EC2: Infrastructure As a Service to rent virtual machines

- Instances have underlying hypervisors of Nitro and Xem:
  - Reserved: for long term workloads (1 or 3 years) with a 75% discount but they are locked to the region they are created in
    - Convertible Reserved: change instance type over time
    - Scheduled Reserved: launch instances within specific time window without getting interrupted
    - To stop incurring charges: terminate the reserved instance ASAP to avoid getting billing at the On-Demand price when it expires OR sell it on AWS Reserved Instance Marketplace
  - On-Demand: for short, uninterrupted workloads at predictable pricing
    - Incurs charges when going from 'stopping' to 'hibernate' state
  - Spot: for short, non critical jobs that are resilient to failures and interruptions (only gives 2 minutes of notification)
    - Use as spare compute capacity (comes with steep discounts)
  - Dedicated Hosts: to book the entire physical server to control instance placement within hardware rack
    - Reduced cost for 3 year reservation
    - Use existing server bound software licenses and address compliance/regulatory requirements
  - Dedicated Instances: to run on hardware dedicated to you (isolates a single tenant hardware)
    - Share hardware with other instance types from the same AWS account (physically isolated at hardware level)
- Instance Types:
  - General Purpose: for a diversity of workloads - has a balance of CPU & RAM (t, m names)
  - Compute Optimized: for compute intensive tasks that require high performance professors (c names)
    - For tasks that require powerful CPU usage, like encoding
  - Memory Optimized: for fast performance loads that process large datasets in memory, > 700GB memory (r, i, x names)
  - Storage Optimized: for tasks that require high and sequential R/W access to large datasets on local storage
    - Has attached hard disk (1, d, h1 names)
  - Accelerated Computing: for graphics intensive GPU instances (videos) and reconfigurable FPGA instances (g, p names)
  - High Memory: for 22 TB of RAM on a single instance, only available upon specific requests (u names)
- Placement Groups are for instances to communicate with each other in a distributed fashion for faster network:
  - Cluster: hardware setup packs instances close together within 1 AZ (low latency for great network speed but high risk)
  - Spread: small groups of instances spread across different hardware (for critical applications, decreases risk of failure)
    - Maximum 7 running instances per placement group per AZ (cannot be set up across regions, only across AZs)
  - Partition: spread instances across different local partitions and set of hardware racks
    - Used by large, distributed and replicated workloads
  - Launch instances within a placement group in a single launch request with only 1 instance type
    - Launching more than one instance type within a PG OR adding more instances to the PG after it has already been launched: increases the chances of an insufficient capacity error (better to stop and restart all the instances)
- Instance States:
  - Stopping an instance shuts it down without losing the data
    - If an instance is stopped, the ENI stays attached but the EIP is disassociated (EC2-Classic) or associated (EC2 VPC)
  - Terminating an instance deletes it and cannot recover the data
  - Hibernate State: preserves in memory RAM state (saves and dumps it to a file in root EBS volume)
    - Restarting an instance is faster (OS is not stopped): only need to load the EBS volume into RAM state of the instance
  - To shut down instance but retain the data: take snapshots of the EBS volume containing the state, terminate the instance and sell the instance on Marketplace
  - Recovered Instance: identical to the original (instance ID, private IP, EIP, all metadata) and it can take over space on a placement group - If instance has IPv4 address during instance recovery, it retains the public IPv4 address
  - To update software or troubleshoot instances before returning them to service, change 'in service' state to 'standby' state
  - The 'replace unhealthy' process terminates unhealthy instances and creates new ones to replace them
- User Data Script: can bootstrap instances to launch commands when the machine first starts on boot cycle
  - Performs common configuration tasks, passes or runs shell scripts
  - By default, scripts are executed as the root user within the instances
- Instance Metadata: can retrieve the instance ID, public keys, public IP address via https://169.254.169.254
- IAM:
  - Any changes in IAM policies associated with the instance happen almost immediately
  - To attach an IAM role to an instance that currently has no role, the instance must be in stopped or running state
  - To replace an IAM role for an instance that already has an attached IAM role, the instance must be in running state
- To automate log collection from instances:
  - Add a lifecycle hook to ASG that moves instances from 'terminating' to 'terminating:wait' state to delay termination of unhealthy instances

    - Configure a CW event rule for ‘ec2-instance-terminate-lifecycle-action’ autoscaling event with lambda
    - Trigger a CW agent to push app logs and resume the instance termination once logs are sent to CW
- Create an CW alarm to monitor an instance and automatically reboot it: recommended for Instance Health Check failures
    - Instance reboot is equivalent to an OS reboot (only takes a few minutes)
    - Instance remains on the same physical host: keeps public DNS name, private IP address and any data on its instance store volume (doesn’t start a new instance billing hour)

Spot Instances and Spot Fleets
- Obtain a spot instance while the current spot price < your defined max spot price
- Use a spot block if you don't want instances to be reclaimed for a certain time frame (only 1-6 hours)
- A spot request defines the number of desired instances, the max price and the request type:
    - One-Time Request: as soon as the spot required is fulfilled, the instances are launched
    - Persistent Request: want the number of instances to be valid if within validity dates
    - Cancelling an active spot request does not terminate the associated instances
- Spot Fleets define a set of spot instance pools (unused instances with the same type, size, AZ/region) to meet target capacity by launching replacement instances after spot instances in the fleet are terminated
- Capacity Rebalancing: helps maintain workload availability by proactively augmenting Spot Fleets with a new Spot Instance before a Spot Instance receives the 2 minute interruption notice
    - When enabled, ASGs or Spot Fleets try to proactively replace Spot Instance that received a rebalance recommendation
    - Opportunity to rebalance workload to new Spot Instances (not at great risk of interruption, low chance of being reclaimed)
    - Compliments the:
        - Capacity optimized allocation strategy (finds the most optimal spare capacity)
            - AWS recommended: automatically provisions instances from the most available spot instance pools
        - Mixed instances policy (enhances availability by deploying instances across multiple types running in multiple AZs)
- If AWS intends to shut down your spot instances, possible scenarios:
    - AWS sends a notification of termination but you do not receive it within the 120 seconds and instance is shutdown
    - AWS sends a notification of termination and you receive it 120 seconds before the forced shutdown but the defined duration period (Spot blocks) hasn’t ended yet
    - AWS sends a notification of termination and you receive it 120 seconds before the intended forced shutdown
    - Cannot enable termination protection for Spot instances

AMI (Amazon Mirror Image)
- Represents a customization of an instance for faster boot/configuration time
    - Contains all information required to launch an instance from its underlying EBS volume snapshot
    - Built for a specific region but can be copied/shared across regions or with other AWS accounts
    - Launch instances from: public AWS-provided AMIs, create your own AMI, or AWS marketplace
- When an AMI is copied from region A to B, it automatically creates a snapshot in region B
    - AWS does not copy launch permissions, user defined tags, or security group rules from source to new AMI
    - After copy operation is complete, can apply all of these to the new AMI
- Copying an AMI backed by an encrypted snapshot always results in an encrypted snapshot
    - Cannot delete a snapshot of a root device of EBS volume used by a registered AMI
- Golden AMI: an AMI that is standardized through security patching and that contains agents for logging
    - Use to quickly instantiate applications: installs applications and OS dependencies
    - Launch EC2 instances from AMI (faster than AWS Beanstalk: resolves dependencies at deployment time)
- Boot from:
    - EBS backed volume: created from EBS snapshot and cannot be stopped (if host fails, all data is lost)
    - Instance Store backed volume: created from a template in S3 and can be stopped without losing data

Compute and Networking
- Enhanced Networking uses single root I/O virtualization (SR-IoV) and provides high performances networking capabilities on support instance types to give higher bandwidth/PPS with low latency and high throughput
    - No additional charge for speeding up the network
- Enabled using ENA (up to 100 Gbps) or VF interface (up to 10 Gbps)
    - ENA is a special adapter that gives enhanced networking over the internet or through interfacing with your own servers
- EFA creates a separate networking channel that allows inter-instance communication
    - Provides lower and more consistent latency with high throughout
    - ENA Enhancement: for HPC clusters, ML apps by bypassing the OS kernel to communicate directly with the EFA device

Storage and Databases

EBS (Elastic Block Storage)

- Block level storage that is used within instances while they run and uses the AWS network to communicate
  - Subset of EC2 functionality: use 'aws ec2' command to operate on EBS volumes (instance state does not affect EBS)
  - Recommended storage for instances if running a dataset on an instance
  - Attach multiple volumes to a single instance but they must all be in the same AZ
  - Persists independently from instance: can mount an old EBS volume to an instance to reload its data
  - EBS volumes of a stopped instance is preserved but billed for volumes attached to stopped instances
- Volumes are AZ locked so EBS does not work well when scaling across multiple regions (EFS is better)
  - Creating a volume in a AZ will automatically replicate it within that AZ
- Not in-memory storage (must provision capacity): storing shared data in volumes will not make apps stateless
- For data that requires frequent updates or for throughput intensive apps with continuous disk scans
  - Supports live, in production configuration changes: can modify that type, size, IOPS capacity without service interruptions

SSD	HDD
• For small and random I/O operations • Used to boot volumes where OS is going to be running • High cost • For IOPS performance	• For large and sequential I/O operations • CAN'T be used to boot a volume when creating instance • Low cost • For throughput performance

- SSD: for transactional workloads to perform small read/write operations and critical business apps that need sustained IOPS performance and large databases (has 16 TB limit)
  - General Purpose SSDs (gp2, gp3): balances price and performance
    - Maximum 10,000 IOPS per volume
    - Run on bucket credit model to calculate performance
  - Provisioned IOPS SSDs (io1, io2): highest performance for critical low latency database workloads with sustained IOPS
    - For I/O intensive workloads that are sensitive to storage and performance consistency
    - For > 32,000 IOPS (can increase IOPS independently from storage)
    - Io1 gives baseline of 50 IOPS/GB to 64,000 IOPS and 1000 MB/s for throughput/volume
    - Io2 Block Express: next generation EC2 instance for the most demanding apps that run on Nitro (up to 256,000 IOPS)
    - Multi attach (only for io1, io2): attach the same volume to multiple instances within the same AZ, where each instance has full read/write permissions to a volume
      - For high app availability in clustered Linux apps that manage concurrent write operations
- HDD: for large, streaming and sequential workloads where throughput is more important
  - Throughput Optimized ST1: low cost volumes for frequently accessed and throughput intensive workloads
  - Cold SC1: lowest cost volumes for infrequently accessed workloads and archive data (Maximum of 250 IOPS per volume)
- Magnetic Volumes: lowest cost per GB of all EBS volume types, best for infrequently accessed data
- Snapshots are incremental backups of the volume state and are used to restore the volume if the instance gets terminated
  - Stored in S3 and occur asynchronously: can use a volume while snapshot is in progress
- Encryption for EBS is region specific and is done using AWS KMS (not all volume types support encryption)
  - Allows native encryption of data while at rest: the user can configure encryption at rest using AWS KMS or CMKs
    - Use custom symmetric CMK key for encryption (to share snapshots of encrypted volumes, share CMK)
  - To encrypt a volume: create a snapshot of the volume, encrypt the snapshot with the copy function, create a new EBS volume from the snapshot and attach the encrypted volume to the original instance
  - To copy an encrypted volume to a new region: create a snapshot of the volume and copy the snapshot to the new region while specifying keys
  - Always encrypted:
    - Any snapshot created from an encrypted volume
    - Any data at rest inside an encrypted volume
    - Any data moving between an EBS volume and an instance
  - To encrypt data at rest inside EBS volumes:
    - Use third party volume encryption tools
    - Encrypt data using native encryption tools available in OS
    - Encrypt data inside your application, before storing it on EBS
- 'DeleteOnTermination' attribute determines if to preserve or delete the volume when an instance terminates
  - By default, attribute is set to True for the root volume and is set to False for all other volume types

    - An EBS root volume will persist independently from the life of the terminated EC2 instance to which it was previously attached to if 'DeleteOnTermination' is set to False
- RAID: used to increase performance/reliability of data storage (consists of 2 or more drives working in parallel)
    - RAID0: increases performance (IOPS), increases risk (combines total disk space on one)
        - Leverages volumes in parallel to linearly increase performance, while accepting risks
    - RAID1: increase reliability and fault tolerance (decreases risk by writing to both volumes at a time)
- When you launch an instance, the root device volume contains the image used to boot the instances:
    - By default, the root volume for AMI that is backed by EBS is deleted when the instance terminates
        - Can be changed to ensure that the volume persists after the instance terminates
        - Non root EBS volumes remain available even after you terminate the instance
- EBS-backed instance: can be stopped and later restarted without affecting data stored in the attached volumes
    - By default, the root volumes for an AMI backed by EBS is deleted when the instance terminates
        - Change this behavior to ensure that the volume persists after the instance terminates by setting the DeleteOnTermination attribute to false using a block device mapping
- EBS volume be deleted if it is the root device of a registered AMI while that AMI is in use: need to register the AMI, remove the AMI and then delete the EBS volume and its snapshot

EC2 Instance Store
- A virtual machine attached to a real hardware server and serves as a high performance cache for short term buffer/scratch storage for apps without shared data that are not at risk for data loss
- Requires IOPS of 210,000 for the underlying file system
- If the instance is stopped or terminates, the instance store storage is lost (ephemeral storage)
    - Lost if: underlying disk drive fails, instance stops or terminates, instance is terminated, hardware disk failure

EFS
- A managed, Linux-only network file system that can be mounted on many instances across multiple AZs
    - Highly available, scalable but expensive service
    - EFS-IA: storage class for files that are not accessed everyday (92% cheaper)
- To store shared software updates that should be dynamically loaded without heavy operations across 100s of instances: store the updates on EFS and mount a network drive to attach to all the instances
- Allows native encryption of data while at rest: the user can configure encryption at rest using AWS KMS or CMKs
- Performance Modes:
    - General Purpose (default): for many small files that need to be accessed quickly (high latency)
    - Throughout Mode: for bursting and provisioned files (increases throughput)
    - Max I/O: to scale to higher levels of aggregate throughput and OPS (best for big data operations)
        - Tradeoff: high latency for metadata operations
- To restrict access so that only permissed EC2 instances can read from EFS file system:
    - Use EFS Access Points to manage application access
    - Attach an IAM Policy to the file system to control clients who mount the system
    - Use security groups to control network traffic to and from the file system

