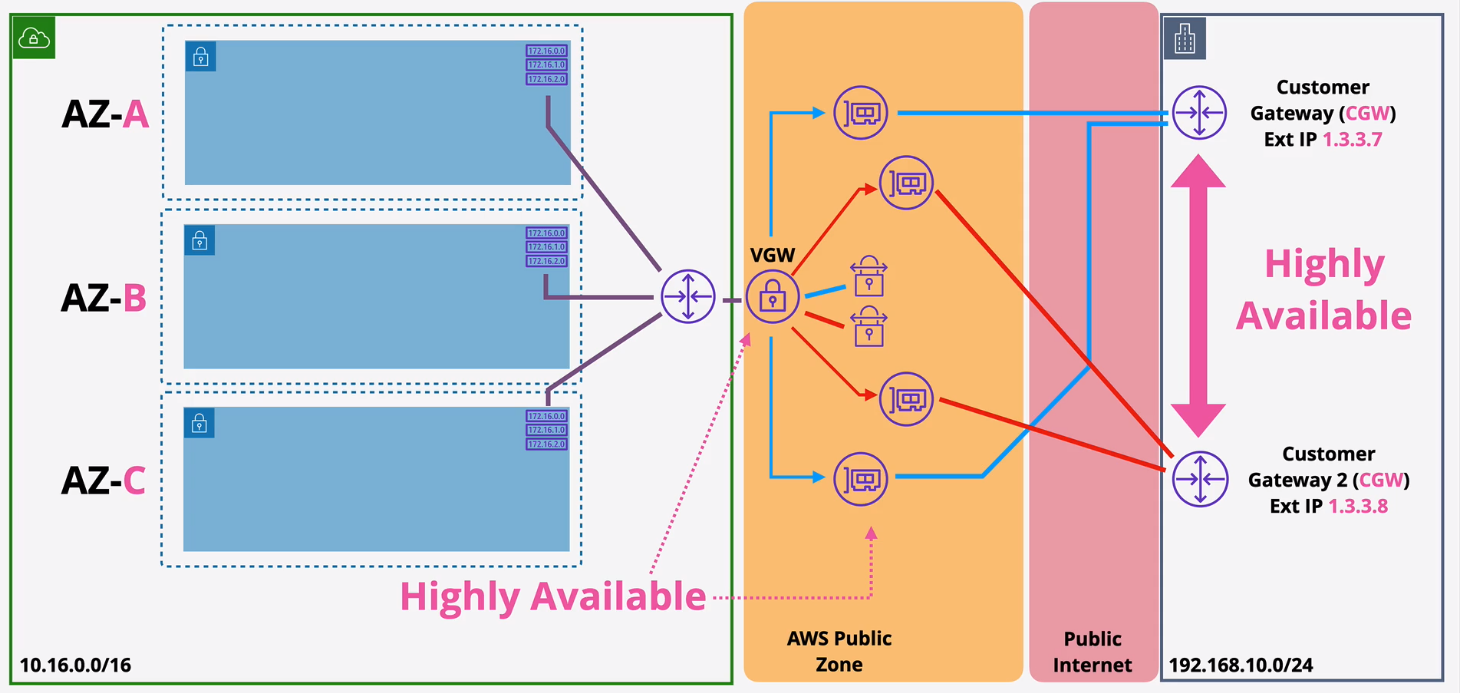
* Provide private access to AWS Public Services except S3 and DynamoDB
* Added to specific subnets – an ENI – Not HA
* For HA …add one endpoint, to one subnet, per AZ used in the VPC
* Network access controlled via Security Groups
* Endpoint policies – restrict what can be done with the endpoint
* TCP and IPv4 only
* Uses PrivateLink
* Endpoint provides a New service endpoint DNS
* Endpoint Regional DNS
* Endpoint Zonal DNS
* Applications can optionally use these or
* PrivateDNS overrides the default DNS for services

