AWS ALL SERVICES

COMPUTE

- Amazon EC2 (Elastic Compute Cloud): Provides resizable virtual servers in the cloud, ideal for customizable compute capacity for web servers, application servers, and general-purpose needs.
- Amazon EC2 Auto Scaling: Automatically adjusts EC2 capacity to maintain performance and availability based on predefined conditions, ideal for handling variable workloads and ensuring cost efficiency.
- AWS Lambda: A serverless compute service that runs code in response to events, automatically managing compute resources, suitable for microservices, real-time file processing, and event-driven applications.
- AWS Fargate: A serverless compute engine for containers that works with Amazon ECS and EKS, eliminating the need to manage servers, ideal for deploying microservices and containerized applications.
- Amazon ECS (Elastic Container Service): A fully managed container orchestration service for running Docker containers, suitable for large-scale containerized applications.
- Amazon EKS (Elastic Kubernetes Service): A managed service for running Kubernetes on AWS, providing highly available and secure clusters, suitable for managing containerized applications with Kubernetes.
- Amazon ECR (Elastic Container Registry): A managed Docker container registry that makes it easy to store, manage, and deploy container images, ideal for CI/CD pipelines and containerized application deployments.

DATABASE

- Amazon Aurora: A high-performance, MySQL- and PostgreSQL-compatible relational database, suitable for enterprise applications requiring high availability and scalability.
- Amazon DocumentDB: A managed NoSQL document database service that is MongoDB-compatible, ideal for applications needing flexible, JSON-like document storage.
- Amazon DynamoDB: A managed NoSQL database service providing fast and predictable performance with seamless scalability, suitable for web, mobile, gaming, and IoT applications requiring low-latency data access.
- Amazon ElastiCache: A managed in-memory caching service compatible with Redis and Memcached, ideal for accelerating application performance by caching frequently accessed data.
- Amazon Keyspaces: A scalable, highly available, and managed Apache Cassandra-compatible database service, suitable for IoT and time-series data.
- Amazon MemoryDB: A Redis-compatible, in-memory database service optimized for applications requiring microsecond read and write speeds.
- Amazon Neptune: A managed graph database service for building and running applications with highly connected datasets, suitable for social networks, recommendation engines, and fraud detection.
- Amazon Quantum Ledger Database (QLDB): A managed ledger database providing a transparent, immutable, and cryptographically verifiable transaction log, ideal for financial systems and applications requiring a verifiable transaction log.
- Amazon RDS (Relational Database Service): A managed service for relational databases supporting MySQL, PostgreSQL, Oracle, and SQL Server, suitable for traditional business applications.
- Amazon Redshift: A fast, scalable data warehouse service for large-scale data analysis using SQL and existing BI tools, ideal for business intelligence and complex query execution.
- Amazon Timestream: A fully managed time series database optimized for fast, scalable storage and analysis of time series data, suitable for IoT applications, operational monitoring, and real-time analytics.

STORAGE AND BACKUP

- Amazon EBS (Elastic Block Store): Provides persistent block storage volumes for use with Amazon EC2 instances, suitable for databases, file systems, and applications requiring persistent storage.
- Amazon EFS (Elastic File System): A scalable, fully managed NFS file system for use with AWS Cloud services and on-premises resources, suitable for shared access across multiple EC2 instances.
- Amazon FSx: Provides fully managed third-party file systems (e.g., Windows File Server, Lustre), ideal for specialized file storage needs.
- Amazon S3 (Simple Storage Service): An object storage service offering scalability, data availability, security, and performance, suitable for storing and retrieving any amount of data, including backups, archives, and multimedia files.
- Amazon S3 Glacier: A secure, durable, and low-cost cloud storage service for data archiving and long-term backup, ideal for infrequently accessed data.
- AWS Elastic Disaster Recovery: Helps minimize downtime and data loss with fast, reliable recovery of on-premises and cloud-based applications.

NETWORK AND CONTENT DELIVERY

- Amazon CloudFront: A fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs globally with low latency, suitable for web content, videos, and APIs.
- Amazon Route 53: A scalable domain name system (DNS) web service designed to route end users to internet applications with high availability.
- Amazon VPC (Virtual Private Cloud): Allows launching AWS resources in a logically isolated virtual network, suitable for running AWS resources securely.
- AWS Direct Connect: Establishes a dedicated network connection from your premises to AWS, ideal for high bandwidth and low latency requirements.
- AWS Transit Gateway: Connects Amazon VPCs and on-premises networks through a single gateway, suitable for large-scale network architectures.
- AWS VPN: Establishes a secure and private tunnel from your network or device to the AWS global network, suitable for secure and encrypted data transmission.
- AWS Network Firewall: Deploys essential network protections for Amazon VPCs, enhancing security and compliance.
- AWS WAF (Web Application Firewall): Protects web applications from common web exploits, improving security of internet-facing applications.
- AWS Firewall Manager: Centrally configures and manages firewall rules across your AWS accounts and applications, suitable for managing AWS WAF rules across multiple accounts and applications.