FsX
- EFS version for Windows: single/multi AZ deployment, backups, no licensing fees, removes redundant data
- Use as a fully managed, file system that shared storage space that multiple applications can access in parallel

Pricing: S3 Standard < text file on EBS < text file on EFS

EBS	EFS	Instance Store
• Attached to 1 instance at a time • Migrate volumes across AZs via snapshots • Can't run backups while app is handling traffic	• Attached to 1000s of instances across multi AZs • Regional service, Linux only • More expensive	• Get the maximum amount of I/O from EC2 instances • Lose storage if instance is lost

Lustre (Linux + Cluster)
- Parallel, hot storage, distributed file system for high performance, large scale computing for ML or HPC
- Cluster with shared file system that stores data across multiple network file servers

Elasticache

- Managed Redis and MemCached data cache: in-memory, key/value store with high performance with low latency
  - Reduce load from database for frequent read/write intensive workloads
  - Store session data to make applications stateless (ensure user stays logged in without stickiness)
  - Does not support IAM authentication (authenticate Redis using AUTH)
- Query to see if query is already made (cache hit) or if need to read from DB and write to cache (cache miss)
  - Cache must have an invalidation strategy to ensure only current data is used and stored
- Redis: multi-AZ with automatic failover, has data durability with persistence and sub millisecond latency
  - Read replicas are used to scale reads for high availability (data resides in the servers main memory)
  - For real time transactional processing: supports cluster sharding, geospatial spread, media streaming
  - AUTH enable Redis to require an AUTH token before allowing clients to execute commands
  - Offers a native encryption service: secures session data by granting permission to users using AUTH by creating a new Redus cluster that can enable transit encryption and authenticate token parameters
  - Can offer operations: pub/sub, sorted sets and in-memory data store (doesn't support multithreaded architecture)
- MemCached: multi node partitioning of data (sharding) used for compatible in-memory key/value stores
  - No replication, no high availability, not persistent: just very simple cache to offload database
  - Does not offer a native encryption service
- Patterns:
  - Lazy Loading: all read data is cached but the data can become stale in the cache
  - Write Through: add or update data in the cache when it is written to a DB (ensures no stale data)
  - Session State: store temporary session data in a cache using TTL features

RDS

- Managed SQL relational DB service for online transactional processing (OLTP) but must provision storage
  - SQL Server, Oracle, MySQL (port 3306), PSQL, MariaDB, Aurora
  - Create RDS for SQL server DB instances with up to 16 TB of storage
  - Select the AZ to create a single AZ RDS instance
  - Cannot SSH into RDS: it is deployed within a private subnet, leverages SGs attached to the instance
  - Use RDS with provisioned IOPS SSD to implement SQL based RDS for high performance workloads
  - Provision in multiple AZs in order to achieve high availability
- Read Replicas: scale reads by replicating all DBs in the primary instance to serve traffic
  - Used for scaling compute or I/O capacity for read heavy DB workloads and business reporting quieres
  - No extra charge for primary to secondary data replication
  - Must have automatic backups turned on to deploy a RR
  - Create up to 5 RRs for a single instance (slow replication process)
  - Each RR has its own DNS endpoint: connects using DNS address
  - Replicas can be promoted to the master DB: must update connection string to leverage list of all RRs
  - A write to the DB will always be replicated to the replica (no automatic failover: create new connection string and update instances to point to the new RR)
- Multi AZ: increase DB availability in case of failover or system updates to ensure HA (automatic failover)
  - Instance connects to the DB using a connection string (1 DNS address) that points to the primary DB
  - If the primary instance is lost, the DNS is automatically redirected to the same point in the secondary DB (keeps the same connection string)
  - Data transferred between AZs for replication of multi AZ deployment is free
  - Cannot use the standby database to offload reads from an application: is it only there for failover

Read Replicas (for performance)	Multi AZ (for disaster recovery)
● Asynchronous replication (eventually consistent) ● All RRs are accessible (read scaling) ● Cross AZ or cross regions ● By default, no backup configuration ● Manually promoted to standalone DB instance	● Synchronous replication (immediately consistent) ● Durable but expensive ● Only DB engine on the primary instance is active ● Automated backups from standby ● Spans 2 AZs with 1 region auto failover

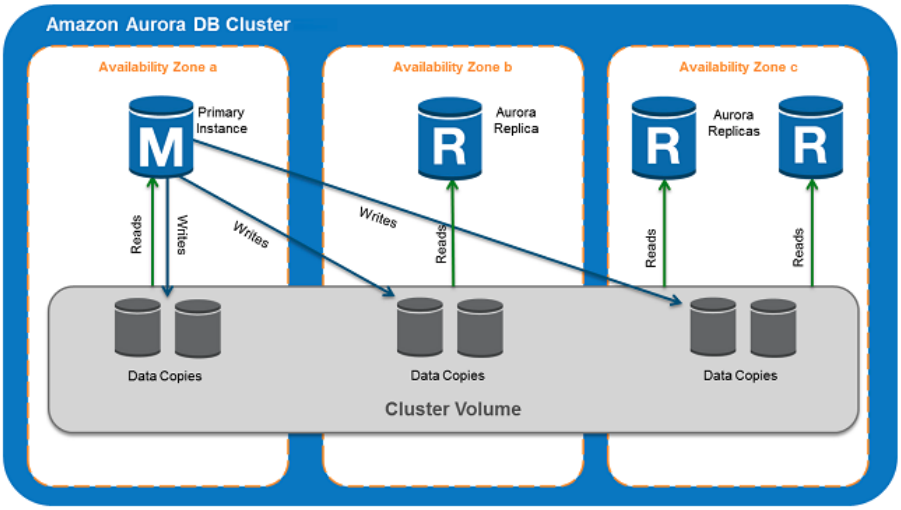
- Automated Backups: recover to any point in time (down to seconds) within a retention period of 1-35 days
  - By default, enabled to take a full daily snapshot and store transaction logs during the day
  - During a recovery, AWS first chooses the most recent daily backup (which is stored in S3 where the storage space equals the size of the DB) and then the daily transactional log

    - Backups are taken within a defined window but are deleted when the DB is deleted
        - Changes to backup window are implemented: immediately or on next scheduled maintenance
    - During the backup window: storage I/O may be suspended and may experience elevated latency
- Failover: RDS automatically flips CNAME for DB instance to point to the standby (same connection string URL)
    - Auto failover if: loss of availability, loss of network connectivity or storage failure in primary AZ or compute unit
        - Handles by RDS: flips CNAME for DB instance to point at the standby (promoted to become the primary)
    - Enabled Multi-AZ: If an outage of DB instance happens, RDS automatically switches to a standby replica in another AZ
        - Force a failover manually when you reboot a DB instance
    - The endpoint for DB instance remains the same after a failover so the application can resume database operation without the need for manual administrative intervention
- Snapshots: done manually (user initiated) and stored even after the original RDS instance is deleted
    - When restoring from a snapshot or an automated backup: the restored version of the DB will be a new RDS instance with a new DNS endpoint (does not restore to the same RDS instance)
- Scaling: to scale an RDS instance that consists of MySQL database to meet anticipated traffic:
    - Add an RDS Read REplica for increased read performance
    - Use Elasticache to cache the frequently read, static data
    - Modify the DB instance SSD storage settings for Provisioned IOPS
- Auto-Scaling Storage: increases storage on RDS instance dynamically (enable and set max storage threshold)
    - Happens when: free available space is less than 10% of allocated storage, low storage condition lasts at least 5 minutes and at least 6 hours have passed since the last storage modification
- Encryption: happens at rest and only at creation using KMS (If primary is encrypted, the RRs are encrypted)
    - To encrypt RDS backup: take a snapshot of the unencrypted DB, use copy function to encrypt it, restore the DB from the encrypted snapshot and terminate the old DB
    - To secure in flight data between servers and RDS: force all connections to the DB instance to use SSL (reboot instance), download RDS root CA certification, import the certification to the servers and configure the applications to use SSL
        - Ensure network traffic to/from RDS DB is encrypted (SSL) using profile credentials specific to the EC2 instance: set up RDS DB and enable IAM DB authentication
- Only MySQL and PSQL can be configured with IAM DB authentication
    - To secure access to RDS MySQL that is used by multiple apps:
        - Each IAM user must use a short lived authentication token to connect to the DB
        - Use IAM DB authentication to create DB accounts using AWS provided AuthPlug to MySQL
    - To secure access to RDS PSQL used by multiple users:
        - Each user should be authenticated using short lived credentials
        - Authenticate user using a token obtained through RDS PSQL service
- Other:
    - Use enhanced monitoring to monitor how different processes use the CPU
    - For an app to check RDS for an error, look for Error Node in the response from RDS API

Aurora

- Managed relational (MySQL, PSQL) DB engine for OLTP transactional processing
    - Stores a minimum of 2 copies of data across 3 AZ (6 copies in total)
    - Starts with 10 GB and scales in 10 GB increments up to 64 TB per DB using autoscaling
    - Compute resources can scale up to 32vCPUs and 244 GB of memory
- Involves a cluster of DB instances where each connection is handled by a specific DB instance
    - When you connect to a cluster, the hostname/port that you specify points to an intermediate handler called an endpoint: Aurora uses the endpoints to abstract these connections
        - Configure multi master for an immediate failover on the write node
    - Writer endpoint (master instance) always has a DNS name pointing to it
    - Reader endpoint connects load balancers to the read replicas
        - No need to hard code the hostnames to the LB connections when DB instances not available
    - Create a custom endpoint for workloads to query a subset of replicas or to map connections to appropriate instances
- Replication happens in milliseconds and can backtrack to restore data to any point in time (high performance)
    - Each RR has a priority tier (between 0 to 15): if multiple replicas hold the same tir, choose the biggest
- Failover: only the master instance takes writes and if it fails, the failover happens in less than 30 seconds
    - Master can have up to 15 read replicas
    - In case of system failover on primary instance, Aurora tries to create a new DB instance in the same AZ as an original instance (best effort basis) and flips the CNAME to point to healthy replica

- Automated Backups: default enabled and does not impact performance (share snapshots with other accounts)
- Self Healing Scaling: handles the loss of 2 copies of data without affecting write availability and 3 copies of data without affecting read availability
- Global Aurora: allows cross region read replicas for fast disaster recovery (RPO of seconds)
    - Has 1 primary region where all R/W happens
    - Set up to 5 secondary read-only regions (up to 10 read replicas per secondary region)
    - Managed Planned Failover: for controlled environments (disaster recovery testing, operational maintenance) to relocate the primary DB cluster of the global DB to a secondary regions (RPO of 0)
    - Unplanned Failover: performs cross region failover to one of the secondary regions but RTO depends on how quickly you perform the tasks listed in recovering a global DB (RPO usually seconds)
- Aurora Serverless: has on demand scaling capacity, autoscaling configuration with auto start up/shut down
    - Simple option for infrequency, intermittent, unpredictable workloads (only pay per invocation)
    - Can save money but risky if you don't know how the app will be accessed (need to cope with unpredictable workloads)



Neptune

- Fully managed graph database for high relational data that is highly available across 3 AZs with up to 15 RRs

Elasticsearch

- Search any field (even partial matches) - best when combined with another database
- Comes with Kibana for virtualization and log stash for ingestion

S3

- Stores objects (files) into buckets (directories) where each object has a key (full path to file) and value (content)
    - For static content: charges by storage size, # of requests, storage tiers, data transfer, cross region replication
    - Cannot upload more than 5GB at a time (maximum size that can be uploaded is 5 TB)
    - To indicate a successful PUT, need MD5 checksum and HTTP 200 code
    - By default, all buckets are privates and there is no limit on the limit of prefixes in a bucket
        - Add prefixes to a bucket to increase R/W performance by parallelizing reads
- Performance:
    - Ensures eventual read consistency for PUTs and DELETEs of already existing objects
    - Immediate read access for new objects after an initial PUT request
    - Strong read after write consistency (does not affect performance)
    - At least 3500 COPY, PUT, POST, DELETE or 5500 GET, HEAD requests per second per prefix in a bucket
- Retention Period:
    - Applies to individual object versions since different versions of a single object can have different retention modes
    - Place a retention period on an object version explicitly (specify 'RetainUntilDate') or implicitly (with bucket default settings)
- Storage Options: all provide the same first byte latency
    - Standard: can sustain 2 concurrent facility failures
        - When transitioning from standard to IA/One Zone IA: object must be larger than 120 KB and it must have been stored in Standard for at least 30 days before the transition
    - Infrequent Access (IA): for data that is not often accessed but requires rapid access when needed (cheaper)
    - One Zone IA: for data that is stored in a single AZ (low latency), lower cost IA but single point of failure
    - Intelligent Tiering: automatically moves objects between access tiers based on changing patterns (with monthly monitoring)

