Region	Cluster of data centers	ap-southeast-2
Availability Zones (AZ)	Each region has many AZ Each AZ is ≥ 1 data centers w redundant power, networking & connectivity Separated from each other Connected w high bandwidth, ultra low-latency networking	ap-southeast-2a, ap-southeast-2b, ap-southeast-2c,
Point of Presence (Edge location)	Content delivered to users w low latency	216 locations globally

IAM	Identity and Access Management, Global service	Root account created by default,
Users	People within an org, can be grouped	Can belong to multiple/no group
Group	Can only contain users, not other groups	
Policies	JSON documents assigned to users or groups Defines the permissions allowed Least privilege principle	Version, ID, Statement {Sid, Effect, Principle, Action, Resource}
Password Policy	MFA - Password + security device (virtual/physical) Access key - Access key id + Secret Access Key	Management Console - MFA CLI - access keys, SDK - access keys
IAM role for services	Certain AWS service require permission to perform actions on our behalf	E.g. EC2, lambda fn, CloudFormation
Security Tools	Credentials Report (account) - list all account users + status of their credentials Access Advisor (user) - service permission granted to user + when last accessed those services	

EC2	Elastic Compute Cloud = IAAS Consists of: Renting virtual machines (EC2 instances), Storing data on virtual drives (EBS volume), Distributing load across machines (ELB), Scale services using an auto-scaling group (ASG)	Configurations: OS, CPUs, RAM, Storage Space (Network attached: EBS & EFS, Hardware: EC2 Instance Store), Network card (speed, Public, IP address), Firewall Rules (security grp), Bootstrap script (configure at 1st launch -> install updates, software, import libraries...; EC2 User Data)
	Instance Types: General Purpose, (Compute, Memory, Storage) Optimized, Accelerated Computing, General Purpose: balance btw compute, memory, networking (web servers/code repositories) Compute Optimized (gaming servers, machine learning) Memory Optimized: fast performance for workloads that process large data sets in memory (high performance databases, processing of big unstructured data)	Storage Optimized: high sequential read & write access to large data sets on local storage (high freq online transaction processing sys, cache for in-memory databases, distributed file sys) Naming Convention: e.g. m5.2xlarge m: instance class, 5: generation, 2xlarge: size within instance class
Security Groups	Control how traffic is allowed in/out of EC2 instances Only contain allow rules Rules can reference by IP address, or security groups (i.e. can attach security group to another instance to allow access) Can be attached to multiple instances Is locked down to a region/VPC combination If application time out - usually due to security group issue	Regulate: access to Ports, authorised IP ranges (IPv4, IPv6), inbound and outbound networks 21 = FTP (file transfer protocol) 22 = SSH - log into linux instance 22 = SFTP (secure file transfer protocol) - upload files using SSH 80 = HTTP - unsecured websites 443 = HTTPS - secured websites 3389 = RDP (remote desktop protocol) - log into windows instance
EC2 Purchas-ing Options	On-demand instances - short & un-interrupted workload, highest pricing, pay by second Reserved (1 & 3 years) - Reserved instances (long workload): reserved instance attribute, for databases; Convertible Reserved Instances (long workload + flexible instances): can change instance type Savings Plan (1 & 3 years) - commitment to an amt of usage (e.g. $10/hr for 1 year), instance family & region locked, size & OS & tenancy can change, long workload Spot instances - short workload, cheap, can lose instance if your max price < curr spot price(not reliable, highest bid), suitable for workload resilient to failure (batch jobs, data analysis,...) Dedicated Host - book entire physical server, control instance placement, suitable for server-bound software licenses, if have complicance requirements Dedicated Instances - no other customer will share your hardware, no control of instance placement	

	Capacity Reservations - reserve On-demand instances capacity in any AZ for any duration, no timing commitment but no discounts, charged even when not using, but always available for use	
Spot instance	Spot request - define max price, # of instances, request type (1 time/persistent), Valid from, valid until	Can only cancel spot instance request that are open, active or disabled 1st cancel request, then terminate instance
Spot fleet	Spot fleet = set of Spot instances + (optional) on-demand instances Spot fleet will try to meet target capacity w price constraints	Strategies: lowest price, diversified, capcityOptimized

IP address	IPv4, IPv6 (newer) Public IP: can be identified on internet, unique across whole internet, can be geo-located easily Private IP: can only be identified on private network, unique in private network, but can have same IP in different private networks, connect to internet using an internet gateway (proxy), only a specified range of IPs can be used as private IP	
Elastic IP	Public IP of EC2 instance can change after stop and starting If need to have fixed public IP for instance, use elastic IP	Only 1 instance at a time
EC2 Placement Groups	Cluster: cluster instances into a low-latency group in a single AZ, all on same rack (can all fail at same time) Spread: spread instances across underlying hardware (max 7 instances per group per AZ), for critical applications, 1 instance on 1 rack Partition: spread instances across many diff partitions/racks, instances in a partition is do not share racks w the instances in the other partitions, can have multiple instance on 1 rack	
Elastic Network Interface (ENI)	Logical component in a VPC that represents a virtual network card Attributes: - Primary private IPv4, 1 or more secondary IPv4 - 1 Elastic IP per private IPv4, - 1 Public IPv4, - 1 or more security groups Bound to a specific AZ, can move ENI across EC2 instances	
EC2 Hibernate	Stop instance: data on disk (EBS) is saved until next start Terminate: any EBS volumes (root) set up to be destroyed is lost On start, EC2 User Data script is run, and OS boot up Hibernate: - RAM state is preserved (written to a file in root EBS volume), - instance boot faster (OS not stop/restarted),	Hibernate used for long-running processing, saving RAM state, services that take time to initialise Root volume (EBS) must be encrypted
EC2 Nitro	Underlying platform for next gen of EC2 instances Better underlying security	- Better networking options (HPC, IPv6), - Higher Speed EBS (64,000 EBS IOPS/max 32,000 for non-Nitro)
vCPU	Multiple threads can run on 1 CPU (multithreading) Each thread is represented as a vCPU	Can decr # of CPU to decr licensing cost
Capacity Reservation	Manual/planned end date for reservation, No need for 1/3 year commitment Combine w Reserved Instances and Savings Plans for cost savings	