SECURITY AND IDENTITY

- AWS IAM (Identity and Access Management): Helps securely control access to AWS services and resources, ensuring secure access control.
- AWS KMS (Key Management Service): Manages encryption keys used to encrypt your data, securing data with encryption.
- Amazon Detective: Analyzes, investigates, and identifies the root cause of potential security issues or suspicious activities.
- Amazon Macie: Uses machine learning and pattern matching to discover and protect your sensitive data in AWS, suitable for data privacy and compliance.
- Amazon GuardDuty: Continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads.
- Amazon Inspector: Provides automated security assessments to improve the security and compliance of applications deployed on AWS.
- AWS Security Hub: Aggregates, organizes, and prioritizes security alerts from various AWS services, providing a comprehensive view of your security posture, suitable for managing and monitoring security across AWS accounts.

MANAGEMENT AND GOVERNENCE (MONITORING)

- AWS CLI (Command Line Interface): Manages AWS services using commands in your shell, ideal for automating workflows and managing resources through scripts.
- AWS X-Ray: Helps developers analyze and debug distributed applications, suitable for tracing requests and diagnosing performance issues.
- Amazon CloudWatch: Monitors AWS cloud resources and applications, suitable for performance monitoring, alerting, and log analytics.
- Amazon CloudFormation: Models and provisions AWS and third-party application resources, suitable for automating resource deployment and management.
- Amazon CloudTrail: Enables governance, compliance, and operational auditing of your AWS account, ideal for security auditing, compliance, and operational troubleshooting.
- AWS Config: Provides AWS resource inventory, configuration history, and configuration change notifications, suitable for compliance and governance.
- AWS Trusted Advisor: Optimizes AWS resources, providing recommendations on cost, performance, security, and fault tolerance.
- AWS License Manager: Manages software licenses from vendors such as Microsoft, SAP, Oracle, and IBM across AWS and on-premises environments, suitable for tracking and optimizing license usage.

APPLICATION INTEGRATION

- Amazon SES (Simple Email Service): A cost-effective, flexible, and scalable email service for sending bulk and transactional emails, suitable for marketing, notifications, and email communication.
- Amazon API Gateway: Creates, publishes, maintains, monitors, and secures APIs at any scale, suitable for backend services and applications needing RESTful APIs.
- Amazon Amplify: Enables front-end web and mobile developers to build, ship, and host full-stack applications.
- Amazon GameLift: Deploys, operates, and scales dedicated game servers for session-based multiplayer games.

ANALYTICS

- Amazon Athena: An interactive query service using SQL to analyze data directly in S3, suitable for log analysis and business intelligence.
- Amazon QuickSight: Creates and shares interactive dashboards and visualizations from various data sources, ideal for business intelligence and reporting.
- Amazon Q in QuickSight: A natural language query tool within QuickSight enabling users to ask questions and receive visual answers quickly.
- AWS Glue: A fully managed ETL service preparing and loading data for analytics, supporting data cataloging, data transformation, and data integration.

STREAMING DATA AND EVENT PROCESSING

- Amazon Kinesis: Collects, processes, and analyzes real-time streaming data, suitable for log and event data collection, real-time analytics, and IoT data.
- Amazon EventBridge: Integrates applications using events from AWS services, SaaS applications, and custom applications, suitable for real-time application integration and automation.
- Amazon SNS (Simple Notification Service): Sends notifications and messages to distributed systems, ideal for alerting, push notifications, and message broadcasting.
- Amazon SQS (Simple Queue Service): Decouples and scales microservices and distributed systems, suitable for task queues and message buffering between application components.

DEVELOPER TOOLS (CI/CD)

- Amazon CodeCatalyst: An integrated DevOps toolchain for building, testing, and deploying applications on AWS.
- AWS CodeArtifact: Manages software packages and dependencies, ideal for storing and sharing artifacts across development teams.
- AWS CodeBuild: Compiles source code, runs tests, and produces software packages, suitable for continuous integration and build automation.
- AWS CodeCommit: Hosts secure Git repositories, suitable for version control and collaborative software development.
- AWS CodePipeline: Automates release pipelines, suitable for continuous delivery and deployment of applications and infrastructure.
- AWS CodeDeploy: Automates code deployments to any instance, including EC2 and on-premises servers.