    - Glacier: long term object storage for archive and backup (alternative to on premise magnetic tape storage)
        - Minimum of 90 days storage where each item is an archive of up to 40 TB
        - Stored in vaults for 3 retrieval options: expedited (1-5 mins), standard (3-5 hours), buly (5-12 hours)
    - Glacier Deep Archive: long term storage for a minimum of 180 days of storage
        - Stored in vaults for 2 retrieval options: standard (12 hours) and bulk (48 hours)
    - Offers 99.999999999% durability: Standard, Standard IA, One Zone IA, Glacier, Glacier Deep Archive
        - Reduced redundancy storage is the only class that doesn't offer
- Encryption:
    - In Transit (passing data between points): achieved by SSL/TLS
    - At Rest (data sitting inside S3 encrypted):
        - Server Side Encryption: encrypts data as it is being written to disk, decrypts when access is requested
            - SSE-S3 (S3 managed keys): AWS automatically manages encryption and decryption keys for you
                - Each object is encrypted with a unique key: AWS encrypted he key with a regularly rotated master key (uses one of the strongest block ciphers available)
            - SSE-KMS: AWS and you manage keys together using KMS (set header to KMS)
                - Used CMKs for separate permissions for usage of CMK (added protection)
                - Provides an audit trail to show when CMK was used and by whom
            - SSE-C (customer provided keys): give AWS your own keys to manage (AWS does not store your keys, they must be in the header of every request) and AWS manages the encryption
        - Client Side Encryption: client encrypts objects on their side, uploads encrypted object to S3 and decrypts the data themselves
        - Allows native encryption of data at rest: user can configure encryption at rest using AWS KMS or CMKs
    - Encrypt data in S3: use encryption key, copy data to S3 over HTTPS endpoint, enable SSE on S3 bucket
    - Bucket metadata is not encrypted while stored in S3: don't place sensitive data there
- Security:
    - User Based Policies: for managing permissions within account
        - Consists of IAM policies authorize API calls that access S3 objects
    - Resource Based Policies (bucket policies): for managing permissions for outside accounts (cross account access)
        - Bucket wide rules to set what principles can do what actions on buckets/objects
        - Resource owner is the account that crates buckets and objects
    - Object ACL: sets access rules at the object level
    - By default: objects are owned by the account that uploaded them (bucket owner will not implicitly have access to objects)
        - To grant the bucket owner full access to all uploaded objects: create a bucket policy that requires users to set the object ACL to bucket-owner-full-control-policy
    - To share S3 buckets across AWS accounts:
        - Use IAM and bucket policies to share entire buckets (programmatic access only)
        - Use ACLs and bucket policies to share objects (programmatic access)
        - Set up cross account IAM roles: only method that allows IAM users to access cross account S3 buckets programmatically and via the console (terminal access)
    - Restrict access to an S3 bucket: use ACL or Bucket Policies
- Versioning: When enabled, S3 stores all versions of an object (includes writes/deletes) but cannot be disabled (only suspended) - when uploading new files with versioning enabled, files will not inherit properties of previous versions
- Cross Origin Resource Sharing: a web browser based mechanism to allow requests to other origins while visiting main o
    - An origin is a schema that hosts a domain and post
    - Defines a way for client apps that are loaded in one domain to interact with resources in a different domain
    - Enable CORs on bucket to enable user web pages to make authenticated GET requests to S3 bucket
- Cross Region Replace: replicate content from one bucket in one region to another (can also be cross account)
    - Must enable versioning on source and destination buckets (they must be in different regions)
    - No pre existing data is transferred: only newly uploaded objects (deletes are not replicated)
- Pre Signed URLs: give temporary URLs to a growing list of federated users to allow them to perform actions in S3
    - Object owner of a private bucket can share objects without having to change bucket permissions to the public
    - Default expiration date of 1 hour but can extend to 7 days
    - User with URL inherits the permissions of the user who generated it
- URLs for accessing buckets:
    - Path-Style URLs (https://s3.region.amazonaws.com/bucket-name/key are supported (will be deprecated)
    - Virtual-Host-Style URLs (https://bucket-name.s3.region.amazonaws.com/key are supported (strongly recommend)
- Access Logs: Check who is trying to access objects by logging all requests that are made

- Lifecycle Rules: Automate the transition of all versions of objects between tiers
- Transfer Acceleration: Uses CloudFront network edge locations to speed up the transfer of long distance data aggregation
  - Uploads to a distinct URL for edge location (not a bucket) and then transfers over the AWS network
  - Only pay for transfers that actually accelerated the upload
- Multipart Uploads: Upload parts of the object in parallel for faster upload with improved throughout
  - Recommended when: the object size reaches 100MB or if uploading large objects over stable high bandwidth network or if uploading over a spotty network
  - Can retransmit a part without affecting other parts, begin an upload before you know the final object size, pause/resume uploads and delivers improved throughput - able to quickly recover from network issues
- S3 Select: Retrieve only a subset of data from an object using SQL based on the bucket name and key
- Event notifications can be published to SNS, SQS or Lambda
- If S3 is receiving many GET requests from users, then use CF for performance optimization: it will distribute content via cach to users for lower latency and higher data transfer rate
- When you configure your bucket as a static website, it is available at the region specific endpoint of the bucket:
  - https://bucket_name.s3-website-region.amazonaws.com (s3-website-region)
  - https://bucket_name.s3-website.region.amazonaws.com (s3-website.region)

Athena
- Serverless, SQL layer on top of S3 to query the data, output the results to S3: JSON, Apache ORC, Apache Parquet
- Integrated with Glue data log to create a unified metadata repository

Snow Family
- Secure, portable devices to collect and process data or to migrate data in/out for data preprocessing at the edge
  - Data migration uses offline devices and takes at least 1 week to transfer over the network
    - Transfer itself is done at the transfer facility
  - Edge computing processes data while it's being created at an edge location (no internet, far from cloud)
  - Snow devices can only move data from on-premise to S3 Standard: use lifecycle policy to move to storage tiers
- Snowball Edge (TB/PB): storage with intermittent connectivity, pay per transfer job to move 100s of TBs into S3
  - Storage Optimized Edge: provides 40 vCPUs of compute capacity with 80 TBs of usable blocks of S3 compatible storage (for local storage and large scale data transfers)
  - Compute Optimized Edge: provides 53 vCPUs of compute capacity with 42 TBs of usable block per object storage
    - CPU for ML processes and offers storage clustering feature
- Snow Cone: small, portable device that can withstand harsh environment for smaller transfers, Snowmobile: truck

DynamoDB
- Serverless, managed noSQL database that scales to massive workloads
  - Has schema flexibility, stored in SSD and spread across 3 distinct data centers
    - Delivers predictable performances by the use of SSDs
  - Replaces Elasticache as key/value store and document database for applications that need microsecond latency
  - Made of tables: each table has a primary key with infinite number of items per row
    - Each item per row has attributes and a max size of 400 KB
  - By default, eventually consistent reads (all copies of data is usually reached within a second)
  - To improve DB performance by evenly distributing workload: use partition keys with high cardinality attributes and a large number of distinct values for each item
    - Database is partitioned across a number of nodes: visualize each partition as an independent DB server of a fixed size, each responsible for a defined block of data (sharding)
  - Table must have provisioned R/W capacity units:
    - RCU (read capacity unit): defines throughput for reads (default is eventually consistent reads)
      - 1 RCU = 1 strongly consistent read OR 2 eventually consistent reads
    - WCU (write capacity unit): defines a throughout for writes (1 WCU = 1 write of 1KB/second)
  - Use cases: storing JSON data, storing metadata of BLOB data stored in S3, storing web session data
    - Cannot run relational joins on DynamoDB
- Cost:
  - Always a charge for provisioning R/W capacity and storage of data
  - If within a single region, no charge for transfer of data into DynamoDB or local secondary indexes
- Transactions: multiple all-or-nothing operations among multiple tables with 2 underlying read or writes (1 to prepare and 1 to commit the transaction) but it consumes more capacity

- Backups: has on-demand full backups and restore at anytime with 0 impact on table performance or availability
  - Consistent within seconds and retained until deleted: restores anytime within the last 35 days
  - Point in time recovery protects against accidental writes or deletes
- On-Demand Capacity: instantly accommodates workloads as they ramp up or down, no charge when storage is idle
  - No minimum capacity to purchase: pay per request pricing for R/W requests to only pay for what you use
  - Pay more per request with provisioned capacity so should only use for a new product launch (can't predict demand)
- Auto-Scaling: Uses the AWS Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns (must make sure it is turned out if having throttling issues)
  - Enables a table or a global secondary index to increase its provisioned read write capacity to handle sudden increases in traffic without throttling
- Security: All tables are encrypted under AMS or AWS owned CMK (doesn't automatically write to CloudTrail logs)
- Dynamo Accelerator (DAX): fully managed, in-memory cache that gives 10x performance improvement to tables
  - Reduces request time from milli to microseconds
  - Fronts dynamo tables to cache the most frequently read values to offload heavy reads on hot keys
  - Place read often items in DAX to prevent provisioned throughput exceeded exception
- Streams: Time ordered sequence of item level changes in a table that creates a log of the changes (1 day retention)
  - Each stream record represents a single data modification in the table to which the stream belongs: assigned a sequence number which reflects the order in which the record was published to the stream
    - Organized into groups per shards which acts as a container for multiple stream records and contains information required for accessing and iterating within the records
- Global Tables: managed multi master active replication solution for fast local performance
  - Based on streams: use it to enable multi region redundancy (has replication latency under 1 second)
- Data Consistency:
  - Specify reads to be eventually consistent or strongly consistent (Most up to date data)
  - Strongly consistent: returns a result that reflects all writes that received a successful response prior to the read
    - Disadvantages: read might not be available if there is a network delay or outage, has higher latency, global secondary indexes are not supported and use more throughout capacity
- Integration with EMR: Use EMR with a customized version of Hive that includes connectivity to DynamoDB to perform operations on data stored in Dynamo:
  - Loading DynamoDB data into the Hadoop Distributed File System (HDFS) and using is as input to EMR cluster
  - Querying live DynamoDB data using SQL like statements (HiveQL)
  - Joining data stored in DynamoDB and exporting it or querying against the joined data
  - Exporting data stored in DynamoDB to S3 and Importing data stored in S3 to DynamoDB

Redshift

- Data warehousing solution for PB scale data, OLAP analytics and large scale BI applications (10x better performance)
  - Parallel query execution engine to automatically distribute load across all nodes
  - Based on PSQL with an SQL interface for querying, not multi AZ
  - Uses advanced compression by compressing individual columns (via columnar data storage)
  - Billed for: compute node hours (billed 1 unit per node per hour), backups, data transfers within VPC
  - Must create separate clusters with the same input or manually restore snapshots until the cluster is delete
  - Highly available cluster needs 3 copies of data: 1 in Redshift and 2 in standby in S3
  - Load data in from Kinesis Firehose, S3 (copy command), EC2 instance (write data in batches)
- Redshift Cluster: configured with a single node (160GB) or a set of nodes (1 eader node and >=1 compute nodes)
  - Leader node manages client connections, receives and sends queries to compute nodes
  - Scale up to 128 compute nodes which store actual data, perform queries and send the data back
  - Locked down with no access by default, associate with SG to grant other users inbound access
- Backups/Snapshots: backups are enabled by default with 1 day retention period (maximum 35 days)
  - Asynchronously replicate snapshots to different regions with free storage for snapshots until cluster is deleted
  - Always tries to maintain at least 3 copies of your data
  - Will never delete manually taken snapshots: they are retained after the cluster is deleted (can accrue charges)
  - Enable cross region snapshot copy in cluster (enable copy feature for each cluster/ where to copy the snapshots to)
- Encryption: encrypted in transit via SSL and at rest using AES-256 (by default, manages all encryption keys)
- Redshift Spectrum: runs queries against large unstructured data in S3 with no loading or ETL required
  - Uses massive parallelism to execute very fast (occurs in Redshift layer) but cluster/S3 must be in the same region
- Enhanced VPC Routing: forces all traffic between clusters and data repositories through VPC
  - If not enabled, then Redshift routes traffic through the Internet

RDBMS (SQL, OLTP, Joins)	RDS, Aurora
NoSQL DB (No Joins)	DynamoDB (JSON to store session data), Elasticache (key value pairs to store session data), Neptune (Graphs)
Object Store	S3 (objects), Glacier (backups and archive)
Data Warehouse / Analytics	Redshift (OLAP), Athena (SQL)
Search	Elasticsearch (JSON)
Graphs	Neptune (show relationships)

Decoupling Applications

- When deploying applications, they can communicate:
  - Synchronously: application is directly connecting to another application (bad if there are sudden spikes in traffic)
  - Asynchronously: event based using middleware to connect applications

Amazon SQS

- Queue contains messages sent from 1 or more producers, consumers pull messages from the queue
  - Duplicate messages (at least once delivery) and out of order delivery (best effort ordering)
  - Can scale the number of instances based on the load of messages going through the queue
  - Only standard SQS queues are allowed as S3 event notification destination (FIFO not allowed)
  - approxAgeOfOldestMsg metric: ensures sensitive messages are processed in a time specific manner
  - Uses multiple hosts: each host holds a portion of all messages and a consumer process does not see all the hosts or messages so messages are not necessarily delivered in the order they were generated
- Transits any volume of data at any level of throughput without losing messages: multiple copies of every message and stored redundantly across multiple AZs (available when needed)
- Each message is short lived (default 4, max 14 days) with size limitations: once the limit is reached, messages are deleted
  - Unlimited throughout/# of messages: 120k message limit for inflight messages, 20k message limit for FIFO queue)
- Messages persist in the queue until they are deleted by consumers: a consumer can process 10 messages at a time
  - Receive messages in parallel if there are multiple consumers
- Queue Access Policies: allow cross account access or public S3 bucket event notification sent to queue
- Message Visibility Timeout: controls how long a message is invisible in the queue while it is being worked on by a processing instance (default 30 seconds): if not deleted within that time, the message becomes visible on the queue again
  - Maximum visibilityTimeout of an SQS message in a FIFO queue is 12 hours
  - Visibility timeout begins when SQS returns a message: consumer processes and delete the message
    - If the consumer fails before deleting the message and the system does not call the DeleteMessage action before the timeout expires, the message becomes visible to other consumers and is received by others
    - If a message must be received only once, consumer should delete it within the direction of visibility timeout
  - Use to prevent other consumers from receiving and processing a message while it is already being processed
    - SQS does not automatically delete the message form the queue when a consumer is processing it
- Dead Letter Queue: if the same message continues to be back on the queue, there is something wrong with it
  - Set a threshold of how many times the message is allowed to be back on the queue before going to the DLQ
- Delay Queue: delay messages so consumers don't see them immediately by 0 (default) to maximum 15 minutes
  - Postpone delivery of new messages to a queue (remains invisible for duration of delay period)
- Message Timers: Use to postpone delivery of certain messages to the queue by 0 (default) to maximum 15 minutes
  - To set delay on individual messages (rather than on entire queue), use message timers to allow SQS to use the timers DelaySeconds value instead of the delay queues DelaySeconds value
- FIFO Queue: process messages in order and send exactly once (name must end in .fifo suffix, with 80 character limit)
  - By default, supports up to 3000 messages/second with batching and 300 messages/second without batching
  - Cannot convert an existing standard queue to a FIFO queue: must create new FIFO queue or delete the existing standard queue and recreate it as a FIFO queue
- Long Polling: Allows SQS to wait until a message is available and has arrived before sending a response
  - Reduces the number of empty responses: doesn't return a response until after message arrives in queue
  - Use short polling by default which sends messages immediately, even if it is empty
  - Set receiveMsgWaitTimeSecs attribute can be 0 (short polling) or 1 (long polling)