^SECONDARY

EC2 Instance Storage	EBS (elastic block store) volume is a network drive you can attach to different instances Allows instances to persist data, even after termination Mount to 1 instance at a time (at CCP level)	Bound to a specific AZ Have a provisioned capacity (size, IOPS) Delete on termination attribute (default on for root EBS when instance terminate, default off for other attached EBS volume)
EBS Snapshot	Recommended to detach volume then take snapshot (backup) Can copy snapshots across AZ or region EBS Snapshot archive - move snapshot to 'archive tier', cheaper but take 24-72hr to restore from archive	Recycle Bin for snapshot - set up rules to retain deleted snapshots - specify length of retention
Amazon Machine Image (AMI)	AMI = customization of EC2 instance - add own software, OS -> faster boot as all software pre-packaged - Built for a specific region (can copy to other regions)	Public AMI: AWS, Own AMI Marketplace AMI: made by someone else
	Golden AMI: Install app, OS dependencies,... beforehand and launch EC2 instance from this AMI	
EC2 Instance Store	High-performance hardware disk, better than EBS volume EC2 Instance Store lose their storage when instance stopped Good for buffer/cache/temporary content	Risk of data loss if hardware fails Highest IO compared to EBS, EFS
EBS Vol types	1. gp2/gp3: general purpose SSD that balance price and performance, low latency, IO incr if size incr	

	2. Provisioned IOPS (PIOPS) io1/io2: highest performance SSD for low-latency and high throughput workloads, for databases workloads (sensitive to perf and consistency), can incr IO independently 3. st1: low cost HDD for frequently accessed, throughput- intensive workload, for big data, logs processing, data warehouses 4. sc1: lowest cost HDD for less frequently accessed workload, for data that is infrequently accessed - Characterised by size, throughput, IOPS, - only gp2/gp3, io1/io2 can be used as boot volume	
EBS Multi-Attach	Only for io1/io2 - attach same EBS vol to multiple EC2 instances in same AZ For higher application availability in clustered Linux applications (Teradata), app must manage concurrent write ops Must use a file sys that is cluster-aware (not XFS, EX4,...)	
EBS Encryption	Has minimal impact on latency, encrypted w KMS Ununcrypted EBS vol -> snapshot -> encrypt snapshot (using copy) -> create new vol from snapshot (auto encrypt) -> attach encrypted vol to original instance	
EFS (more ex than EBS)	Elastic file sys: can be mounted on many EC2 instance across AZ For content management, web serving, data sharing, Wordpress Use security grp to control access to EFS Sys scale automatically, pay per use Performance mode: general purpose - latency-sensitive (web server), max I/O - higher latency, throughput (big data) Throughput mode: Bursting (scales w storage size), Provisioned (set throughput regardless of storage size)	NFS on linux POSIX-compliant Storage Tiers: Standard, Infrequent access (EFS-IA) - lower price to store, but cost for retrieving. Enable EFS-IA w lifecycle policy (move file after not accessing for x days) Availability and durability: Regional - Multi AZ (for production), One AZ - for development, compatible w IA

Instance storage: physical storage for EC2 instance (high IOPS)

Scalability	- system can handle greater load by adapting: vertical scalability & horizontal scalability (elasticity) Horizontal scalability similar to high availability High availability = running sys in ≥ 2 diff AZ, used to survive a data center loss	Vertical: incr size of instance, used for non-distributed sys (databases) Horizontal: incr num of instance, used for distributed sys (applications)
Elastic Load Balancing (ELB)	Load balances (LB) = servers that forward traffic to multiple servers (EC2 instances) downstream Expose a static single point of access (DNS) to ur app Seamlessly handle failures of downstream instances Do regular health checks to ur instances Provide SSL termination (HTTPS) for ur website	Enforce stickiness with cookies High availability across zones (Multi-AZ) Separate public traffic from private ELB managed by AWS: - guaranteed to work, less work for u, - takes care of upgrade, maintenance, high availability
	Health check: enable ELB to know if instances it forward traffic to is working Health check is done on a port and route (/health is common) 200 = http status code for ok. If response ≠ 200, then instance is unhealthy	
	1. Classic load balancer (CLB) (v1 - old gen, 2009): HTTP, HTTPS, TCP, SSL (secure TCP) 2. Application load balancer (ALB) (v2- new gen, 2016): HTTP, HTTPS, Websocket 3. Network load balancer (NLB) (v2 - new gen, 2017): TCP, TLS (secure TCP), UDP 4. Gateway load balancer (GWLB) (2020): operate at layer 3 (network layer) – IP protocol Some load balancer can be set up as internal (private), or external (public) ELBs Users access ELB from anywhere, but EC2 instance only allow access from ELB	
CLB	Support TCP (layer 4), HTTP & HTTPS (layer 7) Health check are TCP or HTTP based	Load balancers have a fixed hostname (DNS)

ALB	Support HTTP (layer 7) Load balancing to multiple HTTP applications across machines (target groups) Load balancing to multiple applications on the same machine (containers) Support for HTTP/2 and websocket Support redirects (e.g. from HTTP to HTTPS) Have fixed hostname (DNS)	Routing tables to diff target groups: - based on path in URL (eg.com/users & eg.com/posts) - based on hostname in URL (one.eg.com & other.eg.com) - based on query strings, headers (eg.com/users?id=123&order=false) ALB suited for micro services and container-based application (Docker, Amazon ECS) - port mapping feature to redirect to a dynamic port in ECS In contrast, would need multiple CLB per application	
	Possible Target groups: - EC2 instances (can be managed by an Auto-Scaling Group) – HTTP		The application server (instances) don't see the IP of client directly, but the private IP of the ALB:

	- ECS tasks (managed by ECS) – HTTP - lambda fn – HTTP request is translated into JSON event - IP addresses (private IP) Health checks are at the target grp level	- true IP of client is inserted into the header X-Forwarded-For - can also get Port (X-Forwarded-Port) and proto (X-Forwarded-Proto)
NLB	NLB (layer 4) allows: - handle millions of requests per sec - forward TCP & UDP traffic to instances - less latency ~ 100ms (vs 400 ms for ALB) Used for extreme performance, TCP or UDP traffic	1 static IP per AZ, & static DNS & support assigning elastic IP Possible Target groups: EC2 instances, IP addresses (private IP only), ALB
GWLB	Deploy, scale & manage fleet of 3rd party network virtual app in AWS (e.g. firewall, payload manipulation, ...) Operate at Network layer (layer 3) – IP packets Use the GENEVE protocol on port 6081	Basically combines: Transparent Network Gateway (single entry/exit for all traffic) & LB (distribute traffic to virtual apps) Possible Target groups: EC2 instances, IP addresses (private IP only)
Sticky Sessions (Session Affinity)	Stickiness: same client always redirected to same instance behind a LB Works for CLB and ALB	"Cookie" used for stickiness has an expiration date you control Used to make sure user doesn't lose his session data Enabling stickiness may cause imbalance on the EC2 instances
Sticky Sessions – Types of cookies:	1. Application-based cookies: - custom cookie: generated by target, can include custom attribute, cookie name must be specified for each target group, (AWSALB, AWSALBAPP, AWSALBTG cannot be used as reserved for ELB already) - application cookie: generated by LB, cookie name is AWSALBAPP)	2. Duration-based cookies: - generated by LB - cookie name is AWSALB for ALB & AWSELB for CLB
Cross-Zone Load Balancing (CZLB)	Each LB instance distributes evenly across all registered instances in all AZ (even if the EC2 instance are behind diff ALB)	w/o CZLB: requests are distributed in the instances of the node of the ELB (EC2 instance receive diff load according to num of instances in each AZ)
	ALB: CZLB always on (free for inter AZ data) CLB: disabled by default (free for inter AZ data)	NLB: disabled by default (pay $ if CZLB enabled)
SSL / TLS	- Secure Sockets Layer (SSL): used to encrypt connection - SSL certificate: allows traffic btw client and LB to be encrypted in transit (in-flight encryption) - Transport Layer Security (TLS): newer version of SSL - Nowadays, TSL certs are mainly used (but still referred to as SSL) - Public SSL cert are issued by Cert Authorities (CA) - SSL cert have an expiration date (you set) & must be renewed	User <- {HTTPS (encrypted), over www} -> LB <- {HTTP, over private VPC} -> EC2 instance LB uses an X.509 cert (SSL/TLS server cert) Can manage cert using ACM (AWS cert manager) When setting HTTPS listener: - must specify default cert - can add list of cert to support diff domains - can use SNI (Server Name Indication) to specify hostname they reach - can set specific security policy to support older version of SSL/TLS (legacy clients)
SNI	- solves problem of loading multiple SSL cert onto 1 web server (to serve multiple websites) - require client to indicate the hostname of the target server in the initial SSL handshake	- server will then find the correct cert, or return the default one - only works w ALB, NLB (newer gen), CloudFront - does not work for CLB
	CLB: support only 1 SSL cert (must use multiple CLB for multiple hostname w multiple SSL cert)	ALB & NLB: support multiple listeners w multiple SSL cert (uses SNI to make it work)
Connection Draining	CLB: connection draining, for ALB & NLB: aka deregistration delay - Give extra time to complete "in-flight request" when instance is deregistering or unhealthy - ELB will then stop sending new requests to the instance that is de-registering	- Extra time can be set to btw 1-3600s, default 300s - Can be disabled (set value to 0) - Set to low value if requests are short
Auto-scaling groups (ASG)	- Used to scale out (add instances)/scale in - Ensure min or max of instances are running depending on load - Auto register new instances to a load balancer - min size, actual size (desired capacity), max size	Attibutes: - Launch configuration (AMI + instance type, EC2 User data, EBS volume, Security Group, SSH Key Pair) - Min / Max Size, Initial Capacity, - LB info - Network & Subnet, - Scaling policies
	- Possible to scale ASG based on CloudWatch alarms Alarm monitors a metric (eg average CPU)	Auto Scaling New Rules:

	Metrics are computed for overall ASG instances	- Target average CPU Usage, num of requests on ELB per instance, average network in/out
	Auto Scaling Custom Metric: 1. Send custom metric from app on EC2 to CloudWatch (PutMetric API) 2. Create CloudWatch alarm to react to low/high values 3. Use CloudWatch alarm as scaling policies for ASG ASG is free	ASG use Launch Configurations/Launch Templates (new) To update an ASG, must provide a new launch configuration/template IAM role attached to ASG will be attached to EC2 as well Instances under ASG will be auto created as replacement is they get terminated ASG can terminate instance marked unhealthy by ELB (and hence replace them)
Scaling Policies	Dynamic Scaling Policy: 1. Target Tracking Scaling: Easiest to setup (ave ASG CPU to stay ard 40%) 2. Simple/Step Scaling: CloudWatch alarm triggered (CPU> 70%), add 2 units, (CPU < 30%), remove 1 unit Scheduled Actions: Anticipate scaling based on known usage pattern (incr min units to 10 at 5pm)	Predictive Scaling: Continuously forecast load and schedule scaling ahead Scaling Cooldown: After scaling occurs, cooldown period (default 300s), ASG will not do any further scaling (to allow metric to stabilize) - Use ready-to-use AMI to reduce config time to serve request faster and reduce cooldown period
ASG extra	ASG Default Termination Policy (simplified): 1. Find the AZ w most num of instances 2. If multiple instances in AZ, delete the one w oldest launch configuration - ASG try to balance num of instance in ea AZ by default	Lifecycle Hooks: By default in service when instance launched in ASG. - Have ability to perform extra steps before instance in service (Pending State) - Have ability to perform extra steps before instance terminating (Terminating State)
	Launch Template vs Launch Configuration Launch Configuration (legacy): must be re-created every time Launch Template: - can have multiple versions, - Provision using On-Demand and Spot instances (or mix) - create params subset (partial config for reuse & inheritance), - Can use T2 unlimited burst feature	
	A web app hosted on a fleet of EC2 instances managed by an ASG. You are exposing this application through an ALB. Both the EC2 instances and the ALB are deployed on a VPC with the following CIDR 192.168.0.0/18. How do you configure the EC2 instances' security group to ensure only the ALB can access them on port 80? Add an inbound rule w port 80 and ALB Security Group as source	