**VPC Peering**

* Direct encrypted network link between two VPCs
* Works same/cross-region and same/cross-account
* (optional) Public Hostnames resolve to private IPs
* Same region SG’s can reference peer SG’s
* VPC peering doesn’t support transitive peering
* Routing Configuration is needed, SGs & NACLs can filter

**Site-To-Site VPN**

* A logical connection between **a VPC and on-premise** network encrypted using IPSec, running over the public internet
* Full HA – if you design and implement it correctly
* Quick to provision … less than an hour
* **Virtual Private Gateway** (VGW)
  + It is outside the VPC, in AWS public space
  + Has physical endpoints in different AZ so by design it’s HA
* Customer Gateway (CGW)
  + Can be made HA, by creating CGW on a ideally on different site
* VPN connection between the VGW and CGW
* VPN tunnel gets created between VPN and VGW endpoints and VGW endpoints and CGW
* Configure the route path in VPC and VPN so that they both can send the traffic to each other
* The traffic flow through the VPC using VPC router -> VGW -> VPN -> VGW Endpoints -> CGW
* **Considerations**
  + Speed Limitation of 1.25 Gbps
  + Latency Consideration – inconsistent, public internet
  + Cost – AWS hourly cost, GB out cost, data cap
  + Speed to setup – hours
  + Can be used as a backup for Direct Connect (DX)
  + Can be used with Direct Connect (DX)



**Difference between Static vs Dynamic VPN**

|  |  |
| --- | --- |
| **Static** | **Dynamic** |
| Routes for remote side added to route tables as static routes | Routes for remote side added to route table as static routes. Route Propagation (if enabled) means routes are added to route tables automatically |
| Networks for remote side statically configured on VPN connection | Border Gateway Protocol (BGP) is configured on both customer and AWS side using (ASN). Networks are exchanged via BGP |
| No load balancing and multiconnection failover | Multiple VPN connections provide HA and traffic distribution |

**Direct Connect (DX)**

* A 1 Gbps or 10 Gbps Network port into AWS
* …at a DX location (1000-Base-LX or 10GBASE-LR)
* …to your Customer Router (requires VLAN/BGP)
* …or Partner Router (if extending to your location)
* Multiple Virtual Interfaces (VIFS) over one DX
* Private VIF (VPC) and Public VIF (Public Zone Services)
* Take much longer to provision vs VPN
* DX Port provisioning is quick …the cross-connect takes longer
* …extension to premises can take weeks/months
* Use VPN first then replace with DX (or leave as backup)
* Faster 40 Gbps with aggregation
* Low consistent latency, doesn’t use business bandwidth
* No Encryption

**Transit Gateway**

* **Network Transit Hub to connect VPCs to on premises networks**
* Significantly **reduces network complexity**
* Single network object – **HA and Scalable**
* Attachments to other network types
* VPC, Site-to-Site VPN & Direct Connect Gateway
* VPC attachments are configured with a subnet in each AZ where service is required
* Supports transitive routing
* Can be used to create global networks
* Share between accounts using AWS RAM
* Peer with different regions …same or cross account
* Less complexity vs w/o TGW

**Storage Gateway**

* Hybrid Storage Virtual Appliance (On-premises)
* Extension of File and Volume storage into AWS
* Volume storage backups into AWS
* Tape backups into AWS
* Migration of existing infrastructure to AWS

There are modes in which Storage Gateway works

**Tape Gateway (VTL) mode**

* Pretends to be an iSCSI tape library, changer and drive
* Virtual tapes => S3 and Glacier
* Active tapes are stored in S3, archived/exported tapes are stored in VTS in Glacier
* Unlimited VTS (Archive) Storage

**File Gateway Mode**

* Files are stored to share using **SMB and NFS**
* SMB shares can integrate with active directory for file authorization
* File Storage are backed by S3 Object
* Lifecycle policies can automatically control storage class

**Volume Gateway Mode**

* Primary data is stored on-premises
* Backup data is asynchronously replicated to AWS
* **AWS creates EBS snapshot from backup data**. Can be used to create standard EBS Volumes. Ideal for migration to AWS
* Gateway Cache/Stored – iSCSI
* Volumes are made available via iSCSI for network-based servers to access (single connection per volume unless server are clustered)