Amazon SNS
- Publisher sends messages to topics that subscribers subscribe to to receive that message (1 sender, many receivers)
- Filter policy in subscribers to allow parallel asynchronous processing (by default, each subscriber receives every message)

SNS + SQS Fanout
- Push messages to SNS topics, receive in subscribing SQS queries: used for message filtering via subscription
- Message published to a topic is replicated and pushed to multiple endpoints to allow for parallel asynchronous processing

Amazon Kinesis
- Data Streams: real time streaming service that captures, processes and stores data streams
  - Made of numbered shards (more shards, higher throughput) where each shard gives 1000 messages/second
  - Applications send records (partition key + data blob) into the steam and consumers take the record (partition key + shard sequence number + data blob) from the stream
  - If the stream does not have enough capacity, then increase write capacity assigned to the shard table
  - By default, records of a steam are accessible for 24 hours from the time they were added (max 7 days)
  - When a host needs to send many records per second to Kinesis, use batch records and implement parallel HTTP requests to reduce overhead and ensure optimal use of shards (less expensive than increasing shards)
- Data Firehouse: stores data into target destinations (S3, Redshift, Elasticsearch, Kinesis Data Streams) by reading records, transforming them and writing to destinations (not data storage) in near real time
  - When a Data Stream is configured as a source of Firehose delivery: Firehose PutRecord and PutRecordBatchOps are disabled so the Kinesis agent cannot write to Firehose delivery stream (add data PutRecords)
- Data Analytics: reads from source (data stream or Firehose destination) and uses SQL to process data / send results
- Enhanced Fanout: multiple consumers retrieving data from a stream in parallel, receive 2MB of read throughput per shard

SQS	SNS	Kinesis
● Consumers poll data and data gets deleted ● Unlimited number of messages and consumers ● No throughput provisioning ● No guaranteed order	● Pub/sub: data is lost if not delivered ● Through topics ● No throughput provisioning ● Integrates with SQS	● Unlimited consumers poll data ● Replay data: must provision throughout ● Ordering at a shard level ● Able to consume records hours later

Security

IAM
- AWS services need to be assigned IAM roles to perform actions on our behalf from our account
  - Federate workforce into AWS accounts if multiple directories or want to manage permissions of user attributes
- IAM Roles: Global service that gives temporary permissions for applications and users to call AWS resources
  - Gives permission to access cross account resources
  - Assign IAM roles to instances: attach to instances in stopped/running state or replace for instances in running state
- IAM Groups: collection of IAM users that lets you specify permissions for those users
  - Attach policies to a group to give those users in the group the correct permissions
- IAM Conditions: make IAM policies more restrictive based on a condition
- IAM Permission Boundaries: delegate responsibilities to non admins within boundaries or to allow self management
  - Use policies to set permissions (in JSON) that is only for users and roles (NOT groups)
- IAM DB Authentication: ensures that the database can only be accessed with profile credentials specific instances via a token
  - Authenticate to DB instance (MySQL, PSQL) with an authentication token: no password until connecting to a DB instance
    - RDS generates tokens on request with a 15 minute lifetime (no need to store user credentials in the DB)
  - Network traffic to and from the DB is encrypted using SSL: use IAM to centrally manage access to your DB resources and use profile credentials specific to instances to access your database
- Power User: Access to all AWS services (except management of groups and users within IAM)
- Full Admin Access: perform almost all AWS tasks except closing the company account and configuring S3 bucket to enable MFA delete (only the root account user can do this)
- Trust Policies: the only resource based policy that IAM supports (principal entities can assume roles)
- Service Control Policies (SCPs): used to manage your organization and offers central control over the maximum available permissions for all accounts to ensure your accounts stay within an organization's access control guidelines
  - Restrict AWS services, resources, individual API actions the users and roles in each member account can access

    - Does not affect any service linked rules but does affect all users and roles in attached accounts (including roots)
    - If a user/role has an IAM permission policy that grants access to an action that is either not allowed or explicitly denied by the applicable SCPs, the user/role cannot perform that action
- To allow users to make API calls to AWS resources: create a set of access keys for users and attach permissions:
    - For CLI and SDK: new users are given access key ID and password to grant access
    - For console: new users need a customized sign in link to sign in with their credentials
    - By default, new users have no access to any AWS services (it must be granted to them)
    - Once logged in, all users should set up MFA

AWS SSO
- Centrally managed permission management service: allows access to multiple AWS accounts and third party business applications, provides users with single sign-on access to all their assigned accounts and applications from one place
- Integrated with AWS Organizations to sign into SSO to get access to login to all accounts with that Org
    - Federates workforce into AWS accounts based on group membership in a single directory
    - Does not issue short lived credentials itself

STS (Security Token Service)
- Grants short lived, limited, temporary access to AWS resources by issuing a temporary token that is valid for up to 1 hour
- Web Identity Federation: federated users outside of AWS assume temporary roles and credentials for accessing AWS resources by assuming a web based identity provided
    - No need to create custom sign in code or manage user identities
    - Users sign in using a well known identity provider (IdP) or openOD connect OIDC compatibility IdP, and receive an authentication token that can be exchanged for temporary security credentials
- SAML 2.0 integrates with other directories for access to AWS without creating IAM user (must set up a trust)
    - Use assumeRole policy: SAML allows you to provide federated single sign on access to the AWS console
    - If on premise is not compatible with SAML, must write custom identity broker application
    - If a company has resources hosted on premise and on cloud but wants to access resources on both environments using on premise credentials (stored in AWS AD), then set up a SAML 2.0 based federation by using Microsoft AD federation service

AWS Cognito
- Provides web identity federation to give an authorization code from IdP to gain temporary AWS credentials
    - Manages users for applications: adds sign in, sign up and access control to applications (no need to create IAM users)
- Cognito User Pools: create users and handle user directories used for authentication and sign up/in functionality
    - Used to create serverless databases of uses for mobile applications (login and verification)
    - Gives a web UI to sign in users and allows third party sign in / user management
    - After creating user pools in API gateway, must create a cognito_user_pools authentication that uses the pool
- Cognito Identity Pools: allows users temporary access through AWS credentials to direct AWS services and resources
    - Has an IAM role so all logged in users share this role to access AWS services
    - Federated Identity Pools: provides direct access to AWS from the client side and log in from the federated IdP
        - Get temporary login credentials with predefined IAM policy
- Cognito Sync: synchronize data from device to cognito

SSM Parameter Store
- Securely store your configuration and secrets with optional seamless KMS encryption (using hierarchy pattern)
- Store plan text or encrypted data, reference parameters in scripts (must rotate secrets yourself)

AWS Secrets Manager
- Store secrets with capability to force rotation every x days and automate the generation of secrets in rotation
- Users / applications retrieves secrets with a cal to Secrets Manager API so no need to hardcode sensitive information
- To safely import SSL/TLS certificates for applications: use AWS Certificate Manager or IAM Certificate Store (if certificate is from a 3rd party CA)

Encryption
- Encryption in Flight (SSL/TLS): protects traveling data before sending and decrypt data after receiving
    - Only sender/receiver know how to encrypt/decrypt via SSL certificates (HTTPS guarantees security)
    - SSL certificates are only useful to encrypt data in transit

- Server Side Encryption (at rest): data is encrypted after being received by the server and stored in encrypted form via keys
  - Client Side Encryption: data is encrypted by client and servers will never be able to decrypt that data
  - If the data in the cloud needs to be encrypted at rest at all times using keys store on premise, use either::
    - Client side encryption to provide at rest encryption
    - Lambda function triggered by S3 events to encrypt data using the customers key

KMS (Key Management Service)
- AWS provides software for encryption and manages all encryption keys
- Customer Master Keys (CMKs) can never be retrieved by the user but can audit trail usage of keys:
  - Symmetric Keys: single key for encryption and decryption (envelope encryption)
  - Asymmetric Keys (RSA and ECC key pairs): public key for encryption, private key for decryption
  - Deleting CMKs is destructive: KMS enforces a waiting period from 7-30 days (default) where CMK state is pending
    - Can cancel key deletion before waiting period ends
- Enable automatic key rotation every year (new key has same CMK ID), the previous key is kept active to decrypt old data
- Prevent other developers from seeing credentials by creating a new KMS key and using it to enable encryption helpers that leverage KMS to store and encrypt the sensitive information

CloudHSM
- AWS provisions encryption hardware to manage keys and the client handles the software
  - AWS has no access to the keys, they just make sure that the HSM device is tamper resistant
- Logging in as admin more than 2 times with the wrong credentials with zeroize the HSM and all keys, certificates, data will be destroyed (AWS cannot recover any of it)
  - Use HSM in at least 2 AZs via clusters to prevent unauthorized users
- For all data in the cloud to be encrypted at rest with full control over encryption of created keys and the ability to immediately remove key materials from KMS and audit key usage independently of CloudTrail:
  - Use AWS KMS to create a CMK in a custom key store and store non extractable key material in CloudHSM

AWS Shield
- Standard: Free, managed DDoS protection service that safeguards applications running on AWS (activated by default)
- Advanced: Mitigation service that protects against more sophisticated DDoS attacks (has 24/7 access to a response team)

AWS WAF (Web Application Firewall)
- Protects web applications from common web exploits at layer 7 HTTP (where it has more information on how requests are structured) and controls how traffic reaches your applications
  - Can be deployed on ALB, API Gateway, Cloudfront to apply rules to the CF distribution
  - To use: define a web ACL and SGs against blocking traffic/common attack patterns, IP addresses, HTTP headers
- To allow or block web requests based on IP addresses that requests originates from: create at least 1 IP match condition which lists up to 10,000 IP address ranges that your requests originate from
- To block specific countries, create a geo match statement listing the countries you want to block
- A website receives a large amount of illegitimate external requests from multiple systems that constantly change:
  - Want minimal impact on legit traffic
  - Create a rate based rule that tracks the rate of requests for each originating IP address and triggers for IPs over a limit, then associate web ACL on ALB

GuardDuty
- Intelligent threat discovery service to protect account by analyzing logs from CloudTrail, VPC FlowLogs and DNS Logs
  - Uses ML algorithms and anomaly detection techniques
  - Disable service in general settings to stop using and delete existing findings

Inspector
- Automated security assessments for EC2 instances: must install an Inspector agent in the OS of the EC2 instance for the service to run (returns a report with the list of vulnerabilities found)

Macie
- Fully managed data security and privacy services that uses ML pattern matching to discover/protect PII sensitive data

Serverless Services

- Managed, deployment only: lambda, DynamoDB, Cognito, API gateway, S3, SQS, SNS, Kinesis, Aurora, Fargate
- Compute services that could be used for multi thread processing: EC2, ECS, Lambda
- Already built in a fault tolerant fashion so no need to provision these services across multiple AZs

Lambda

- Virtual functions that run on demand and are continuously scaling out
  - Supports Go, C#/Net, Java, Node.js, Python, Ruby and standard rate cron jobs for frequencies of up to one/minute
  - Deploy a CW alarm that notifies you when the function metrics exceeds the expected threshold
  - By default, always operate from an AWS owned VPC so they have access to any public internet address or public AWS API (once function is VPC enabled, it needs a route through a NAT gateway to access public resources)
  - Use lambda layer to place code you want to reside in more than one lambda function
  - Limits: memory allocation (128MB to 10GB), default timeout is 3 seconds, maximum duration per request is 15 minutes, 4 KB of environment variables, only 1000 concurrent executions, cannot directly handle API requests
  - Each time your function is triggered, a new/separate instance of that function is started
- Billing is based on: the number of requests for each time it executes in response to an event notification or invoke call, compute duration (in 100ms units), MB of RAM reserves
- A dynamic PB_URL variable loaded lambda environment variables to allow for lambda to have dynamic variables from within
- For lambda to have access to a DB password, run the password as an encrypted env variable and decrypt it at run time
- Use AWS X-Ray to understand how your lambda is working: helps analyze and debug your applications

Lambda@Edge

- Run lambda at edge locations for faster service: CloudFront runs code closer to users to improve performance and reduce latency: only pay for compute time, can run code in response to events generated by CloudFront
  - No need to provision or manage infrastructure in multiple locations

API Gateway

- Fully managed service that makes it easy for developers to create, publish, maintain and monitor secure APIs at scale
  - APIs act as the front door for applications to access the data, business logic or functionality from backend services
  - Only pay for API calls received and the amount of data that is transferred out
  - Handles API versioning and can use with lambda for serverless API (invokes abda to expose REST API)
- Create RESTful APIs and WebSocket APIs that enable real time two way communication applications
  - Creates APIs that enable stateless client server communication and WebSocket APIs that adhere to web socket protocols (enables stateful duplex communication between client and server)
- Enable API caching to cache your endpoint responses: reduce the number of calls made to your endpoint and improve the latency of requests to your API
- Throttling: process of limiting the number of requests an authorized program can submit to a given operation in a certain amount of time (API Gateway, SQS, Lambda can handle sudden traffic spikes)
  - Uses token bucket algorithm: a token counts for a request by setting a limit on the steady state rate and bursts of request submissions against all APIs in your account
  - Use throttling limits (protects from traffic spikes) in API gateway to track the number of requests per second
- To prevent API Gateway from being overwhelmed by too many requests and improve overall performance across the APIs in your account, enable caching and set throttling limit
  - If a cache is configured, then API gateway will return a cached response for duplicate requests for a customizable time but only if it is under the configured throttling limits