Relational Database Service (RDS)	Managed DB service using SQL as query language Create DB in cloud managed by AWS: - Postgres, MySQL, MariaDB, Oracle, Microsoft SQL Server, Aurora Storage Auto-Scaling: - auto incr storage on RDS DB instance dynamically - Have to set max storage threshold - auto modify storage if free storage < 10% of allocated storage & low storage last ≥ 5mins & 6 hrs have passed since last modification - Useful for app w unpredictable workloads	Advantage of RDS over DB on EC2: - RDS is a managed service: • automated provisioning, OS patching, • cts backups & restore to specific timestamp • read replicas for improved read performance • multi-AZ for DR (disaster recovery) • monitoring dashboards • Maintenance windows for upgrades • Scaling capabilities (vertical/horizontal) • Storage backed by EBS (gp2/io1) - But can't SSH into RDS instances
RDS backups	Auto enabled Auto backup: - daily full backup of DB (during maintenance window) - transaction logs backed up every 5 mins - ability to restore to any point in time (Point in Time Restore) - 7 day retention (can extend to 35 days)	DB Snapshots: - Manually triggered by user - Retention of backup as long as you want
Read Replicas	- Up to 5 read replicas (kinda like a 'copy') - Within AZ, Cross AZ, or cross Region - Replication is Asynchronous, so reads are eventually consistent (if read from read replicas before they can 'update' the data, the app might read old data) - read replicas can be promoted to become a database	Use cases: - already have production app reading and writing on DB on normal load, - now have new app that wants to read, so create read replica to read from there that won't affect DB

	- App must update connection string to leverage read replicas - Read replicas in same region but diff AZ has no network cost	- only used for SELECT statement (not INSERT, UPDATE, DELETE)
Multi-AZ (DR)	- Synchronous replication, Increased availability - 1 DNS name – auto app failover to standby DB - Failover in case of loss of AZ, loss of network, instance or storage failure during failover, URL same, CNAME updated - Not used for scaling - Read replicas can be setup as multi AZ for DR	From single to multi-AZ: - 0 downtime operation (no need to stop DB) - Just click 'modify' for DB and enable multi-AZ Internally: - snapshot taken, - new DB restored from snapshot in new AZ, - synchronization established btw the 2 DB
Security	At-rest encryption: - can encrypt master & read-replicas w AWS KMS (key management service) w AES-256 encryption - encryption has to be defined at launch time - if master not encrypted, read-replicas cannot be encrypted - Transparent Data Encryption (TDE) available for Oracle and SQL server In-flight encryption: - SSL certs to encrypt data to RDS in flight - Provide SSL options w trust cert when connecting to DB - To enforce SSL: [PostgreSQL: rds.force_ssl = 1 in AWS RDS Console (Parameter grp)], [MySQL within DB: GRANT USAGE ON *.* TO 'mysqluser'@'%' REQUIRE SSL;]	Encryption Operations: Encrypting RDS backups: - snapshot of unencrypted (encrypted) DB are unencrypted (encrypted) - can copy a snapshot into an encrypted one To encrypt unencrypted DB: - create snapshot of unencrypted DB, - copy snapshot and enable encryption, - restore DB from encrypted snapshot, - migrate apps to new DB, delete old DB
	Network Security: - RDS DB usually deployed in private subnet - Leverage security groups – control which IP/security group can communicate w RDS Access/User Management: - IAM policies to control who can manage RDS - Traditional Username and pw to login to DB - IAM-based authentication can be used to login to RDS MySQL & PostgreSQL	IAM authentication: - for MySQL & PostgreSQL - no pw, just authentication token obtained through IAM & RDS API calls - auth token has lifetime of 15min - Network in/out must be encrypted w SSL - IAM to centrally manage users instead of DB - can use IAM roles & EC2 instance profiles for easy integration
Amazon Aurora	Proprietary tech from AWS (not open-sourced) Postgres and MySQL are supported as Aurora DB (drivers will work as if Aurora was a Postgres / MySQL DB) 'Cloud optimized': 5x performance improvement over MySQL on RDS, and 3x over Postgres Storage auto grow in increments of 10GB, up to 128TB Can have 15 replicas while MySQL have 5, & replication process is faster Failover is instantaneous. High availability (HA) native Cost more than RDS (20% more) – but is more efficient	High availability & Read scaling - 6 copies of data over 3 AZ: - 4 out of 6 needed for writes - 3 out of 6 needed for reads - self-healing w peer to peer replication - storage is striped across 100s of volumes - 1 aurora instance take writer (master) - auto failover for master in <30s - support for cross region replication
	Aurora DB Cluster Client -> writer endpoint (pointing to master) -> master instance that is not down -> data - auto-scaling for read replicas Replicas -> reader endpoint (connection load balancing) -> client	Auto failover, Backup & recovery, isolation & security, industry compliance, push-button scaling, auto patching w zero downtime, advanced monitoring, routine monitoring, backtrack (restore data at any point of time w/o backups)
	Security is similar to RDS: - encryption at rest using KMS - auto backup, snapshots and replicas are encrypted - encryption in flight using SSL (same as MySQL/Postgres)	- can authenticate using IAM token - can add security groups - cannot SSH
Aurora Extra	Replicas - Auto Scaling: When new replicas created, the reader endpoint will auto cover these new replicas Custom Endpoints: Define a subset of Aurora instances as custom endpoints (eg these subset are larger instance type) Reader endpoint usually not used when custom endpoints defined	Serverless: Auto DB instantiation & auto scaling. Good for infrequent, intermittent or unpredictable workload. No capacity planning needed. Pay per second
	Multi-Master: If u want immediate failover for writer node (HA)	Global Aurora: 1. aurora cross region read replicas (for DR), 2. aurora global DB : - 1 primary region (R/W) - Up to 5 secondary (R only) regions, replication lag < 1s