AI AND MACHINE LEARNING

- Amazon Bedrock: Builds and scales generative AI applications with foundational models.
- Amazon CodeGuru: Provides intelligent recommendations to improve code quality and identify issues.
- Amazon SageMaker: Builds, trains, and deploys machine learning models quickly, suitable for data scientists and developers working on ML projects.
- Amazon CodeWhisperer: An AI-powered code completion tool helping developers write code faster and with fewer errors.

AWS GENERAL

Adv of Cloud → 1. create and operate quickly and instantly, 2. Pay on demand, 3. reliable and secure global infrastructure, 4. Scaled based on needs, 5. innovation: focus on business differentiators.
Why people choose the cloud? → Most functionality | Largest community of customers and partners | Most Secure | Fastest pace of innovation | Most proven operational expertise
Shared Responsibility → Customer responsible for security in the cloud (data, platform, application, firewall config, network traffic protection) as well as operating system updates. / AWS responsible for the security of the cloud (compute, storage, regions, AZs, database, networking) ||| Choosing cloud vendor → global infrastructure (What is the underlying architecture and design of this to enable High Availability and synchronous replication?), security (How many third-party attestations and security features do the vendors have?), innovation (Is the vendor investing for the long term and have demonstrated capability with reference customers taking advantage of innovative services?), cost (philosophy of company, Have you evaluated price to features across vendors?)
AWS Global Infrastructure → Region (A physical location where we cluster data centers. Each Region consists of a minimum of three isolated and physically separate AZs.AWS Regions are comprised of multiple AZs for high availability, high scalability, and high fault tolerance. Applications and data are replicated in real time and consistent in the different AZs.), AZ (One or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. Each AZs in a AWS Region are physically separated by a 100km), Local Zones (place compute, storage, database, and other AWS services closer to end-users so you can run latency-sensitive applications and meet data residency requirements in more locations), Points of Presence (Reduce latency by delivering data through 550+ globally dispersed Points of Presence (PoPs) with automated network mapping and intelligent routing using Amazon CloudFront.), DX (The Direct Connect service establishes a private, physical network connection between AWS and your data center or office environment – bypassing the public internet.), Outposts(AWS Outposts are a family of fully managed solutions delivering AWS infrastructure and services to virtually any on-premises or edge location for a truly consistent hybrid experience.).
Amazon CloudFront Infrastructure: Edge locations (A network of data centers that CloudFront uses to quickly deliver popular content to customers), Regional EdgeCache (A CloudFront location that caches content that is not popular enough to remain in an edge location. It is located between the origin server and the global edge location.)
Pillars of AWS Well-Architected: Operational Excellence | Security | Reliability | Performance Efficiency | Cost Optimization | Sustainability

AMAZON EC2 OVERVIEW

Region → AZs (min:1, max:6 | 100km sep) → data center ||| Local Zones → limited service, keep data in country ||| AZ → high throughput, low latency, traffic encrypted, provide all services ||| PoPs(Points of Presence): reduce latency by delivering data through dispersed PoPs with automated network mapping and intelligent routing using Amazon CloudFront.

- SaaS(all managed by vendor), PaaS, IaaS, On-Premises(you manage all) → Application(I, P), Data(I, P), Runtime(I), Middleware(I), OS, Virtualization, Servers, Storage, Networking
- Why are companies adopting containers? • Accelerate software development • Build modern applications • Automate operations at web scale
- Application environment components ⇒ Runtime Engine | Code | Configuration | Dependencies
- Docker: Lightweight container virtualization platform. Ecosystem of tools to manage and deploy your applications ||| Container →1.packages code, library, dependencies etc. 2. Linux primitives for isolation 3. share linux kernel 4. fast start, minimal overhead 5. flexible isolation ||| VMs → 1.Emulate hardware components 2. completely separate kernels (all OSs supported)) 3. slower start, must boot kernel and set-up hardware ||| Container images: 1. Read only images used as a template to launch a container 2. Start from base images that have your dependencies, add your custom code. 3. Dockerfile for easy, reproducible builds. ||| Container Benefits: Runs reliably everywhere | run different apps simultaneously | better resource utilization