Containerization- Docker

- It is a standardized platform for deploying packaged applications into containers that can run on OS
  - Containers allow co locating applications on instances to efficiently use compute instances without a lot of overhead
- Create a Docker file that builds into an imager and run it to get a container
- Container management platforms: ECS, Fargate, EKS

ECS	EKS
• Container registry: a private repository to store images • Used to store, manage and deploy containers on AWS • Pay for what you use • Associated and fully integrated with ECS	• Kubernetes service: launch/manage cluster on AWS • Open source ECS alternative • Launch modes: worker nodes or target to deploy serverless containers

ECS (Elastic Container Service)

- Fully managed container orchestration service to deploy, manage, and scale containerized applications
  - Must provision and maintain infrastructure (can run compute intensive custom script here)
  - Supports batch jobs: short lived, often parallel and can be packaged into a Docker image via RunTask action to run one or more tasks (starts ECS task on an instance that meets the task requirements)
- Place ECS agent (needs IAM role) on container instances to register them with the cluster and launch tasks
  - Inject sensitive data into containers by storing them in AWS Secrets Manager or Parameter Store, then referencing them in the container definition
- ECS with EC2 Launch Type: charged based on EC2 instances and EBS volumes are used
- ECS with Fargate Launch Type: charged based on vCPU/memory resources that containerized app requests (serverless)

Fargate

- Serverless compute engine for containers that works with both ECS and EKS
  - Runs containers based on the CPU/RAM needed
- Launches tasks by itself: uses ENI (distinct IP) in VPC to bind tasks to network IP (where 1 task = 1 IP)
  - Must have enough IPs in VPC for each task

Networking

Private IP Addresses

- Machine can only be identified through private network (not accessible over the Internet): unique IP across network
- Allows certain values:
  - 10.0.0.0 to 10.255.255.255 (10.0.0.0/8) for big networks
  - 172.16.0.0 to 172.31.255.255 (172.16.0.0/21) for default AWS network
  - 192.168.0.0 to 192.168.0.0/16 (192.168.0.0/16) for home networks
- Use to reduce the cost of data transfer between instances in 2 AZs of a single region without sacrificing reliability or scalability

Public IP Addresses

- Machine can be identified through public Internet network (accessible over the internet): unique IP across the entire web
  - Easily find geolocation: released when the instance is stopped

Elastic IP Address

- A static IP address that you can attach to an instance (one at a time) and once attached, it stays with the instance even if you stop/start it up again: masks failure of an instance by remapping address to another instance
  - Does not incur charges as long as: EIP address is associate with an instance, instance associated with the EIP is running and the instance only has 1 EIP attached to it
- Acts as a fixed public IPv2 address until an explicit removal of EIP and termination of the instances
- If your instance does not have a public IPv4 address, associate an EIP with your instance to enable communication with the Internet (not necessary for communication between instances in a VPC)

CIDR

- Defines IP address range where components includes base IP (xx.xx.xx.xx) and subnet mask (/26)
  - Base IP represents an IP contained in the range and subnet masks defines how many bits an IP can change

SSH

- Connects to a server to perform actions and control a remote machine over port 22
- To establish an SSH connection from a home computer to an EC2 instance:
  - On SG: Add an inbound rule to allow SSH traffic to EC2 instance
  - On NACL: Add both inbound and outbound rules to allow SSH traffic to the EC2 instance

Ports

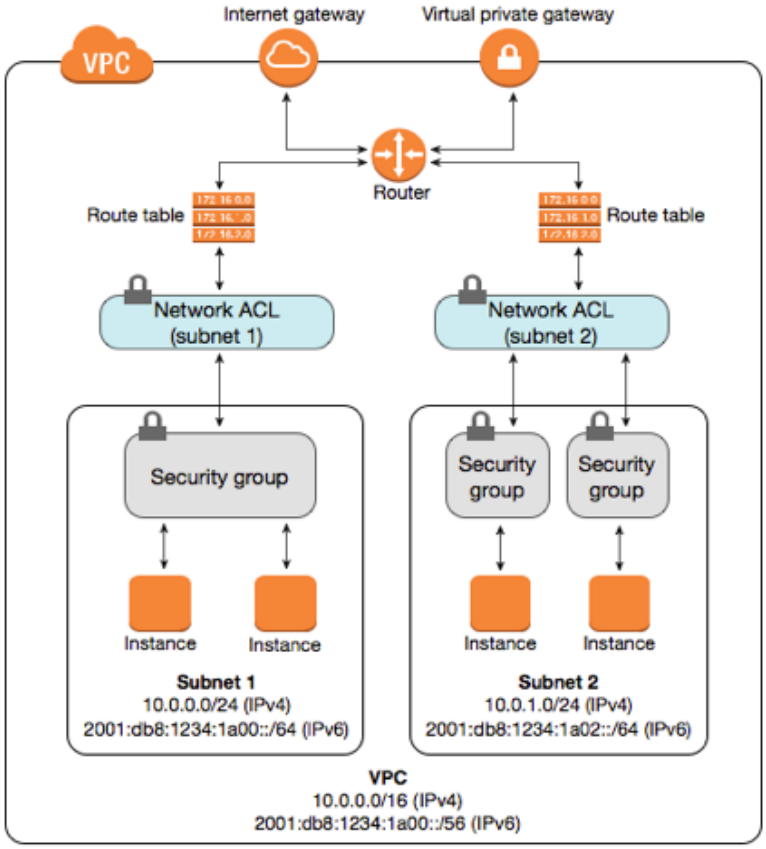
- FTP (port 21) via TCP: upload files to a file share
- SFTP (port 22): securely upload files to a file share
- HTTP (port 80): allows access to unsecured websites on internet
- HTTPS (port 443): allows access to secure websites
- RDP (port 3389): log in to a windows instance

Security Group

- A firewall that controls access to and from the instance: only applies if SG was specified when launching the instance
- Operates at the instance level: locked to a region or EC2/VPC combination (can be attached to multiple instances)
- Allow rules only: evaluates all rules before deciding whether to allow (Updates are supplied immediately)
- Stateful: returning traffic is always allowed out (if inbound request passes, the outbound request will always pass)
- Cannot be used to block IP addresses and timeout errors indicate SG issues
- Database SGs are used with DB instances that are not in a VPC and on EC2-Classic Platform:
  - No need to specify a destination port number when you create DB SG rules
  - Port number defined for the DB instance is used as the destination port number for all rules designed for DB SG
- Revoke-security-group-ingress command moves 1 or more ingress rules from a security group: eacj rule consists of the protocol and CIDR range of source SGs (rule changes are sent to instances within SG with a possible small delay)
- If the SG of an EC2 instance is allowing port 80 for HTTP but instances are still showing unavailable from the load balancer, then the health configuration of the SG was not properly defined
- Change the SGs that are associated with an instance launched in a VPC when the instance is in the running or stopped state

NACL (Network Access Control List)

- A firewall that controls access to and from the subnet: always evaluated before SGs
- Operates at the subnet level: automatically applies to all instances in the subnet the NACL is associated with
- Allow and deny rules: processes rules in an numbered order when deciding whether to allow
  - Rules are numbered and evaluated starting with the lowest numbered rule until an explicit deny: as soon as the rule matches the traffic, it is immediately applied (rules take effect immediately)
- Stateless: returning traffic must be explicitly allowed (response must be evaluated, no matter the inbound rule)
- Can be used to block IP addresses: to deny access, modify NACL associated with all public subnets in the VPC
- Default NACL allows all inbound/outbound rules without restrictions: subnet added to VPC is associated with default NACL
- To enable the connection to a service running on an instance: the associated NACL must allow inbound traffic on the port that the service is listening on AND allow outbound traffic from the ephemeral port range (1024 - 65535)
  - Ephemeral port becomes the destination port for return traffic from the service so outbound traffic from the port must be allowed in NACL
- Creation of custom NACL: the default configuration denies everything until rules are added
  - Add rules for HTTP, HTTPS, SSH to inbound/outbound files to allow access to and from the Internet
  - In inbound rules, must add ephemeral ports (102 - 65535)



VPCs

VPC

- A logically isolated datacenter in the cloud where you have complete control and has its own data center
  - All new accounts have 1 VPC per region: VPCs are tied to a specific region with defined CIDR
  - Public IP, private IP and DNS names allow all subnets to have a route out to the internet
  - Default VPCs have public subnets and an internet gateway for access to the internet (Custom VPCs do not)
  - Can assign multiple IP addresses and network interfaces to your instances
  - Consists of virtual private gateways, route tables, NACLs, subnets and SGs
  - When you create a VPC, a default route table, NACL and SG are created (won't create subnets or IGs)
  - Can have up to 5 Amazon VPCs per AWS accounts per AWS region but can request for an increase
  - AWS releases your instances public IP address when it is stopped, hibernated or terminated
    - Stopped or hibernated instance receives a new public /IP address when it is started
- Subnets: Created within a VPC (1 subnet = 1 AZ) - AZs are randomized between AWS accounts
  - AWS always reserves 5 IP addresses within your subnets
    - AWS reserves both the first 4 and the last IP address in each subnets CIDR block
    - Possible for an ASG to provision too many instances in a subnet and exhaust the number of internal IP addresses available in the subnet
  - Non Default subnets and their instances are not public (by default, assigned a private IP address)
    - To make internet accessible, assign a public IP address and add an IG to the subnet
  - To check that instances in different subnets can communicate inside the VPC:
    - Check that NACL allows connection between the subnets
    - Check that all SGs allow the application port to communicate to the database on the right port or protocol
  - An AZ is represented by a region code and letter ID (us-east-1a).
    - To ensure that resources are distributed across AZs for a region, AWS maps AZs to each account
    - To coordinate AZs across accounts, use AZ ID: unique ID that enables you to determine the location of resources in 1 account relative to the resources in another account
  - Each subnet must be associated with NACL: if you don't explicitly associate a subnet with NACL, then the subnet is automatically associated with the default NACL
  - Launch an instance in default VPC and don't specify a subnet, it is automatically launched into a default subnet in your default VPC: public and private IP addresses are available for instances if you don't specify the CIDRs
  - Instances in new subnets in a custom VPC can communicate with each other across AZs
  - To ensure that two instances in different subnets can communicate properly:
    - All SGs allow communication between the app and database on the correct port using the proper protocol
    - NACL allows communication between the two subnets
- Route Tables: Controls routing and network traffic for the subnet
  - Each subnet must be associated with a route table (implicitly associated with the main route table)
  - A subnet can only be associated with 1 route table at a time but can associate multiple subnets with a route table
  - A public subnet route table must have an entry for the internet gateway
  - VPC has a single subnet, an EC2 instance within that subnet with a IG but the instance can't be reached from the Internet: route table must not have an entry for the IG (add route with destination of 0.0.0/0 for IPv4 traffic)
  - Add a route to routing table that allows connections to the internet from your subnet:
    - Destination: 0.0.0.0/0 to target: your Internet gateway
- VPC Flow Logs: captures IP information for all traffic flowing in/out of the VPC/subnet/instance and sends to S3 or CW
  - Created at VPC, subnet and individual network interface levels
  - Cannot enable flow logs that are peered with your VPC unless the peer VPC is in your account
  - cannot have flow logs across multiple AWS accounts or change its configuration once created
  - Traffic generated by instances when they contact DNS server is not logged:
    - Using your own DNS server, all traffic to that server will be logged
    - Traffic generated by a windows instances for Amazon windows license activation is not monitored
    - Traffic to and from 169.254.269.254 for instance metadata and user data is not monitored
      - Neither is DHCP traffic or traffic to reserved IP addresses for default VPC router
  - Once created, can retrieve and view data in chosen destination
- Tenancy Attribute: each instance launched inside a VPC has a tenancy attribute
  - Default: instances run on shared hardware
  - Dedicated: instance runs on a single tenant hardware (like Dedicated instances)
  - Host: instance runs on an isolated server with configurations you control (like Dedicated hosts)
  - Able to change tenancy of instances between dedicated and host attributes

    - Can change the instance tenancy attribute of a VPC from dedicated to default: modifying the tenancy of the VPC does not affect the tenancy of any existing instances in the VPC (use CLI, SDK, EC2)
        - Next time you launch an instance in the VPC, it has a tenancy of default unless you specify otherwise
- Private Hosted Zone: created and associated with a VPS for DNS queries (resolved by AWS DNS server only)
    - If DNS queries remain unresolved, enable DNS hostnames and DNS resolution (required settings)
- If VPCs are provisioned into multiple AWS accounts and you want to reduce admin overhead while providing shared access to resources: build a shared services VPC which provides access to services required by workloads in each of the VPCs via directory services or VPC endpoints

AWS Private Link: Connects AWS services with other AWS services through a private network
- Privately connects between different VPCs, AWS services, on-premises apps securely over the AWS network
    - Secures traffic from VPC environments already in AWS
- For peering 10s, 100s, 1000s of customer VPCs: doesn't require VPC peering or route tables

VPC Peering: Connects VPCs through a private network
- Allows private connection between 2 VPCs directly via the AWS network so they behave as if they were the same VPC
    - Must have non-overlapping CIDR addresses: route traffic between VPCs within the same account, different accounts or cross region to facilitate data transfer and file sharing
    - Non-Transitive: every VPC subnet you want to peer to must update its route table
- Supports IPv6 communication: VPCs must be in same region and have routing set up to use the addresses
- Able to VPC Peer between regions but cannot do VPC Peering
- Only routes traffic between source and destination VPCs: does not support edge to edge routing