	Every node can read/write (R/W) vs single-master (have to promote R/R node to new master) Machine Learning: can connect to SageMaker or Comprehend & return results w SQL query	- Up to 16 read replicas per secondary region - Decr latency - Promoting another region (for DR) has a recovery time objective (RTO) < 1min
Elasti-Cache	Get managed Redis or Memcached Caches are in-memory DB w high performance and low-latency Help reduce load off DB for read-intensive workload Help make app stateless (session data stored in cache not in app, so user already logged in if redirected to another instance) AWS take care of OS maintenance/patching, optimizations, setup, config, monitoring, failure recovery & backup Involves heavy application code changes	App queries ElastiCache 1st, if not available, then get from RDS and store in ElastiCache Cache must have invalidation strategy to make sure only most curr data is stored
	Redis: - multi-AZ w auto-failover - read replicas to scale reads and high availability - data durability using append-only files (AOF) persistance - backup and restore features	Memcached: - multi-node for partitioning of data (sharding), - no HA (replication) - non-persistent, - no backup & restore - multi-threaded architecture
Elasti-Cache Extra	Security. All cache in ElastiCache: - do not support IAM authentication, - IAM policies only used for AWS API-level security Redis AUTH: - can set pw when u create Redis cluster - extra security on top of security groups - support SSL in flight encryption Memcached: - support SASL-based auth (advanced)	Patterns for ElastiCache: - Lazy Loading: all read data is cached, data can become stale in cache - Write Through: adds/update data in cache when written to a DB (no stale data) - Session Store: store temp session data (using time-to-live [TTL] features)
Extra	Redis Use Case: gaming leaderboards stored in ElastiCache Redis Sorted sets guarantee both uniquess and element ordering Ea time a new elem added, it's ranked in real time and added in correct order	RDS Databases ports: - PostgreSQL: 5432, - MySQL: 3306 - Oracle RDS: 1521, - MSSQL Server: 1433 - MariaDB: 3306 (same as MySQL) - Aurora: 5432 (PostgreSQL) or 3306 (MySQL)

DNS	Domain Name System translates human friendly hostnames into machine IP addresses (www.google.com -> 172.217.18.36) DNS uses hierarchical naming structure Terminologies: Domain registrar – Amazon route 53, GoDaddy, ... DNS records type – A, AAAA, CNAME, NS, ... Zone file – contain DNS records Top Level Domain (TLD) – .com, .us, .in, .gov, .org, ... Second level domain (SLD) – example.com, google.com Sub domain – www.example.com Domain name – api.www.example.com Protocol – http, Fully qualified domain name (FQDN) – http://api.www.example.com. (last dot is called the root)	How it works? E.g. web browser want access example.com -> local DNS server (LDNSS; managed by company/assigned by ISP dynamically) -> root DNS server (ICANN) - .com NS at IP 1.2.3.4 -> LDNSS -> TLD DNS server (IANA) - example.com NS at IP 5.6.7.8 -> local DNS server -> SLD DNS server (Domain registrar) - example.com IP 9.1.2.3 -> LDNSS -> web browser -> example.com at IP 9.1.2.3
Route 53	Highly available, scalable, fully managed & Authoritative DNS Authoritative = customer (you) can update DNS records Route 53 also a Domain Registrar Ability to check health of resources Only AWS service w 100% availability SLA (service level agreements), is also Multi-AZ by default	DNS records: define how to route traffic for a domain - contains: domain/subdomain name, record type, value (12.34.56.78), routing policy (how route 53 respond to queries), TTL (time record is cache at DNS resolvers)
	A – maps hostname to IPv4 AAAA - maps hostname to IPv6 CNAME - maps hostname to another hostname: target is a domain name w A/AAAA record, can't create a CNAME for the top node of a DNS namespace (Zone Apex) (cannot create for example.com but can for www.example.com)	Hosted Zones (HZ) – container for records that define how to route traffic to domain/subdomains Public Hosted Zones – record for routing traffic on Internet (public domain names)

	Name server (NS) for Hosted Zones– resolve DNS queries (authorative/non-authorative) $0.50 per month per Hosted Zone	Private Hosted Zones – record for routing traffic within ≥ 1 VPC (private domain names)
Record TTL	When client send DNS query to route 53, route 53 will send back a NS w IP & TTL TTL tells client to cache this result for duration of TTL, so that won't have to query route 53 again if same request is made - Except for Alias records, TTL is mandatory for DNS requests	High TTL (24hr) – less traffic to route 53 – possibly outdated records in cache Low TTL (60s) – more traffic (more $$) – use if u change IP, then can update records in cache of client
CNAME vs Alias	AWS resources (LB, CloudFront...) expose a AWS hostname: e.g. lb-1-1234.us-east-2.elb.amazonaws.com CNAME: - points hostname to any other hostname (app.domain.com -> abcde.anything.com) - Only for non-root domain (something.domain.com) Alias: - points hostname to a AWS Resource (app.domain.com -> abcde.amazonaws.com) - Works for root domain/Zone Apex (domain.com) and non-root domain - Record type always A/AAAA, cannot set TTL, Free, and have native health check Alias record targets: ELB, CloudFront Distributions, API Gateway, Elastic Beanstalk Environment, S3 Websites, VPC Interface Endpoints, Global Accelerator accelerator, Route 53 record in the same HZ Cannot set Alias for EC2 DNS name	
Routing Policies (RP) - Simple	Define how route 53 responds to DNS queries Diff from the routing of traffic in ELB Types of routing policies: Simple, Weighted, Failover, Latency based, Geolocation, Multi-Value Answer, Geoproximity (using route 53 Traffic Flow feature)	Simple: - typically route traffic to single resource - can specify multiple values (IP) in same record - if multiple values are returned, client randomly choose 1 - Using Alias, can only specify 1 AWS resource - Cannot use health checks
RP - Weighted	Control % of requests that go to ea resources Assign ea record a relative weight: traffic (%) = weight for specific record/sum of weights for all records (weights dont need to add up to 100) DNS record must have same name and type, Can be used w health checks Used for: LB btw regions, testing new app versions, Assign weight = 0 to stop sending traffic there If all records have weight 0, then equal distribution of requests to resources	
RP - Latency based	Redirect request to resource that has least latency. Latency is based on traffic btw users and AWS region. Can use with health checks (has failover capabilities)	
RP - Health Checks	HTTP health check are only for public resources. Health check are integrated with CW metrics Health check -> automated DNS failover: 1. Health check monitor endpoint (app, server, aws resource) 2. Calculated health checks (monitors other health checks) 3. Monitor Cloudwatch alarms (helpful for private resource)	
	1. About 15 global aws health checkers will check endpoint health: - healthy threshold (default 3), - interval (default 30s), - Supported protocol (HTTP, HTTPS, TCP), - If >18% of health checkers report healthy, route 53 will deem endpoint healthy, - can choose which location to use Health check pass only when endpoint responds w 2xx and 3xx status codes Health checks can be set up to pass/fail based on the 1st 5120 bytes of the response Need configure router/firewall to allow Route 53 health checks to access endpoints	
	2. Calculated health checks: - combine multiple health checks into a single health check, - can use OR AND NOT, - can monitor up to 256 children health checks, - specify how many children health check to pass to make parent pass Use for maintenance of website w/o causing all health checks to fail	
	3. Monitor Cloudwatch alarms: - to health check private endpoints (private VPCs, on-premise resource), - create Cloudwatch metric, associate Cloudwatch alarm, then create health checker that checks alarm	
RP - Failover	Route 53 to primary instance has health check (mandatory) If unhealthy, auto failover to secondary instance (can put health check for 2nd instance also) Only can have 1 primary and 1 secondary instance active-passive failover Client will get DNS record of instance that is healthy from route 53 (primary 1st, then secondary)	
RP - Geolocation	Routing based on user location. Can be associated w health checks Specify location by Continent, Country, or by US state (if overlapping, most precise location selected) Should create a "Default" record, in case no match on location Used for website localization, restrict content distribution, LB...	