AWS Compute Offerings: EC2(configure servers, storage, networking, and my OS)/ ECS(run servers, configure applications, and control scaling) / Fargate (run containers) / Lambda (run code) Choices for Compute: EC2 (Virtual server instances in the cloud) / ECS-EKS-Fargate (Container management on a managed cluster of EC2) / Lambda (serverless compute, stateless, trigger)
AWS instance naming → c7gn.xlarge(c: instance family, 7: instance generation, gn: attribute(s), xlarge: instance size) ||| c7gn.8xlarge = 2 - c7gn.4xlarge = 4 - c7gn.2xlarge = 8 - c7gn.xlarge(this one cheaper)
Instance Types: 1.General Purpose(T, M) 2.Compute Opt(C) 3.Memory Opt(R, X, X2, Z1) 4.Accelerated Computing(G,P,F) 5.Storage Opt(I, D, H)
vCPU(virtual CPU): 1) hyper-threaded physical core (CPU Optimizing options allow disabling hyperthreading and reduce number of cores) 2) Divide vCPU count by 2 to get core count. 3) On LInux, "A" threads are enumerated before "B" threads. 4) On Windows, threads are interleaved.

- +650 instance type = Categories[General purpose, Burstable, Compute intensive, Memory intensive, Storage (High I/O), Dense storage, GPU compute, Graphics intensive] + Capabilities [Choice of processor (AWS Graviton, Intel, AMD), Fast processors (up to 4.5 GHz), High memory footprint (up to 24 TiB), Instance storage (HDD and SSD), Accelerated computing (GPUs ,FPGA & ASIC), Networking (up to 800 Gbps), Bare Metal, Size(Nano to 48xlarge)] + Options[Elastic Block Store (EBS), Elastic Fabric Adapter, Elastic Inference, Elastic Graphics, OS (Linux, Unix, Windows, macOS)]

GiB: Memory is presented as GibiBytes (GiB) and not Gigabytes (GB), 256 GiB = 275 GB.
EC2 Operating Systems: • Windows Server 2012/2012 R2/2016/2019/2022 • Amazon Linux (NEW: Amazon Linux 2023) • Debian • SUSE • CentOS • Red Hat Enterprise Linux (RHEL) • Ubuntu • Mac, including M1 Mac instances

- AMI(Amazon Machine Image): 1.info to launch an instance (single AMI → launch several inst with the same config) 2. AMI includes the following: • one or more EBS snapshots or a template for root volume(os, application), • launch permission that controls which accounts can use the AMI, • Block device mapping that specifies volumes to attach to the inst.
- EC2 Purchase Options: 1.On-Demand(no long-term commitment - useful for new or stateful spiky workloads) 2.Reserved Instances(RI)(1 or 3 year commitment, cheaper - useful for known/steady-state workloads) 3.Saving Plans(more flexible, great discount - useful for known/steady-state workloads) 4.Spot Instances(saving up to 90/%, flexible, fault-tolerant, stateless workloads - useful for test env. - spot instances are terminated automatically by AWS when the maximum price you specified exceeds. Spot instances are used for fault-tolerant, flexible, stateless workloads like batch processing, data analysis, HPC workloads or CI/CD pipelines)

Advantage of AWS Savings Plans over AWS Reserved Instances → Reduce the complexity: You don't need to consider which type of RI, size, family,…you should use before you commit. You also get rid of the complicated processes when you want to manage, convert or resell the RIs on the marketplace. ||| Save time for infrastructure planning: Your engineers no longer have to plan complex infrastructure for 1-3 years in advance, they can simply commit to using a certain amount of compute.||| Enhance flexibility: Firstly, Savings Plans unleash you from the burden of using only 1 specific instance for 1 to 3 years despite your changing requirements. Secondly, Savings Plans allow you to transfer workloads between instance types to meet demand and leverage something akin to elasticity without resulting in downtime. You cannot gain the same advantages when using RIs, which require you to commit 24/7/265 usage of a specific instance type.
EC2 Security Groups: 1) it is a virtual firewall 2) SG rules are name, description, Type(HTTP, SSH), protocol (TCP), port range(22), IP address, IP range
EC2-Specific Credentials:
1)EC2 key pairs (linux or windows) 2)Standard SSH RSA key pair(•Public/Private keys •private keys are not stored by AWS) 3)AWS approach for providing initial access to a generic OS (•Secure •Personalized •Non-generic)
!!! An EC2 is actually a VM.
Resource Allocation → All resources assigned to you are dedicated to your instance with no over commitment(•All vCPUs are dedicated to you. •Memory allocated is assigned only to your instance. •Network resources are partitioned to avoid "noisy neighbors")
Which hypervisor do we use? → Original host architecture: Xen-based (Hypervisor consumed resources from the underlying host. Limited optimization) ||| AWS Nitro Hypervisor: Custom KVM based hypervisor (AWS Nitro System. Less server resources used, more resources for the customer. AWS optimized) ||| Bare metal: Direct access to processor and memory resources (uilt on the AWS Nitro system. Enables custom hypervisors and micro-VM runtimes)
AWS Nitro System → Nitro Card (Local NVMe storage. Elastic Block Storage. Networking, monitoring, and security) ||| Nitro Security Chip (Integrated into motherboard. Protects hardware resources) ||| Nitro Hypervisor (Lightweight hypervisor Memory and CPU allocation. Bare metal-like performance)
Hibernation: When hibernation is initiated, the instance moves to the stopping state. Amazon EC2 signals the operating system to perform hibernation (suspend-to-disk). The hibernation freezes all of the processes, saves the contents of the RAM to the EBS root volume, and then performs a regular shutdown.