VPC Endpoints: Connects VPC with AWS services through a private network
- Allows private connection between VPC and supported AWS services or VPC endpoint services via AWS network
    - No setup required, via AWS Private Link: all traffic from VPC stays in the network
    - Doesn't require IG, NAT Gateway, VPN connection or DX connection
    - Instances in VPC don't need public IPs to communicate with resources: traffic doesn't leave AWS network
- Endpoints: Virtual devices that allow communication between instances in VPC and AWS services without imposing availability risks or bandwidth constraints on network traffic
    - When creating VPC endpoints: must create and attach an endpoint policy that controls access to the services you connect to (doesn't override or replace IAM / bucket policies)
- Interface VPC Endpoint: MOST services use this (provisions ENI as an entry point)
- Gateway VPC Endpoint: Used by S3 and DynamoDB (provisions a target that must be used in the route table)
    - Need to launch instances in a private subnet for a web portal and these instances must send data to DynamoDB and S3 via private endpoints that don't pass through the internet: route all access to S3 and DynamoDB
    - Ensure traffic between your VPC and the other service does not leave the Amazon network

ENI (Elastic Network Interface)
- A logical component in a VPC that represents a virtual network card that provides network connectivity and a private IP address to an instance (By default, each instance has an ENI)
    - Create ENIs independently from instances and can move them between instances fast
    - Each ENI lives in a subnet (ENIs are associated with subnets, not instances)
- If an application (using a private IPv4 address) fails or becomes unreadable and want traffic to be directed to a standby instance, then attach a secondary ENI to instance configured with a private IP address (for quick network failover) and move ENI to standby instance if the primary instance fails

Internet Gateway: connects VPC and public subnets with the Internet
- Allows instances in a public subnet (with public IP addresses) to connect with the Internet
    - Must be created separately from the VPC: 1 IG can be attached to 1 VPC
    - Provide target in VPC route tables for Internet routable traffic
    - Performs network address translation for instances with public IPv4 addresses
- Instances in a private subnet should not access the Internet through IG , or they would be accessible through the Internet
- For IG to access Internet:
    - NACL associated with the subnet must have rules to allow inbound/outbound traffic
        - On port 80 for HTTP and on port 443 for HTTPS
    - Route table in subnet must have a route to IG

- To access instance from the Internet:
  - Must have IG attached to a public VPC and route the route entry to the IG in the route table of the VPC
  - SSH the public IP address attached to the instance

NAT Gateway: connects private subnets to the Internet
- Allows instances in private subnets with IPv4 addresses to access the Internet without being accessible themselves
  - Supports 5 Gbps of bandwidth (autoscales up to 45 Gbps) and can associate exactly 1 EIP to it
  - Supports TCP, UDP, ICMP and up to 55,000 simultaneous connections to each unique destinations
  - Cannot associate SG with it: only NACL to control traffic to/from the subnet where the NAT is located
  - Must be created within a public subnet (in a specific AZ) but can only be used by instances from other subnets
  - Update route table so NAT has a route out to the Internet
- To configure NAT gateway in a public subnet:
  - Specify EIP to associate NAT gateway to (cannot be changed once associated)
  - Define custom route table with route to NAT gateway for Internet
  - Associate with private subnet by updating route table to point internet bound traffic through them
- A single NAT gateway is not HA: need one in each AZ and configure routing such that resources use NAT in the same AZ

NAT Instances
- Older version of NAT gateway: supports port forwarding, can be used by Bastion Host (can be associated with SG)
  - Must have EIP attached: route table goes to a fixed IP (routes traffic from private subnets to NAT instances)
- When creating a NAT instance: always disable source and destination check on the instance
  - The amount of traffic it is able to support depends on the instance size: MUST be in a public subnet with a route out to the private subnet to the NAT instance for it to work (always behind an SG)
- Network bandwidth depends on the bandwidth of the instance type (increase class size to solve performance issues)
- Ensure “Source/Destination Checks” is disabled to allow the sending and receiving of traffic for the private instances

Egress (Outbound) - Only IG
- NAT Gateway for IPv6 based traffic only: all IPv6 addresses are public (no private range)
- Allow IPv6 traffic within a VPC to access the Internet (denies any Internet resources from connecting back to the VPC)

Bastion Host
- Allows public instances in public subnets to SSH or RDP to instances within private subnets: must only have port 22 traffic
- It is a computer ron a network designed to withstand attacks (blocks SSH brute force attacks)
  - Should always be placed on a public subnet: this public subnet is then connected to a private subnet with either a public IP or EIP address with the RDP/SSH access defined in the SG
- Implement a Bastion Host:
  - create a small instance that will be the host that acts as a jump server to connect to other instances in the VPC
  - Instance should have a SG from a particular IP address for maximum security

Site to Site VPN (IPsec VPN Connection): Connects on-premise to VPC over the Internet
- Allows the creation of an encrypted, private connection over the Internet between VPC and on premise data center
  - Link customer and VPN gateway using site to site VPN connections
  - Use IPsec to establish encrypted connectivity to secure IP communication
  - Quick and easy to set up: configure connections within minutes with modest bandwidth requirements
- Customer Gateway: a software app or physical device on the on-premise data center
  - A resource that is installed on the customers side and provides a customer gateway inside a VPC
  - IP address is either the static Internet routable address for the customer gateway or if it’s behind a NAT, it uses the public IP address of the NAT gateway on the network
- Virtual Private Gateway: VPN concentrator set up on AWS (created an attached to VPC)
  - Sits at the edge of VPC and is a key component when using a VPN since it is responsible for site to site connection from on-premise to a VPC

Direct Connect (DX): Connects on-premise to VPC over a private and secure network
- Provides a secure and dedicated private connection from on-premise data center to VPC (bypasses the Internet)
  - Must set up a connection between DX and AWS DX locations (does not encrypted traffic in transit)
    - Established DX routers in large facilities across the world (provides access to all AWS regions)

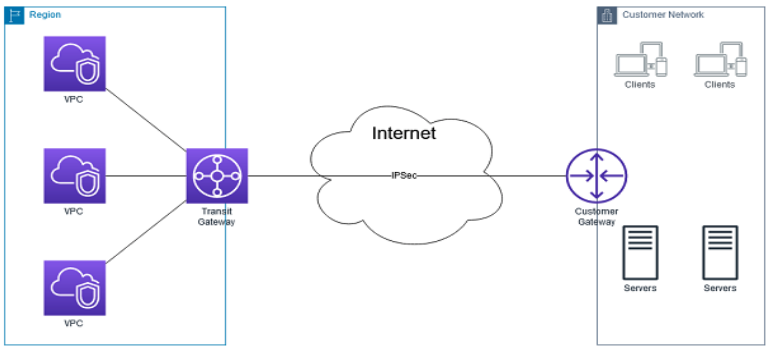
    - Occurs over a standard 1 GB or 10 GB Ethernet fiber-optic cable with one end of the cable connected to your router and the other to an AWS DX router
    - For workloads with an increased throughput and business that want ultra low latency connectivity with AWS
- Set up DX Connection:
    - Create a virtual interface in DX console and a customer gateway for on-premise data center
    - Create a virtual private gateway and attach it to VPC environment: select VPN connections and create new VPN connection (select both customer gateway and virtual private gateway)
    - Once VPN connection is available, set up the VPN
- Direct Connect Gateway: To set up DX on one or more VPCs in many different regions on the same account, must set up a dedicated connection with a physical Ethernet port and a hosted connection
    - With DX and VPN: combine at least 1 DX connection with VPN to provide a consistent experience
- On-premise data center is connected to the AWS cloud via DX:
    - Resolve any DNS queries for resources in VPC from on-premise: create inbound endpoint on Route53 Resolver and DNS resolvers on the on-premise network to forward DNS queries to Route53 Resolver via the endpoint
    - Resolve DNS queries for resources in on-premise from VPC: create outbound endpoint on Route53 Resolver so it can conditionally forward queries to resolvers on the on-premise network via this endpoint
    - To conditionally forward queries: create resolver rules that specify the domain names for DNS queries that you want to forward and IP addresses of the DNS resolvers on the on-premise network you want to forward the queries to

VPN CloudHub: Use Internet to link on-premise sites

- Provides secure communication between on-premise sites if there are multiple VPN connections: primary or secondary backup connections between all locations (low cost hub and spoke)
- Simplify VPC connections: have multiple sites, each with its own VPC connection and use AWS VPN CloudHub to connect those sites together: low cost, easy to manage, operates over public Internet (all traffic is encrypted)
- Sites that use DX connections to a virtual private gateway can also be part of VPN CloudHub
    - Suitable for multiple branch offices with existing internal connections

Transit Gateway

- Transitive peering connection between thousands of VPCs and on-premise sites using a single gateway
    - Hub and spoke star connection that controls how traffic is routed
    - It is a regional resource that increases the bandwidth of site to site VPN connections
    - Share direction connection between multiple accounts using RAM
    - Only need to create/manage a single connection from a central gateway to simplify and reduce operational costs
    - Route table limits which VPCs can talk to which other VPcs: the only service that supports IP multicast
    - Supports IP multicast (not supported anywhere else)
- Attach all hybrid connectivity to a single transit gateway that consolidates and controls the entire AWS routing configuration for your organization in one place



High Availability

- To reduce risk of bad deployments and single point of failure for AWS resources:
    - Use several target groups or auto scaling groups under each load balancer
    - Use multiple auto scaling groups and boundaries for a staged or canary deployment process
        - Canary deployment: strategy that releases an application or service incrementally to a subset of users
        - Traffic is shifted in 2 increments: choose options that specify the percentage of traffic shifted to updated Lambda version in the 1st increment and the interval before the remaining traffic is shifted in the 2nd

    - Use Route53 with health checks to distribute load across ELBs
    - Use CLB to spread load over several AZs
- Want to develop a highly available application with a stateless web tier, the most cost effective means is:
    - Use an elastic load balancer, a multi AZ deployment of an ASG of EC2 Spot Instances (primary) running in tandem with an ASG of EC2 On-Demand instances (secondary) and DynamoDB
        - On-Demand instances only spin up if the Spot instances are not available
        - DynamoDB supports stateless web/app installations better than RDS

ELB (Elastic Load Balancer)

- Managed load balancer that balances incoming internet traffic to multiple servers and instances to spread load
    - Works only out of one region: must provision to each region you operate out of
    - Uses regular health checks to know if instances are available on specified port/route
        - Does not make instances available again, it only stops sending requests to instances that failed the check
    - Separated public facing traffic from private traffic by providing the static DNS name to use in the application
    - Lifecycle of a request: browser requests an IP address for LB from DNS, DNS provides the IP, browser makes HTTP request for HTML page from LB with IP, AWS perimeter devices checks and verifies requests before passing it to the LB, the LB finds active web server to pass the HTTP request to, the web server returns the requested file and the browser receives the file (if application stops working, LB gets 504 error)
        - LBs do not have predefined IPv4 addresses, resolve to them using DNS name
    - Need a minimum of 2 public subnets to deploy an Internet facing load balancer
- Types of load balancers:
    - Classic LB: supports HTTP, HTTPS, TCP with fixed hostname
    - Application LB: supports HTTP, HTTPS, Web Socket for routing based in URL paths, hostnames, query strings
        - Good for microservices, container based apps (layer 7 apps): allows routing based on request content
        - Authenticate users to access your apps to offload the work to the LB so other apps can free up
        - Offers SSL termination and makes SSL offload process simple through tight integration with SSL
        - Must be deployed into at least 2 AZ subnets
        - Has functionality to distinguish traffic for different targets (websites) and distribute traffic based on rules for: target groups, conditions, priority
        - Supported path-based and host-based routing
    - Network LB: supports TCP, UDP, TLS for extreme performance at layer 4 apps
        - Good for millions of connections
        - Exposes public static IP addresses of users to the public web (reachable from IPs): bring your own IP addresses to use the trusted IPs as EIP to NLB without needing to re-establish any new IPs
        - Receives connection request, selects a target from target group, attempts to open TCP connection to selected target on a specified port
- Stickiness: Ensures that the same client is always redirected to the same underlying instances behind ALB or CLB
    - Website doesn't need to re authenticate users when they switch pages (May cause uneven distribution)
    - LB is always changing IP address: use cookies to save user session data (heavy HTTP requests are slow) or use session ID with Elasticache for millisecond performance
- Cross Zone Load Balancing: Each LB instance distributes incoming requests evenly across all registered instances across multiple AZs (enabled for ALB, disabled for NLB)
    - Without it: LBs cannot see the distribution between AZs and it sends load evenly to all AZs, regardless of how many instances are actually sitting within each AZ
    - With it: There is weighted routing between AZs so each instance will receive the same percentage of traffic
- Security: SSL certificates allows traffic between clients and LBs to be encrypted while in transit (only supported by ALB)
    - Use SNI to load multiple SSL certificates onto the servers and expose HTTPS traffic for each multiple host name
        - Able to load multiple TLS (newer SSL) secure applications (each with own certificate) behind a single LB
        - SNI makes it easy to support more than one certificate within a single IP address
    - To use: bind multiple certificates to the same listener on the LB, it will automatically choose the best certificate
    - SNI custom SSL relies on the SNI extension of TLS protocol which allows multiple domains to serve SSL traffic over the same IP address by including the hostname of which viewers are trying to connect
- Connection Draining: Instance shuts down what it was doing before de-registration and LB stops sending new requests
    - If the instance fails health checks, LB does not send any new requests but it allows existing in flight requests to complete for the duration of the timeout (when timeout is maxed, LB closes the connection to the instance)
- Perfect Forward Secrecy: Provides additional safeguards against eavesdropping of encrypted data in transit through the use of a uniquely random session key