RP - Geoproximity	Routing based on geographic location of users and resources. Use Route 53 Traffic Flow (advanced) Ability to shift more traffic to resources based on the defined bias To change the size of the geographic region, specify bias values: - to expand (1 to 99): more traffic to resource, to shrink (-1 to -99): less traffic AWS resource (specify AWS region), Non-AWS resource (specify latitude & longitude)
RP - Multi-Value	Use when routing traffic to multiple resources. It is NOT a substitute for ELB ("client-side" LB) Can be associated w health checks (return only values for healthy resources) Up to 8 healthy records returned for ea Multi-Value query
Domain Registrar vs DNS Service	U buy or register domain name w a Domain Registrar by paying annual fee (GoDaddy, Amazon Registrar Inc...). Domain Registrar provide a DNS service to manage your DNS records Can use GoDaddy w Route 53 (just change the NS on GoDaddy to the one provided when creating public HZ on Route 53). Alternatively, use Amazon Registrar w another DNS service (not Route 53)

Elastic Beanstalk	Mix of Golden AMI & Bootstrap w User Data. Developer centric view of deploying an app on AWS Managed service: auto handle capcity provisioning, LB, scaling, health monitoring, instance config,... Beanstalk is free, just pay for instances App: collection of Beanstalk components (environments, versions, config,...), - environment: AWS resources, Tiers (Web server environment tier, worker environment tier)	
	Web server tier: client can access (ELB, ASG, Multi-AZ)	Worker tier: client cannot access Web server tier will push messages to SQS queue EC2 instances in worker tier pull message from SQS queue and work from there Scale based on num of SQS messages

S3	Store objects (files) in buckets (directories) Bucket must have globally unique name. Defined at regional level Naming Convention: no uppercase, no underscore, 3-63 chars long, not an IP, start w lowercase letter/num	Objects have a key (**full path** = *prefix* + object name) s3://bucket/**file.txt** or s3://bucket/***folder1/folder2/*file.txt**
	Object value = contents of body: - max obj size = 5TB, - if uploading > 5GB (must use 'multi-part' upload, reccommended to use when obj > 100 MB) Metadata: list of text key/value pairs – sys or user metadata Tags: unicode key/value pairs – up to 10, for security/lifecycle. Version ID: if versioning enabled	
Versioning	Enabled at bucket level. Same key overwrite will increment the version Best practice to version bucket: - prevent unintended deletes, - easy rollback to previous version Any file not versioned prior to enabling versioning will have version null Suspending versioning does not delete previous versions	
Encryption	1. SSE-S3: encrypts objs using keys handled & managed by S3 - obj encrypted server side (i.e. on AWS), - AES-256 encryption type, - Must set header: "x-amz-server-side-encryption":"AES256" 2. SSE-KMS: use AWS Key Management Service to manage encryption keys - KMS allows user control + audit trail, - obj encrypted server side - Must set header: "x-amz-server-side-encryption":"aws:kms" 3. SSE-C: u manage own keys outside of AWS - server side encryption, - S3 does not store the encryption key, - HTTPS must be used - encryption key must be provided in HTTP headers, for every HTTP request made 4. Client Side encryption - encrypt before uploading to S3, - eg Amazon S3 Encryption Client - client decrypt data themselves when retrieving from S3	
Security	User-based: - IAM policies (which API calls are allowed for a specific user from IAM console) Resource-based: - Bucket policies (bucket wide rules from S3 console, allow cross account), - Object Access Control List (ACL), - Bucket ACL IAM user can access S3 obj if (user IAM permissions allow it OR resource policy allow it) AND (there's no explitict deny)	
Security - Bucket policies	JSON based policies: - Resources (buckets / objs), - Action (set of API to allow/deny), Effect (allow/deny), Principal (account/user to apply policy to) Used to grant public access to bucket, force objs to be encrypted on upload, grant access to another acct (Cross Acct)	

Security - Bucket Settings	Block Public Access to buckets and objs granted through - new ACL, any ACL, new public bucket or access point policies Block public and cross-acct access to buckets and objs through any public bucket or access point policies
Security - Other	Networking: - support VPC endpoint (EC2 in VPC w/o internet access) Logging & Audit: - S3 access logs stored in another S3 bucket, - API calls logged into CloudTrail User security: - MFA delete (MFA required to delete objs in versioned buckets), - pre-signed URL (URL that are only valid for a limited time, contain credentials from AWS)
Website	S3 can host static website and have them accessible on www URL would be <bucket-name>.s3-website-<AWS-region>.amazonaws.com or <bucket-name>.s3-website.<AWS-region>.amazonaws.com If 403 error, ensure bucket policy allows public reads
CORS	Cross-Origin Resource Sharing. Origin is a scheme (protocol), host (domain) and port. (e.g. origin https://www.example.com, w scheme = HTTPS, host = www.example.com, port = 443) Web Browser based mechanism to allow requests to other origins while visiting the main origin Diff origin (http://www.example.com & http://other.example.com). Same origin (http://www.example.com/app1 & http://www.example.com/app2) Request won't be fulfilled unless the other origin allows for the request, using CORS headers (ex Access-Control-Allow-Origin) Preflight request (browser ask cross-origin if allowed to make requests to it from origin), preflight response (cross-origin tells what requests are allowed, eg GET PUT DELETE)
S3 CORS	If client does cross-origin request on our S3 bucket (enabled as website), need to enable the correct CORS headers. Can allow for specific origin or * (all origin)
Consist-ency Model	S3 now has strong consistency. No performance impact & free After a new PUT / overwrite PUT (overwrite of existing obj) / DELETE, any subsequent read request will immediately return latest version of obj (read after write consistency & list consistency)