SERVERLESS ON AWS (W3-W4)

Serverless: no infrastructure management, no server, auto scaling, high availability and security, pay for value, write and deploy code w/o server management. Serverless → IT automation, data processing, web app, ML.
!!!!! Serverless functions are stateless which means that they don't maintain persistent connection or state between invocations. Any required state must be managed externally such as in a database or a distributed cache. Serverless Primitive Services: Lambda, Fargate, ECS, S3, EFS, API Gateway, EventBridge, SNS, App Runner SNS, SQS. Serverless Peripheral Services: DynamoDB, kinesis, Aurora Serverless, CloudWatch, AppSync, StepFunctions, Glue, OpenSearch, QuickSight
LAMBDA: Event driven func (like handler func) as a service. Event Source(changes in data state, request to endpoints, changes in resource state) ⇒ LAMBDA ⇒ Services/other. Some examples for event sources: changes in data state, request to endpoint, changes in resource state. Anatomy of a Lambda Function:Lambda function is like a handler function. Function executed on invocation. Processes incoming events. It gets events and context, and then does what you say and returns a message. Event: Invocation data sent to function. Event shape differs by event source. Context is additional information from lambda service like request ID, time remaining. Lambda function is like a really really small container. Max 15 min run. Pay for running time. *You can run in parallel (run the same lambda func with different params at the same time). Execution role grants function access to resources via IAM which means you need to give permission to call Lambda. Configure access to VPC. Security group rules apply.
Lambda Function Configuration: Power Rating(• Select between 128 MB and 10GB •CPU and network allocated property •Power tune to balance cost and speed) ||| Permission Model (•Execution role grants function access to resource via IAM •Resource-based policy controls invocation from other AWS accounts for organizations and AWS services) ||| Timeout (•Up to 15 min •Synchronous vs Asynchronous (API Gateway timeout is 30 sec)) ||| Network Access (•Configure access to VPC •Security Group rules apply •VPC doesn't enhance security of function)
!!! If you need to securely invoke Lambda functions from within your VPC without internet access, VPC Endpoints are the way to go. If you need to create public or private APIs that external clients can interact with while leveraging advanced API management features, then API Gateway is the appropriate choice. While it's technically possible to invoke Lambda functions directly, for example, from an SDK in a client application, doing so bypasses these benefits and exposes your Lambda functions directly to clients, which can lead to security vulnerabilities, scalability issues, and a tighter coupling between your client and server architecture.