**Volume Gateway (Cached) Mode**

* Primary data is stored in AWS.
* Data which is accessed frequently is cached locally. Ideally for extending storage into AWS.
* Primary data is stored on a S3-Backed volume. Snapshots are stored as standard EBS snapshot
* Volumes are made available via iSCSI for network-based servers to access (single connection per volume unless server are clustered)
* Block storage backed by S3 and EBS snapshot

**Snowball**

* Ordered from AWS, Log a Job, Device delivered not instant
* Data encryption uses KMS
* 50TB or 80TB capacity
* 1Gbps network
* 10TB to 10PB economical range
* Multiple devices to multiple premises
* Only storage

**Snowball Edge**

* Both storage and compute
* Larger capacity vs Snowball
* 10 Gbps network
* Ideal for remote sites or where data processing on ingestion is needed

**Snowmobile**

* Portable DC within a shipping container on a truck
* Special Order
* Ideal for single location when 10PB+ is required
* Up to 100PB per snowmobile
* Not economical for multi-site (unless huge) or sub 10PB

**Directory Service**

* AWS managed implementation
* Runs withing a VPC
* To implement HA…deploy into multiple AZs
* Some AWS services need a directory e.g. Amazon Workspaces
* Can be isolated or integrated with existing on-premises system or acts as a proxy back to on-premises
* **Simple AD** – The default. Simple requirements. A directory in AWS.
* **Microsoft AD** – Applications in AWS which need MS AD DS or you need to Trust AD DS
* **AD Connector** – Use AWS services which need a directory without storing any directory info in cloud…proxy to your on-premises directory.

**Data Sync**

* Data Transfer service TO and FROM AWS
* Migrations, Data Processing Transfers, Archival/Cost Effective Storage or DR/BC
* Designed to work at huge scale
* Keeps metadata (e.g. permissions/timestamps)
* Built in data validation
* Scalable – 10 Gbps
* Bandwidth Limiters
* Incremental and scheduled transfer options
* Compression and encryption
* Automatic recovery from transit errors
* AWS Service integration – S3, EFS, FSx
* Pay as you use … per GB cost for data moved

**FSx Windows Servers**

* Fully managed native windows file servers/shares
* Designed for integration with windows environments
* Integrates with Directory Service or Self-Managed AD
* Single or Multi AZ within a VPC
* On-demand and schedule backups
* Accessible using VPC, Peering, VPN, Direct Connect
* VSS – User Driven Restores
* Native file system accessible over SMB
* Windows permission model
* Supports DFS …scale out file share structure
* Managed – no file server admin
* Integrates with DS and Your own directory

**FSx for Lusture**

* Managed Lusture – Designed for HPC – Linux Clients (POSIX)
* Machine Learning, Big Data, Financial Modelling
* 100 GB throughput and sub millisecond latency
* Deployment types – persistent or scratch
* Scratch – Highly optimized for short term no replication and fast
  + Scratch is designed for pure performance
  + Short term or temp workloads
  + No HA or No Replication
* Persistent – Longer term, HA(in one AZ), self-healing
  + Larger file system means more servers, more disks and more chance of failure
  + Persistent has replication with one AZ only
  + Auto-heal when hardware failure occurs
* Accessible over VPN or Direct Connect
* You can backup to S3 with both (Manual or Automatic 0-35 days retention)

**AWS Secret Manager**

* It does **share functionality with Parameter Store**
* **Designed for secrets** (…passwords, API Keys)
* Usable via Console, CLI, API or SDK’s (integration)
* **Supports automatic rotation** …this uses lambda
* Directly integrates with some AWS products
* Secrets are encrypted using KMS

**AWS Shield**

* Provides AWS resources with **DDoS protection**
* **Shield Standard** – free with **Route53** and **CloudFront**
* Protection against **Layer 3 and Layer 4** DDoS attacks
* **Shield Advanced** - $3000 pm
* EC2, ELB, CloudFront, Global Accelerator & R53
* DDoS Response Team & Financial Insurance