Extra	Create policies: Visual editor from AWS console or Policy Generator Simulate policies: Search AWS policy simulator on google EC2 Instance Metadata: - allows EC2 instance to 'learn about themselves' w/o using an IAM role - URL is http://169.254.159.254/latest/meta-data, - Can retrieve IAM role name but not IAM policy from meta-data SDK: CLI w python 3 uses boto3. If don't specify region or config default region, us-east-1 will be default

S3 MFA-Delete	To use MFA-Delete, must enable versioning on S3 bucket MFA needed when: permanently deleting object & suspend versioning on bucket Only bucket owner (root acct) can enable/disable MFA-Delete. Can only be enabled through CLI for now
Forced encryption	Instead of setting default encryption on bucket, can configure bucket policy to deny any API calls to PUT an S3 obj w/o encryption headers. Bucket policies are evaluated before default encryption
S3 Access Logs	Server Access Logging: Any access made to S3, from any acct, authorized/denied will be logged into another S3 bucket for audit purposes
CRR & SRR	Cross region replication & Same region replication. Must enable versioning in source and destination. Bucket can be from diff acct. Copying is asynchronous. Must give proper IAM permissions to S3 Use CRR for compliance, lower latency access, replication across acct Use SRR for log aggregation, live replication btw production and test accts After activating, only new objs are replicated. To replicate existing objs/objs that failed replication, must use S3 Batch Replication For DELETE: - can replicate delete markers (optional), - deletions w version ID are not replicated (prevent malicious deletes that permanently delete objs) No chaining of replications (i.e. bucket 1 replicated to 2, 2 to 3. Objs in 1 not replicated in 3)
Pre-signed URLs	Generate pre-signed urls using SDK or CLI. Valid for 3600s by default, can change w --expires-in <time> arg Users given a pre-signed URL will inherit the permissions of the person that generate the URL for GET/PUT Use cases: allow only logged in users to download premium video from S3 bucket, allow ever-changing list of users to download files by generating URLs dynamically, temporarily allow user to upload file to specific location in bucket

Storage Classes	Standard - General Purpose, Standard-Infrequent Access (IA), One Zone-Infrequent Access, Glacial Instant Retrieval, Glacial Flexible Retrieval, Glacial Deep Archive, Intelligent Tiering Objects can move btw classes manually or by lifecycle configs	High durability (99.999999999% 11 9s) of objs across multiple AZs (i.e. very low chance of losing objects) Durability same for all Storage class Availability: how readily available a service is. Varies depending on class
Standard	General Purpose: 99.99% availability (i.e. not available for 53 mins a year). For frequently accessed data. Low latency & high throughput. Can sustain 2 concurrent facility failures. Use cases: big data analytics, mobile & gaming apps, content dist,...	IA: 99.9% availability. For data that is less frequently access, but requires rapid access when needed. Lower cost than S3 Standard. Use cases: DR, backups,... One Zone-IA: 99.5% availability. Data lost when AZ destroyed. Use cases: storing secondary backups, data u can recreate easily,...
Glacial	Low cost object storage for archival/backup. Pay for storage & retrieval cost Instant Retrieval: millisecond retrieval, good for data accessed once every quarter. Min storage duration of 90 days	Flexible Retrieval: Expedited (1-5mins), Standard (3-5hrs), Bulk (5-12hrs, free). Min storage duration of 90 days Deep Archive: long term storage. Standard (12hrs), Bulk (48hrs). Min storage duration of 180 days
Intelligent-Tiering	Small monthly monitoring & auto-tiering fee Auto move objs btw Access Tiers based on usage. No retrieval cost Frequent Access Tier (auto): default. IA Tier (auto): objs not accessed for 30 days Archive Instant Access Tier (auto): objs not accessed for 90 days. Archive Access Tier (optional): configurable from 90 days to 700+ days. Deep Archive Access Tier (optional): configurable from 180 days to 700+ days	
Lifecycle Rules	Transition actions: define when objs are transitioned to another storage class Expiration actions: config objs to delete after some time (can be used to delete old versions of objs/incomplete multi-part uploads) Can create rule for certain prefix or obj tags	
S3 Analytics	Help determine whento transition objs from Standard to Standard-IA Does not work for One-Zone or Glacial Report is updated daily. Takes abt 24-48hrs to 1st start. Good 1st step to figure out lifecycle rules	
Baseline Perform-ance	Auto scales to high request rates, latency of 100-200ms > 3500 PUT/COPY/POST/DELETE and 5500 GET/HEAD requests per sec per prefix in a bucket	
	If using SSE-KMS, may be affected by KMS limits On upload, it calls the GenerateDataKey KMS API. On download, calls Decrypt KMS API Count towards the KMS quota per sec (5500, 10000, 30000 req/s based on region) Can request quota incr using the Service Quota Console	
	Multi-Part upload: parallelize upload (speed up transfers) S3 Byte-Range Fetches: Parallize GETs by requesting specific byte ranges. Better resilience in case of failures. Speed up downloads (eg used to retrieve head of the file)	S3 Transfer Acceleration: incr transfer speed by transferring file to an AWS edge location which will forward data to the S3 bucket in the target region Compatible w multi-part upload
S3 Select	Retrieve less data using SQL by performing server side filtering Can filter by rows or columns (simple SQL statements). Less network transfer, less CPU cost client-side	
Event Notifica-tions	Notified when some obj op is done. Can filter by obj name. Can send to SNS, SQS, Lambda, EventBridge Use cases: generate thumbnail of images uploaded to S3 Typically deliver events in seconds but can take up to mins or longer All events will end up in EventBridge, from there can send to 18 AWS services according to rules EventBridge allows advanced filtering w JSON rules (metadata, size, name, ...)	
Requester Pays	In general, bucket owner pays for all S3 storage and data transfer cost associated w the bucket W requester pays bucket, requester pays cost of request and data download from bucket Helpful when u want to share large datasets w other accts Requester must be authenticated w AWS (cannot be anonymous)	
Athena	Serverless query service to perform analytics on S3 objs. Use compressed/columnar data for cost-savings Uses SQL to query the files. Support CSV, JSON, ORC, Avro & Parquet Use cases: BI/analytics/reporting, analyze & query VPC flow logs, ELB logs, CloudTrail trails,...	