- Path Patterns: Create a listener with rules to forward requests based on the URL path set within those user requests
- Target Groups: Each TG is used to route requests to one or more registered targets
  - When creating ech listener rule, specify a target group and conditions. When a rule condition is met, traffic is forwarded to the corresponding target group. Create different target groups for different types of requests.
  - Creating a target group: Specify a target type (can be EC2 instance, Lambda function, IP address)
    - IP Address: can route traffic using any private IP from at least 1 network interface
  - HTTP 503 Error: no registered targets for target groups
  - Specify target with instance ID: traffic is routed to instances using the primary private IP in the primary network interface of the instance so the LB rewrites the destination IP in the packet before forwarding it to target instance
  - Specify target using IP address: route traffic to an instance using any private IP from one or more network interfaces to enable multiple applications on an instance to use the same port (Each NI can have its own SG)
  - If LB marked all instances in a target group as unhealthy but they are still internet accessible, could be:
    - Instance SG does not allow for traffic from SG of the LB or the route for the health check is misconfigured
- Features:
  - Look at X-Forwarded-For header in backend to find the true IP address of connected clients
  - Open EC2 security on port 80 of the LBs SG to ensure only LB has access to instances
  - Enable feature to provide access logs with detailed information about requests sent to the LB
  - Enable IPv6 DNS resolution by creating a 2nd DNS resource so ALIAS AAAA record resolves to the LB with IPv4

Auto Scaling Groups
- Scales out (adds instances) to match increased load or scales in (removes instances) to match decreased load
  - Grows instances with defined minimum and maximum, desired capacity and scaling capacity
  - Creates replacements for terminated instances (free service: only pay for the underlying instances)
  - To add existing instances to ASG: ensure instance is launched in an AZ defined in the ASG and ensure the AMI used to launch the instance still exists
  - Scaling In:
    - To control whether an ASG can terminate an instance when scaling in, use instance scale in protection
    - When ASG initiates a scale in event, increase the deregistration delivery by at least 10 minutes so users do not see their current requests interrupted (helps inflight requests complete)
    - LB will stop sending requests to targets that are deregistering and waits 300 seconds (default) before completing the deregistration process
  - Each ASG always uses a launch template or launch configuration as a template for its instances
- Scaling Policies:
  - Target Tracking Scaling: Scale current capacity based on a target value for a specific metric
    - Example: Keep CPU usage below a certain threshold (Set up CloudWatch alarms that monitors metrics)
  - Simple Scaling: Scale current capacity based on a single scaling adjustment (cannot react to sudden spikes)
  - Step Scaling: Scale current capacity based on a set of scaling adjustments that vary (via CloudWatch alarms)
    - Use if an application receives bursts of traffic in the morning (causing request timeouts) and the instance takes some time to boot up before it can response to requests (configure instance warm up time condition)
  - Scheduled Actions: Scale based on a schedule for predictable load changes
    - Must know application patterns to anticipate scaling (not considered dynamic)
  - Scaling Cooldown: Period that ensures that ASG doesn't launch of terminate additional instances before the previous scaling activity takes effect (default value of 300 seconds)
- Default Termination Policy: Find the AZ with the most number of instances
  - If there are multiple instances in the AZ, priority is given to any allocation strategy for Spot / On Demand instances
  - Then, delete the instance with the oldest launch configuration, with the oldest launch template
  - Finally, select instances closest to the next billing hour
- Lifecycle Hooks: Perform custom actions as ASG launches or terminates instances
- Launch Configuration: Instance configuration template that ASG uses to launch instances
  - Contains AMI ID, key pair, instance type, securite groups and a block device mapping
  - Cannot modify once created: must create a new one, modify ASG to use the new one, then delete the old one
  - ASG is associated with 1 launch configuration at a time
  - Existing instances are not updated: must terminate them so they are replaced by the ASG
- Launch Template: Specifies instance configuration information (able to have multiple versions of a template)
  - Supports a mixture of Spot and On Demand instances (provision capacity across multiple instance types)
- Limitations: Spans AZs, not regions
  - Want to delete an ASG with running instances, the instances are terminated and the ASG will be deleted

    - Data is not automatically copied from existing instances to new dynamically created instances
- If an ASG is not terminating an unhealthy instances:
    - Health Check grace period for the instance has not expired: health checks can complete before the grace period expires but ASG does not act before the expiration time
        - ASG uses the grace period to determine how long to wait before checking the health status
    - Instance is in 'impaired' status: Wait a few minutes for the instance to recover (if not enough metrics data)
    - Instance failed the LB health check status: by default, the auto scaling of instances does not use the results of the LB health checks to determine the instance health status (must set instance health check type to LB)

Route53

- Managed DNS service: converts domain names to IP addresses
    - Contains a collection of rules and records to help clients understand how to reach a server through its domain name (can also manage the reservation of Internet domain names)
    - Connects user requests to AWS: resolver automatically answers DNS queries for local VPC domain names for instances
    - Integrate DNS resolution between resolver and on-premise network by configuring forwarding rules
    - By default: a single account can manage 50 domain names (can be raised by contacting AWS)
    - Supports and shows aliases for all domains
    - Available universally across AWS management console
- Zone Apex Record: DNS record at the root (apex) of a DNS zone
    - Configure DNS zone apex record to point to the LB by creating an A record aliased name to LBs DNS name
- Common Records:
    - A: domain hostname to IPv4 (lock record to a specific IP address)
    - AAAA: domain hostname to IPv6
    - CNAME: hostname to hostname (for non root domain)
        - Maps DNS queries for the name of the current record to another domain / subdomain (charges for queries)
            - Points to a different URL which can be resolved further by DNS
        - Cannot be used as apex record: protocol does not allow CNAME record as top node of a DNS namespace
        - If primary instance fails, the CNAME is switched from the primary to the standby instance
    - ALIAS: hostname to AWS resource (for root and non root domain)
        - More often recommended, no charge for queries
        - To route domain traffic to LB: use Route53 to create an ALIAS record that points to your LB
- Time To Live (TTL in seconds): length that a DNS record is cached on either the resolving servers or the users own cache
    - Cache DNS query response to not overload DNS and wait for TTL to expire before DNS record updates
    - Low TTL: the faster the DNS changes propagate across the Internet
    - High TTL: the longer changes take effect so must wait until it expires for new requests to perform DNS query
    - Updated Route53 record to point the domain name from the old to the new LB but users are still not being redirected to the new LB: check the TTL value (tells clients how long to cache DNS values for)
- Health Checks: if the instance is unhealthy, then Route53 stops sending traffic to that instance
    - URL/IP address of an instance is unhealthy if it fails 3 health checks in a row
- Routing Policies:
    - Simple Routing Policy: redirects to a single resource
    - Weighted Routing Policy: controls the percentage of the requests that go to a specific endpoint and redirects a part of the traffic based on weight
    - Latency Routing Policy: redirects to the server that has the least latency closest to us
        - Latency is evaluated in terms of user to region - Use when latency of users is priority
    - Failover Routing Policy: contains two instances (primary and secondary) and monitors health of primary instance
        - Use for systems that should automatically route live traffic to disaster recovery environments only if primary application has an outages: must add a health check on primary service endpoint
            - Configure Route53 to direct DNS queries to the secondary when primary is unhealthy
            - Configure NACL and route table to allow Route53 to send requests to secondary endpoints
    - Geolocation Routing Policy: routing based on the users geographic location
        - Chooses resources based on the user's location: the location that DNS queries originate from
        - Redirects distribution of content only to locations in which there are distribution rights
    - Geo Proximity Routing Policy: routing based on geographic location of users and resources
        - Choose amount of traffic sent to a resource by specifying a bias value (changes size of location)
        - Prevents users in certain locations from accessing content being distributed via CloudFront
- When routing traffic using Route53 to a website hosted in an S3 bucket, must have:

    - Registered domain name, Route53 as a DNS service for the domain, S3 bucket name = domain name
- VPC contains 2 EC2 instances in different AZs: both are running web servers hosting the same content
    - Web servers will be accessible via the internet
    - Option 1: Set up a load balancer and place your instances behind it: configure a Route53 Alias record to point to the resource of the ALB (Alias when creating a record in Route53 to other AWS resources)
    - Option 2: Assign each instance with an EIP: configure a Route53 A multivalue record with both EIPs and health checks (attach EIPs directly to instances with A record achieves similar results but without the cost of an ALB)

Data Transfer

Storage Gateway

- Bridge on-premise and cloud data for an integrated on-premise application with unlimited cloud backend
    - Provides low latency by caching frequently accessed data on-premise while archiving data in the cloud
- File Gateway: operates via NFS/SMB protocol to store files in S3 over a network file system mount point in supplied VM
    - File system mount on S3: most recently used data is cached on file gateway (not for parallel access)
    - Native AWS service that acts as a file system mounted on an S3 bucket
    - Supports a file interface into S3 and combines a service and virtual software appliance: use combination to store and retrieve objects in S3 using industry standard file protocols like NFS and SMB
    - The gateway is deployed into on-premise environment as a virtual machine running on a hypervisor and provides access to objects in S3 as files or file share mount points
- Volume Gateway: operates via iSCSi to store copies of hard disk drives or virtual hard disk drives in S3
    - Data written can be asynchronously backed up into EBS as snapshots of content for incremental backup
    - Provided cloud storage volumes that you can mount as iSCCI devices from on-premise servers
    - Stored Volumes: store data locally on-premise and backup data to AWS as secondary storage
        - For low latency access to your entire dataset and durable/inexpensive offshift recoverable backups
        - Configure on-premise to store all data locally, backup point in time snapshots of this data to S3
    - Cached Volumes: does not store entire dataset locally, only stores read often data locally
        - AWS is used as a primary data source to store full volumes of data
        - Local hardware is used as a caching layer for ore frequently accessed logs that need low latency
        - Allow you to store your data in S3 and retain a copy of frequently accessed data subsets locally
        - Offers substantial cost savings on primary storage and minimize the need to scale your storage on-premises: retain low latency access to your frequently accessed data
- Tape Gateway: backup processes using physical tape but it must be installed on the data center
    - Virtual tape library replaces physical tapes on-premises with virtual tapes without changing the existing backup workflows
    - Migration to AWS environment: want to have backup continuity and the ability to continue to access old tape archives
        - Use Tape Gateway (VTL) solution: initially installed in the on-premises environment utilizing the existing enterprise backup products to start the transition without losing access to the existing backups and archives
        - During the migration, most of the backup cycles will be replaced by new VTL and VTS tapes
        - During transition, a second VTL solution could be commissioned in the customers new VPC and integrated with the existing VTS so at the same time, the existing enterprise backup solution can be used to perform tape to tape copies to migrate the archives from tape to CTL/TLS virtual tape

DMS (Database Migration Service)

- Migrate from on-premise to cloud: on-premise database remains fully operational during migration (minimize downtime)
    - Create EC2 instance to perform the replication tasks
- If source DB and migration DB do not have the same engine, use SCT (Schema Conversion Tool) to convert schema from one origin to another (streams existing data files and outgoing files updated from S3 and Kinesis)

DataSync: Online (over the internet) data transfer

- Move large amounts of data from on-premise to AWS to synchronize data to S3, EFS, FsX
    - Simple and fast transfer over the Internet or via AWS DX Link (only need to deploy DataSync agent, connect to the file system, select AWS storage resources and start moving data)
        - Replace AWS DX (for privacy) to rapidly move data over a service endpoint (the internet)
    - Scheduled replication tasks (not continuous) leverage DataSync agent to connect to your systems

Other Options

- SMS: Supports incremental replication of on-premise servers in AWS as a backup tool for disaster recovery

- App Discovery Service: enterprise customers plan migration projects by gathering information about on-premise data centers (install agent less connector: builds server utilization map and dependency)
- VM Import/Export: migrate existing applications into EC2 instance to create a disaster recovery strategy
    - Used to export AWS VMs to on-premise: use to transfer large amounts of data from physical storage devices into AWS (mail portable storage devices to AWS, they will transfer data directly off devices to AWS network)

Transfer Large Data Sets in AWS: 200TB of data using 100Mbps internet connection
- Option 1: Over the internet or Site to Site VPN has immediate setup but takes 6 months to transfer the data
- Option 2: Transfer over DX for 1 Gbps but has a long one time setup (1 month) and takes 18 days to transfer
- Option 3: Snowball for one off transfers but must order 2-3 devices in parallel (takes 1 week)
- Option 4: For ongoing replication and transfers, use Site to Site VPN, DX or Data Sync
- Reduce data transfer costs: ensure that instances communicating with instances in the same region have private IPs

Monitoring

CloudWatch Metrics
- Metrics are variables to monitor (time ordered set of data points that are published too CW)
    - Provides metrics for every service: custom metrics with standard 1 minute or high resolution with 1 second
    - 'Requests Per Minute': custom metric that must be made using CW to build alarm to scale ASG
    - Default instance is enabled for basic monitoring but can enable detailed monitoring
- AWS can see the instance but not inside the instance: sees that instance has memory but cannot see how much
    - Design a custom metric for Memory usage (but not CPU usage: can see how much is being used, not for what)
- Create CW dashboard for a global view of graphs from different regions
- EC2 Detailed Monitoring: monitoring graphs for every instance (charged per metric sent to CW)
    - Default every 5 minutes but can enables for every 1 minute
- Enhanced monitoring metrics that CW gathers for RDS database instances: RDS, RDS child and Os processes
- CW Logs: applications send logs to CW for specific functions (from there, can be sent to S3 and Elasticsearch)

CloudWatch Agent
- By default, no logs go from EC2 instances to CW: must install and start agent in EC2 instances that pushes memory and disk utilization data (view custom metrics in CW console that is installed on Linux and Windows)
- To monitor the available swap space of each EC2 instance: install CW agent on each instance and monitor the SwapUtilization metric
- Logs Agent: older but automates sending log data to CW logs from EC2 instance

CloudWatch Alarms
- Trigger notifications for any metric: works only for instance status check failures (not if instance was self terminated)
    - OK state: not doing anything
    - Insufficient state: not enough data for alarm or metric is missing a data point
    - Alarm state: threshold being passes
- If instance fails health check it is recommended to reboot the alarm action
    - Automatic recovery process recovers instances for up to 3 failures per day and only for EBS volumes
- Status Check on EC2 instances: set up an alarm to monitor, trigger instance recovery for over 90% of deployed instances
- CloudWatch Event: Scheduled jobs for events of event patterns (allow rule definition about the change)