**Web Application Firewall (WAF)**

* **Layer 7 (HTTP/s) Firewall**
* Protects against **complex Layer 7 attacks/exploits**
* **SQL Injections, Cross-Site scripting, Geo Blocks, Rate Awareness**
* **Web Access Control List (WEBACL)** integrated with **ALB, API Gateway and CloudFront**
* Rules are added to a WEBACL and evaluated when traffic arrives

**Cloud HSM**

* With KMS …AWS Manage …Shared but separated
* True **Single Tenant Hardware Security Module** (HSM)
* AWS provisioned …fully customer managed
* Fully FIPS 140-2 **Level 3** (KMS is L2 Overall, some L3)
* Industry Standard APIs – PKCS#11, Java Cryptography Extensions (JCE), Microsoft CryptoNG (CNG) libraries
* **KMS can use CloudHSM as a custom key store**, CloudHSM integration with KMS
* No native AWS integration e.g. no S3 SSE
* Offload the SSL/TLS processing for web servers
* Enable Transparent Data Encryption (TDE) for Oracle Databases
* Protect the Private Keys for an Issuing Certificate Authority (CA)

**Amazon Lex**

- Text or Voice conversational interfaces

- Powers the Alexa service

- Automatic Speech Recognition (ASR) - Speech to Text

- Natural Language Understanding (NLU) - Intent

- Build understanding into your application

- Scales, Integrates, Quick to deploy, Pay as you go pricing

- Chatbots, Voice Assistant, Q&A bots, Info/Enterprise Bots

**Amazon Connect**

- Contact Center ..as a service

- Omnichannel - voice and chat, incoming & Outgoing

- Integrates with PSTN networks for traditional voice

- Agents can connect using the internet from anywhere

- Integrates with other AWS services (Lambda/LEX) for additional intelligence and features

- Quick to provision, pay as you use pricing and scalable

**AWS Glue**

- Serverless ETL (Extract, Transform and Load)

- vs Datapipeline (which can do ETL) and uses EMR

- Moves and Transform data between source and destination

- Crawls data sources and generates the AWS Glue Data Catalog

- Data Source: Stores: S3, RDS, JDBC Compatible and DynamoDB

- Data Source: Streams: Kinesis Data Stream and Apache Kafka

- Data Targets: S3, RDS, JDBC Databases

**AWS Glue - Data Catalog**

- Persistent metadata about data sources in region

- One catalog per region per account

- Avoids data silos

- Amazon Athena, Redshift Spectrum, EMR, and AWS Lake Formation all uses Data Catalog

- configure crawlers for data sources

**Migration**

**Discover, Assess and Priorities** applications to Migrate

* **Rehosting** – Lift and Shift
* **Replatforming** – Lift and Shit with Optimization
* **Repurchasing** – Move to something new e.g. SaaS
* **Refactoring/Rearchitecting** – Take advantage of Cloud
* **Retire** – Do we even need this? No? Dump it
* **Retain** – Not worth the time/money or is too scary to migrate

**Rehosting**

* Lift and Shift – Migrate as is (App Virtual Machine => EC2, DB=> EC2)
* + Reduced Admin Overhead (IaaS)
* + Potentially easier to optimize when in AWS
* + Cost Savings … (e.g. Using Burst Instance T3)
* – Not taking full advantage of Cloud
* – Potentially ‘Kicking the can down the road’
* VM Import/Export and Server Migration Service

**Replatforming**

* Just like rehosting but with Optimization
* You might use RDS instead of self-managed DBs
* You might use ELB instead of load balancers
* You might use S3 as backup or media storage
* No real negatives and no world-changing benefits
* Admin overhead reductions, Performance benefits, more effective backups or Improved HA/FT

**Repurchasing**

* Unless you have a reason to self-manage
* Use a XaaS Product
* MS Exchange -> Microsoft 365
* CRM -> Salesforce
* Payroll -> Xero