- API-Driven Architecture(ADA): Immediate response. API defines the interface(REST, GraphQL). Clients must implement error handling, retry logic. You can lose some calls. (front end - middleware - backend ⇒ i can put/delete/update my db but it would be insecure, so I add middleware using API Gateway which communicate fe with private key)
- Event-Driven Approach (EDA): Store first, process after approach. Event payload defines the interface(JSON). Driven by messaging service. Response is like "message received" or not. Higher resilience, durability. Built-in retries, configure to use-case.
- Example: Client → EventBridge/EventBus → lambda → DB changes ⇒ in this example EventBridge or Event Bus allow us to do an event driven approach.
- Trends of EDA → speed & agility | resilience(loosely coupled system can fail independently) | scalability(minimize waiting time through and parallel processing).
- EDA Components: Producer → Event Handler/Broker ← Consumer. Producer: Detect a change in state or notice updating and publishing those facts. Event Broker: Meeting place between producer and consumer. It directs to the consumer. Consumer: Listen for events and use them for their purposes.
- Event Brokers in EDA: There are two types: Push-based [Routers(SNS, EventBridge)] and Pull-based[Queues(SQS, MQ), Streams(Kinesis Kafka/MSK)]. Push-Based Event Broker: actively pushes events to consumers as soon as producer publish available events. Suitable for real-time apps where low latency and immediate delivery needed. (AWS services push-based ⇒ Routers: EventBridge, SNS) Pull-Based Event Broker: Consumers actively pull events from the broker when it is ready. Consumers periodically pull events. (AWS services pull-based ⇒ Queues: SQS, MQ | Streams: Kinesis, Kafka/MSK)
- EDA on AWS: Amazon SQS (Messaging: Durable and scalable Fully managed Comprehensive security) ||| Amazon SNS (Notifications: Performance at scale. Fully managed. Enterprise-ready) ||| Amazon EventBridge (Choreography: Event filtering. Managed & scalable Saas integration) ||| AWS Step Functions (Orchestration: Sequencing. Parallel execution. State management.)
- Thinking Asynchronously: Store first and process after. API → Storage → Compute
- Asynchronous processing makes it easier to do more in less time

* Async allows us to 1.secure more data, 2.determine failures, 3.try failed requests
* If you need fast response then you need to use synchronous but you are ok with losing some calls.
ECR(Elastic Container Registry): Serverless. Managing many containers is hard, for that purpose ECR can be used. It is like a repo to store your image. You push your container image to ECR. *It automates compliance image scanning (check CVEs) Public repo → ECR public Gallery (sell/buy app)
ECR Properties → • Managed and scalable infrastructure •Highly available, high performance •Security with encrypted images and vulnerability scans •Authenticated access, centralized IAM control •Native integration to AWS orchestrators and compute •Docker and OCI compliant to pull anywhere •Container images, Helm charts, OCI artifacts

Docker helps us to launch containers and ensures your containers are isolated. ECS is kind of built on top of docker.

- Scheduling and Orchestration: Cluster Manager | Placement Engine (like tetris)

ECS (ELASTIC CONTAINER SERVICE): Managed service to run containers. Scheduling and Orchestration (Cluster Manager and Placement Engine) 1)Serverless with Fargate, 2)Managed service with EC2(if you need specific EC2 configs)→you manage the instances; -software upgrade, -patches, -security.
ECS→ supports autoscaling, for batch jobs and full services, multiple docker containers on EC2 instances, manages the lifecycle of containers(create/restart/destroy), load balance & deploy the app across multiple servers, autoscale
- If you don't want to use ECS, you will have to manage everything yourself
- ECS is the brain ||| Cluster is a bunch of resources (cluster contains several container instances that can be either ec2 or fargate. Container instances include several containers that runs tasks)
- Container Instance can be either EC2 or Fargate. Container instance contains containers.
- Solely ECS doesn't have compute power it needs Fargate or EC2
- ECS manages the containers

EC2 Launch Type of ECS: You have to manage the underlying infrastructure "the EC2's" ||| You add the EC2 ||| You install docker ||| You install ECS agent (EC2 and ECS communication) ||| You manage the servers yourself, upgrade/maintain etc.
Fargate Launch Type of ECS: Serverless architecture ||| You don't see the EC2 instances ||| Fargate create server on demand

- ECS Task Definition: • Describes how tasks(includes containers) should launch ||| configs: {CPU/Memory, images, port, volume} •Template used by Amazon ECS to launch tasks • Parallels to docker run parameters • Defines requirements (CPU/Memory, Container images, logging, IAM role, etc.)
- ECS Task: • Running instance of task definition • Task contains one or more containers
- ECS Service: •Maintains desired # of running tasks (Ensures a certain number of tasks are running at all times. You inform the service by the number of tasks you want, and it makes sure the number of tasks are running, if there is no tasks it creates the tasks • Restart containers that crashed) •Replaces unhealthy tasks •ELB integration • EC2 fails, restarts a task on a working EC2 instance

Tasks vs Services:
Tasks: On demand workloads/ ECS task scheduler / run once or interval / batch jobs / RunTaskAPI /startTask (custom) ||| Services: long-running apps / ECS service scheduler / health management / Scale-up and down / AZ aware / Grouped container / ELB integration