CloudTrail
- Provides governance, compliance and audit for AWS account to get a history of all events and API calls made within
    - Event log files are encrypted using S3-SSE

AWS Config
- Helps with auditing and recording compliance of AWS resources (records configuration changes over time)
- Use to check compliance on password policy

CloudWatch	CloudTrail	AWS Config
• Performance metrics, utilization alerts, dashboard for events	• Record API calls made within an AWS account	• Record configuration changes • Evaluate resources with

● Log aggregation and analysis	● Define trails for resources	compliance

Disaster Recovery

- Disaster recovery is about preparing for and recovering from a disaster:
  - RPO (Recovery Point Objective): how often backups run
  - RTO (Recovery Time Objective): how long it takes to recover after a disaster
- Backup & Restore: schedule regular snapshots from cloud to backup storage (restore data using AMI or snapshot)
  - Cheapest, only cost is storage, high RPO, low RTO
- Pilot Light: small version of the app is always running in the cloud (only the critical core app)
  - Minimal version of environment is always in the cloud: recovery is added all non critical systems
- Warm Standby: scaled down version of fully functional system up and running, ready to scale to production load
  - More costly since ELB and ASG are already running but low RPO
- Multi-Site / Hot-Site Approach: full production scale environment is running on AWS (very low RTO)
  - Route53 can send requests to both on-premise and the cloud for an active-active setup

Other Services

RAM (Resource Access Manager)

- Free service that allows the sharing of AWS resources that you own with other accounts to avoid resource duplication
- VPC Sharing allows multiple AWS accounts to place their application resources (EC2, RDS, RedShift, Lambda) into shared and centrally managed VPCs (to allow instances in different accounts to communicate privately)
  - To set it up: the account that owns the VPC must share one or more of its subnets with other accounts that belong to the same organization in AWS Organizations.
    - After the subnet is shared, the other accounts can CRUD their application resources shared with them in the subnets (other accounts cannot CRUD resources that belong to other accounts or the owner)
  - Share VPCs to leverage implicit routing within a VPC for applications that require a high degree of interconnectivity and are within the trust boundaries: reduces the number of VPCs managed while using separate accounts for billing

CloudFront

- Cloud distribution network that serves cached content for increased global performance
  - It is a regional edge cache that brings content closer to users (even when content is not popular)
    - Content is cached for TTL: can be cleared beyond TTL for extra cost
  - Use field level encryption to protect sensitive data for specific content
    - Control versions of files served from distribution: invalidate file or give versioned file name
  - Control how long objects stay in the cache before CF forwards another request to origin by reducing duration (dynamic content) or increasing duration (better performance / reduce load on origin)
- Components:
  - Edge Locations: Cache Endpoints
    - Requests are routed/cached in the nearest edge location: put in from of LB to reduce cost and workload
    - To delete a file form an edge location: remove the file from the origin servers and then set expiration period to 0 or via invalidation API
    - CF serves objects from edge locations until the specified cache duration passes (Cache-Control header)
  - Origin: original source location where content is stored form which CF gets content to serve viewers
    - Dynamic content and proxy methods go directly to origin
    - Route multiple origins based on content type and use origin group with primary/secondary origins to configure CF for failover
    - To serve content globally that is stored in S3 but not publicly accessible from S3: create an origin access identity (OAI) to CF and grant access to objects in S3 buckets to that OAI
      - Use OAI for sharing private content: it is a virtual user that gives CF distribution permission to fetch a private object from your origin
    - If users take a long time to log in to an application with HTTP 504 errors:
      - Set up an origin failover by creating an origin group with 2 origins: specify a primary and a secondary that CF automatically switches when primary retains specific HTTP code failure
  - Distribution: arrangement of edge locations from origin
    - Web Distribution: Websites and normal cached items
    - RTMP: streaming content and Abode
- Checks cache for required files: if files are in the cache, CF returns them to the user

    - If not in cache: CF compares the request with the distribution specifications and forwards the request for the files to your origin server for the corresponding file type (to S3 for image files, to HTTP server for HTML files)
        - The origin servers send the files back to the edge location
        - As soon as the first byte arrives from the origin, CF begins to forward the files to the user
        - CF adds the files to the cache in the edge location for the next time
    - If a resource does not exist on the CF server, it is because CF has cached that resource to the edge location
- Customize content that CF web distribution delivers to users with lambda@edge to allow lambda to execute authentication processes in AWS locations closer to users
- To provide access to multiple private files only to paying subscribers without having to change their current URLs:
    - Use signed cookies to control who can access the private files in CF by modifying your application to determine is users should have the access to the content
    - Send members required set-cookie headers to viewers so only they unlock content
- Signed URLs: Use RTMP distribution, restrict access to individual files (if client doesn't support cookies)
- Signed Cookies: provides access to multiple restricted files and don't want to change current URLs

Amazon MQ: use when migrating IoT solutions to the cloud (no need to re engineer protocols)

AWS Global Accelerator
- Service to create accelerators to improve availability and performance of your applications for local and global users
    - Provides two static AnyCast IP addresses that act as a fixed entry point to your endpoints in regions
    - Traverses AWS Internet Network to get to your instance, LBs of different endpoints with more reliability
- Improves the availability of applications offered to global users by providing traffic management across multiple regions:
    - Eliminates the complexity of managing IP addresses for different regions and AZs by associating static IP addresses provided by Global Accelerator to regional resources and endpoint
    - Improves performance for applications over TCP/UDP by proxying packets at the edge to applications running in one or more AWS regions using the AWS Global Network
- Controls a portion of traffic directed to each endpoint: traffic dials control a percentage of the traffic
- Leverages the AWS Global Internal network infrastructure to route application traffic for routing optimizations
    - Keeps latency and packet loss low and improves performance by lowering jitter
    - Reduces the number of hops to get to resources
    - Talks to the closest edge location for public ALB, for TCP/UDP and other non-HTTPS use cases
- Uses AnyCast IP (all servers hold the same IP address and the client is routed to the nearest one):
    - To direct user traffic to the nearest endpoint and route traffic the closest edge location
    - Endpoint weights determine the proportion of traffic directed to endpoints in group
- A network sonze services the static IP addresses for your AWS Global Accelerator from a unique IP subnet
- When you configure an accelerator, GA by default allocates two IPc4 addresses for it so if one IP addresses from a network zone becomes unavailable due to IP addresses blocking by certain client networks
    - Then client applications can retry on the healthy static IP address from the other isolated network zone

Amazon Beanstalk
- Controls the resources powering an application (at no additional charge):
    - No need to deploy code, manage infrastructure, uploading applications, provisioning resources
    - Supports the deployment of applications from Docker
    - Stores application files in S3 and server files in either S3 or CW Logs
- To bring down Beanstalk deployment time: create Golden AMI with static installation setup and EC2 user data to customize the dynamic installation at boot time

Amazon Glue
- Severless, manage ETL service for preparing data (significant efforts to write custom migration scripts)
- Glue Data Catalog: datasets in AWS
    - Define Glue Crawlers to connect to various sources and write metadata to the catalog

Directory Services
- Connects AWS resources with on-premise AD
- Microsoft AD is software in Windows to manage servers and AWS DS provides AD on AWS to manage users locally
    - All Windows machines are connected to domain control so users on every machine can join domain controller
    - Run directory aware workloads, configure trust relationships to provide users and groups with access to resources

    - AD trust extends existing AD inside AWS to on-premise directory services
    - Use when you want to SSO (single sign on) to log into any domain joined EC2 instance
- A company is currently assigning rules with groups in corporate AD but wants to give AWS access using identity and role based access control: use AWS DS AD connector to integrate the already existing corporate AD

CloudFormation
- Declarative way of outlining AWS infrastructure for any resource: define services in a template and run it
    - AWS creates a stack and all services are automatically created in order of how it was specified
    - CRUD infrastructure written in YAML and JSON
    - Building blocks (only required section) declare resources, parameters, mappings, output, conditionals, metadata
- StackSets: extends functionality of stacks by enabling you to CRUD stacks across multiple accounts and regions with a single operation (create stacks in AWS accounts across regions using a simple template)
    - Use admin account of AWS Org to define and manage templates to use it as a basis for provisioning stacks into selected target accounts of an AWS Org across specified regions

AWS SWF (Simple Workflow Service)
- Coordinates work amongst applications (runs on instances so its not serverless): ensures a task if never duplicated
    - Orchestrate multiple AWS services: automatically triggers and tracks each step (retires when there are errors)
    - Web service: easy to coordinate work across distributed application components (messages are tasks)
    - Can have out of band and sequential tasks
- Domains provide a way of scoping SWF resources within your AWS account: all the components of a workflow must be specified to be in a domain
- Keeps track of all tasks and events in an application

EMR (Elastic Map Reduce)
- Creates Hadoop clusters to analyze and process vast amounts of data: clusters are made from 100s of instances
- Managed cluster platform that simplifies running big data frameworks on AWS process to analyze vast amounts of data

Amazon EventBridge
- Decouple system architecture to build an application that reacts to events from SaaS applications or AWS services
- Only event based service that integrates directly with third party SaaS partners

Cost Explorer
- Billing service to visualize, understand, manage AWS costs and usage over time: create custom reports that analyze costs
- Identify over/underutilized instances that may be downsized on an instance by instance basis with the same instance family

AWS Systems Manager
- Unified UI to track and resolve operational issues across your AWS applications and resources from a central place
- Configure EC2 instances without establishing RDP or SSH connection to each instance
- Use Run command to remotely and securely manage configuration of instance

AWS Trusted Advisor
- Gives real time guidance and recommendations to help provision resources following best practices
- Inspects environment and makes recommendations for cost optimization, performance, security, fault tolerance

AWS X-Ray
- Traces and analyzes user requests as they travel through API Gateway APIs to underlying services
    - Collects data across accounts: vows hot the application and underlying services perform to identify and troubleshoot the cause of issues and errors (provides full view of requests)

Amazon LightSail
- Easiest for hosting a website: requires the least amount of AWS knowledge
- Creates EC2 instances where everything is already installed and available: bandwidth is included

Amazon VPC Console Wizard
- VPC with single public subnet: involves IG to enable communication
    - Good for single tier and public facing web applications

- VPC with public and private subnets (NAT): run a public facing application while maintaining backend servers that are not publicly accessible (set up security and routing so web servers can communicate with the database servers)
    - Good for multi tier websites
- VPC with public and private subnet (Site to Site VPN Access): extend the network to cloud and directly access the Internet from your VPC (turn a multi tiered application into a scalable web front end in a public subnet)
    - House your data in a private subnet that is connected to your network by VPN connection
- VPC with private subnet only (Site to Site VPN Access): no IG to enable communication over the Internet
    - Use to extend your network into your cloud using your AWS infrastructure without exposing network to the internet

Others:
- Batch: supports multi node parallel kobs of single node jobs that span multiple EC2 instances
- Parallel Cluster: open source cluster management tool to deploy HPC on AES
- Code Commit to commit code, Code Build to build and test code, Code Pipeline: pipeline orchestrator for deployment
- Elastic Transcoder: convert media files stored in S3 to various formats and optimized for different devices
- AppSync: store and synchronize data across mobile/web pages in real time (uses GraphQL)
- Backup: managed backup service to automate backups across AWS (supports point in time recovery on demand)
- Step Functions: serverless visual workflow to orchestrate lambda functions (JSON state machine, max time of 1 year)
- OpsWorks: managed software (Chef/Puppet- consistent deployment/configuration): performs automatic server configuration
- AWS Workspaces: managed, secure Cloud desktops that can be accessed anywhere, anytime, from any device
- OpsHub: GUI to manage Snowball devices to rapidly deploy edge computing workloads,simplify data migration to the cloud
- Business support plan: has a maximum response time of < 1 hour for ‘production system down’ cases

High Level Ideas
- Vertical Scaling: increase size of instance (upgrade current resources)
- Horizontal Scaling: increase the number of instances (adding more resources)
- High Availability: host instances in 2 AZs for system to keep providing services while in turbulent conditions
- Fault Tolerant: host instances on 3 AZs for system to system loss but still functions the same way
- Active Passive Failover: primary resource(s) are available most of the time and secondary resource(s) are on standby
    - To create with 1 primary and 1 secondary record: create the records and specific failovers for routing policy
- Active Active Failover: all resources are available most of the time (all records have the same name, type, routing policy)
- Blue Green Deployment: releasing applications by shifting traffic between two identical environments running different versions of the application (blue is currently running, green is the new): test features in green without affecting the blue
- AWS customers can carry out security assessments or penetration tests against their AWS infrastructure without prior approval for 8 services only: must request authorization for other simulated events: Customer Service Policy for Penetration Testing
- 4 levels of support; free (billing rebate), enterprise, business, developer

Distribute Paid Content
- CloudFront Signed URLs: allows access to a path (no matter the origin)
    - Account wide key pair: only root can manage a global and secure way (allows distribution of paid content from S3 if bucket is secured to exchange data)
    - For clients to talk to CF to only get access to content there: authenticate client with Cognito (use signed URLs to distributed content), create API gateway, client invokes gateway and get signed URLs
- S3 Presigned URLs: issues a request as the person who presigned the URL and uses IAM key of signing IAM principal

Caching Strategies
- CloudFront: caching at the edge for a quick response (if something changes in the backend, the cache can be outdated)
- API Gateway: regional cache with network lag (caching through Elasticache so query result is stored in shared cache)

Blocking IP Addresses
- 1st Defense: NACL in VPC to create deny rule for IP address
- 2nd Defense: SG of EC2 instance but cannot have deny rules (only allow): not useful if the application is global
    - If you know the subnet of IPs that can access the VPC, can state it
- 3rd Defense: Firewall on EC2 to block requests from within the software
    - Use ALB + WAF + CloudFront: CF sits in front of ALB and ALB needs to allow CF public IPs
    - Use CF geo versioning restriction if attacked by a country or attach WAF to it for IP address filtering