**Refactoring/Rearchitecting**

* Reviewing the architecture of an application
* Adopting **cloud-native** architectures and products
* Service-orientated or Microservices
* APIs, Event-Driven or Serverless
* – Initially very expensive and time consuming
* + best long-term benefits
* Often cheaper, much more scalable, better HA/FT, cost aligned with app usage

**Retire**

* Systems are often running for no reason
* Auditing their usage is often more work than leaving running
* If you don’t need the application switch it off

**Retain**

* Do Nothing
* Old application – not worth the move
* Complex application – leave till later
* Super-important application – risky
* Complete the migration – swing back to focus on the left-overs

**Application Discovery Service**

* Discover on-premises infrastructure
* Agentless (Application Discovery Agentless Connector)
* OVA appliance, integrates with VMWARE – inventory, VM usage
* Measure performance and resource usage
* Agent Based discovery for ‘in VM’ data gathering (network, processes, usage, performance)
* Dependencies between servers (network activity)
* Integration with **AWS Migration Hub** and Amazon Athena

**Server Migration Service**

* Migrate whole VM’s (OS, Data, Apps … migrated as is)
* Agentless …uses a connector which runs on-premises
* Integrates with VMware, Hyper-V and AzureVM only
* Incremental replication of live volumes
* Orchestrate multi-server migrations
* Creates AMIs which can be used to create EC2 instances
* Integrates with AWS Migration Hub

**Database Migration Service(DMS)**

* A managed database migration service
* Runs using a replication instance
* One endpoint must be on AWS
* Common DB Support MySQL, Aurora, Microsoft SQL MariaDB, MongoDB, PostgreSQL, Oracle, Azure etc.
* Replication instance performs the migration between Source and Destination endpoints which store connection information for source and target databases.
* Jobs can be Full Load (one off migration of all data), Full Load + CDC (Change Data Capture) for ongoing replication which captures changes or CDC only (if you want to use an alternative method to transfer the bulk DB data such as native tooling)
* Schema Conversion Tool (SCT) can assist with Schema Conversion

**Schema Conversion Tool (SCT)**

* SCT is used when converting one database engine to another
* Including DB-> S3 (Migration using DMS)
* SCT is not used when migrating between DB’s of the same type e.g. On-premises MySQL -> RDS MySQL
* Works with OLTP DB Types (MySQL, MSSQL, Oracle)
* And OLAP (Teradata, Oracle, Vertica, Greenplum)
* E.g. On-premises MSSQL -> RDS MySQL or On-Premises Oracle -> Aurora

**DMS and Snowball**

* Larger migrations might be multi-TB in size
* Moving data over networks takes time and consumes capacity
* DMS can utilize snowball
* Step 1 : Use SCT to extract data locally and move to a snowball device
* Step 2: : Ship the device back to AWS, They load onto an S3 bucket
* Step 3: DMS migrate from S3 into the target store
* Step 4: Change Data Capture (CDC) can capture changes, and via S3 intermediary they are also written to the target databases.

**Disaster Recovery (DR/BC)**

Effective DR/BC costs money all of the time

You need some type of extra resource

Executing a DR/BC process takes time

How long depends on the type of DR/BC

**Backup and Restore** – Data is constantly backed up at the primary site. The only costs are backup media and management. No ongoing spare infrastructure costs. Restore requires new hardware or a lengthy restore process

**Pilot Light** – A secondary environment is provisioned in advanced running only the absolute minimum of infrastructure – like a pilot light in a heater. It can be powered on much quicker than back up and restore. Critical components such as databases are always synching ready to be used.

**Warm Standby** – A smaller sized, but fully functional version of your primary infrastructure is running 24x7. Ready to be increased in size when failover required. Faster than pilot light. Cheaper than full active/active.

**Active/Active Multi Site** – Data is constantly replicated from the primary site to backup. Costs are generally 200%. A full copy is running at all the time. You can load balance across environments improving HA and performance.