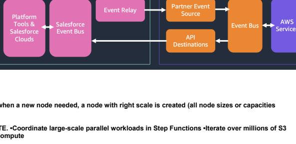
- Scheduling: Where to run your containers in the most efficient way and that's something that kubernetes does really well
- Task Placement: Process of where to put your tasks or in which instance put your tasks. • Cluster ConstraintP: Satisfy CPU, memory, and networking requirements • Custom Constraint: Filter for location, instance-type, AMI, or other custom attribute constraints • • Placement Strategies: Identify instances that meet spread or binpack placement strategy (start from beginning instance to the end by putting one task to each and turn back to the beginning instance)
  - Apply Filter: Select final container instances for placement

Fargate: • Patching and Upgrading OS, agents, etc. • Scaling the instance fleet for optimal utilization
Fargate Supported Operating Systems: • Amazon Linux 2 (ARM64 and X86_64) • Microsoft Windows Server [2019 Full (X86_64), 2019 Core (X86_64), 2022 Full (X86_64), 2022 Core (X86_64)]
Fargate Security Benefits: •Patching (OS, Docker, ECS Agent, etc.) •Task isolation (via Clusters) • No --privileged mode for containers •Requires awsvpc network mode so there is an ENI and SG per Task • Simple secure and auditable way to run commands in container – ECS Exec
ECS on EC2 vs ECS on Fargate:
–ECS on EC2: If you have existing EC2 hardware already and you want to utilize / sustained and predictable tasks with high utilization (services) / pay for hardware behind EC2 instances / great for control
–ECS on Fargate: Pay for duration and the amount of resources / spend less time on set up and maintenance / adhoc jobs with variable utilization / great for flexibility / patching (OS, Docker, ECS Agent , etc.) / Task isolation (via Clusters)
Note: Amazon ECS forms the building blocks for various services at Amazon like SageMaker, Lex, Polly, AWS Batch
ECS & Load balancer: Assign LB to service and it will route the traffic to instances
What is Kubernetes? Open source container management platform. Helps you run containers at scale. Gives you primitives for building modern applications.
What is Amazon EKS? • Amazon EKS runs vanilla Kubernetes; EKS is upstream and a certified conformant version of Kubernetes (with backported security fixes) • Amazon EKS supports 4 versions of Kubernetes, giving you time to test and roll out upgrades • Amazon EKS provides a managed Kubernetes experience for performant, reliable, and secure Kubernetes • Amazon EKS makes Kubernetes operations, administration, and management simple
Amazon EKS Runtimes Overview: Administrator deploys Pod → Kubernetes schedules the pod → EC2 or Fargate (EC2 → Run containers on EC2 instances within your account that you manage and configure || Fargate → A fully managed container environment with no infrastructure management)
Amazon EC2 Runtime Options: 1) Self-Managed Node Groups(Bring your own Autoscaling Groups running your own custom AMI. You are responsible for patching and the underlying OS.) 2)Managed Node Groups(Provisioned by EKS in your VPC. They run the latest EKS optimized AMI. Handles automatic draining and rolling out new AMIs.)
Bring Your Own OS or use ours: •EKS AMI Build Scripts •Source of truth for Amazon EKS Optimized AMI •Easily build your own Amazon EKS AMI •Build assets for Amazon EKS AMI for each supported Kubernetes version •Bottlerocket • Purpose-built by Amazon Web Services for running containers on virtual machines or bare metal hosts •Includes only the essential software to run containers, which improves resource usage and reduces the attack surface
AWS Fargate Runtime: Overview: Secure isolation of Tasks by running a dedicated kernel without shared CPU, memory, or storage. || Right—sized instances based CPU and Memory requests within the pod specification. || AWS manages the patching and upgrading of the underlying infrastructure, you manage the container || Considerations: DaemonSets are not supported, Only supports Application Load Balancers, Privileged Containers are Not Supported, Do not support host networking, Can only run within private subnets.
Kubernetes Ingress and The AWS Load Balancer Controller
Kubernetes Ingress object: • Exposes HTTP/HTTPS routes to services within the cluster • Many implementations: ALB, NGINX, F5, HAProxy etc. • Default service type: ClusterIP
AWS Load Balancer Controller: • Manages AWS Elastic Load Balancing for an Amazon EKS or a Kubernetes cluster •Uses standard Kubernetes resources (v1: Service (Type: LoadBalancer) || networking.k8s.io/v1: Ingress || networking.k8s.io/v1: IngressClass) •Custom resources (elbv2.k8s.aws/v1beta1: TargetGroupBinding) • Supports latest ELB features
EKS (ELASTIC KUBERNETES SERVICE)
-AWS manages the control plane
-Maintain master nodes
-Install control plane process (API server | scheduler || control manager)
-Scaling and backups
-You only have to worry about worker nodes (configure & manage yourself)

-For a single master node, you should have at least 3 worker nodes.
-Worker node is like a machine
-You can create namespaces to separate your project
-There are some IPs and ports that are special for kubernetes.
-EKS supports running containers on EC2 instances or Fargate. Clusters can run containers on a single runtime of multiple runtimes at the same time.
! All worker nodes, which belong to a master node, have their storage and also they have shared storage.