GlacialVault Lock	Adopt a WORM (write once, read many) model. Lock the policy for future edits Helpful for compliance and data retention
S3 Object Lock	Must enable versioning. Adopt WORM model. Block obj version deletion for a specified amt of time Obj retention: - Retention period (specifies fixed period), - Legal Hold (same protection, no expiry) Modes: - Governance mode (users can't overwrite/delete obj version or alter its lock settings unless w special permissions), - Compliance mode (period can't be shortened, any nobody can change anything)

Amazon RDS creates an SSL certificate and installs the certificate on the DB instance when Amazon RDS provisions the instance. These certificates are signed by a certificate authority. The SSL certificate includes the DB instance endpoint as the Common Name (CN) for the SSL certificate to guard against spoofing attacks.
You can download a root certificate from AWS that works for all Regions or you can download Region-specific intermediate certificates.

To make the application instances accessible on the internet the Solutions Architect needs to place them behind an internet-facing Elastic Load Balancer. The way you add instances in private subnets to a public facing ELB is to add public subnets in the same AZs as the private subnets to the ELB. You can then add the instances and to the ELB and they will become targets for load balancing.

EC2 instances behind an Elastic Load Balancer. All data in transit must be encrypted.
You can passthrough encrypted traffic with an NLB and terminate the SSL on the EC2 instances, so this is a valid answer.
You can use a HTTPS listener with an ALB and install certificates on both the ALB and EC2 instances. This does not use passthrough, instead it will terminate the first SSL connection on the ALB and then re-encrypt the traffic and connect to the EC2 instances.

Use CloudWatch Container Insights to collect, aggregate, and summarize metrics and logs from your containerized applications and microservices. Container Insights is available for ECS, EKS, and Kubernetes platforms on Amazon EC2.
With Container Insights for EKS you can see the top contributors by memory or CPU, or the most recently active resources. This is available when you select any of the following dashboards in the drop-down box near the top of the page:
• ECS Services • ECS Tasks • EKS Namespaces • EKS Services • EKS Pods

You can put an instance that is in the InService state into the Standby state, update some software or troubleshoot the instance, and then return the instance to service. Instances that are on standby are still part of the Auto Scaling group, but they do not actively handle application traffic.
Suspend the ReplaceUnhealthy process type for the Auto Scaling group and apply the maintenance patch to the instance. Once the instance is ready, you can manually set the instance's health status back to healthy and activate the ReplaceUnhealthy process type again

You can create an Amazon CloudWatch alarm to automatically recover the Amazon EC2 instance if it becomes impaired. Terminated instances cannot be recovered. A recovered instance is identical to the original instance, including the instance ID, private IP addresses, Elastic IP addresses, and all instance metadata.
If the impaired instance is in a placement group, the recovered instance runs in the placement group.
If your instance has a public IPv4 address, it retains the public IPv4 address after recovery.
During instance recovery, the instance is migrated during an instance reboot, and any data that is in-memory is lost.

AMI: - can copy within or across AWS Regions using the AWS Management Console, the AWS Command Line Interface or SDKs, or the Amazon EC2 API, all of which support the CopyImage action
- can share an AMI with another AWS account. To copy an AMI that was shared with you from another account, the owner of the source AMI must grant you read permissions for the storage that backs the AMI, either the associated EBS snapshot (for an Amazon EBS-backed AMI) or an associated S3 bucket (for an instance store-backed AMI).

Network Load Balancers expose a fixed IP to the public web, therefore allowing your application to be predictably reached using these IPs, while allowing you to scale your application behind the Network Load Balancer using an ASG.

Trust policies define which principal entities (accounts, users, roles, and federated users) can assume the role. An IAM role is both an identity and a resource that supports resource-based policies. For this reason, you must attach both a trust policy and an identity-based policy to an IAM role. The IAM service supports only one type of resource-based policy called a role trust policy, which is attached to an IAM role.

ASG
If you have an EC2 Auto Scaling group (ASG) with running instances and you choose to delete the ASG, the instances will be terminated and the ASG will be deleted
EC2 Auto Scaling groups can span Availability Zones, but not AWS regions
Data is not automatically copied from existing instances to a new dynamically created instance

An IAM user with full administrator access can perform almost all AWS tasks except a few tasks designated only for the root account user. Some of the AWS tasks that only a root account user can do are as follows: change account name or root password or root email address, change AWS support plan, close AWS account, enable MFA on S3 bucket delete, create Cloudfront key pair, register for GovCloud.

You can authenticate to your DB instance using AWS IAM database authentication. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token. An authentication token is a unique string of characters that Amazon RDS generates on request. Each token has a lifetime of 15 minutes. You don't need to store user credentials in the DB, because authentication is managed externally using IAM. IAM database authentication works with MySQL and PostgreSQL engines for Aurora as well as MySQL, MariaDB and RDS PostgreSQL engines for RDS.

You can create a read replica as a Multi-AZ DB instance. Amazon RDS creates a standby of your replica in another Availability Zone for failover support for the replica. Creating your read replica as a Multi-AZ DB instance is independent of whether the source database is a Multi-AZ DB instance.

- All EBS types support encryption and all instance families now support encryption.
- Not all instance types support encryption.
- Data in transit between an instance and an encrypted volume is also encrypted (data is encrypted in trans.
- You can have encrypted and unencrypted EBS volumes attached to an instance at the same time.
- Snapshots of encrypted volumes are encrypted automatically.
- EBS volumes restored from encrypted snapshots are encrypted automatically.
- EBS volumes created from encrypted snapshots are also encrypted.

To enable your Lambda function to access resources inside your private VPC, you must provide additional VPC-specific configuration information that includes VPC subnet IDs and security group IDs

Glacial auto provides encryption at rest and in transit