Worker Nodes
-EKS doesn't manage worker nodes, you have to
1)Self managed nodes → Manually manage the EC2 instances, Kubernetes process, updates/security, register node with plane
2)Managed nodes → Automate management of the EC2 instances (EKS optimized image) ||| Still managed by you but easier (create/update/terminate)
3)Fargate → You don't have to configure any worker nodes ||| Create worker node on demand ||| Automatically select optimal EC2 size

Worker Node Active-Passive Approach: A passive worker node turns into active when something happens to the active worker node. You are not paying for a passive worker node.
Worker Node Active-Active Approach: This approach is like a load balancer. Working nodes are active and work all together. Under high load, load is distributed between worker nodes.



Why EKS
-Kubernetes hard to manage & scale
-Properly secure Kubernetes increases operational overhead
-Integration with AWS services

ECS	vs	EKS
-AWS specific -Simpler architecture -Pricing no cost for control plane, only for infrastructure		-Open source, same api on all cloud providers -Steep learning curve -Large community → more support online ||| access to all K8s world -More expensive as you also pay for control plane
When to use: -App not complex & no need for K8s -Cost factor		-Already using or need access for K8s features -Prioritize open source -Leverage multi-cloud, hybrid cloud solution

Kubernetes Autoscaling:
1)Horizontal Pod Autoscaling(HPA): Increases the pod number by creating new instances.
2)Vertical Pos Autoscaling(VPA): manages the resource allocation / increase the power of existing compute units/pods
3)Cluster Autoscaling: Correct distribution of containers in the worker nodes / prevent unnecessary capacity usage with correct distribution / when a new node needed, a node with right scale is created (all node sizes or capacities won't be the same.) / termination of underutilized notes / karpenter open-source cluster autoscaling

- Large scale data processing with AWS Step Functions: DYNAMICALLY PROCESS LARGE ARRAYS OF DATA WITH DISTRIBUTED MAP STATE. •Coordinate large-scale parallel workloads in Step Functions •Iterate over millions of S3 objects such as JSON or CSV files •Up to 10,000 parallel executions •Invoke Lambda or ECS/Fargate for large scale on demand serverless compute

NETWORKING

AMAZON VPC => It allows you to provision a logically isolated section of the AWS cloud where you can launch AWS resources on a virtual network that you define. Lets you control your virtual network resources, including the following: -Selecting the IP Address Range -Creating Subnets - Configuring Routing Tables and Network Gateways. Amazon VPC allows you to customize the network configuration of your VPC. It allows you to use multiple layers of security.

- VPC IP Addressing: • Support IPv4 and IPv6(-IPv4: VPCs can be between /16 and /28 -IPv6: /56 for VPC's & /64 for subnets (fixed)) • VPCs support subnetting •VPC CIDRs cannot be modified once created •Additional CIDRs can be added to a VPC •Supports bringing your own IP space in the newly launched regions too • AWS announced Amazon-provided contiguous IPv6 CIDR blocks
- VPC IP addressing considerations: • Plan your IP space before creating it (-Overlapping IP spaces = future headache -Consider using multiple VPCs -Consider future AWS region expansion -Consider future connectivity to corporate networks) •The VPC IP Address Manager (IPAM) feature can be leveraged to plan, track, and monitor IP addressing in AWS •Amazon VPC IP Address Manager (IPAM) now manages IP Addresses in your network outside your AWS Organization • Amazon VPC IP Address Manager introduces a customizable dashboard with new insights
- Amazon VPC IP Address Manager (IPAM): Plan, track, and monitor IP addresses across AWS accounts and AWS Regions. Works for IPv4 and IPv6. •Automate IP assignments(Automate across multiple Regions and accounts based on your application's unique networking and security needs) •Monitor across the network(Avoid downtime with overlap detection, tracking IPs in compliance with security policy, and utilization trends) • Retrospective analysis(Faster troubleshooting and auditing)

Public IP: EC2 instance gets a new IP every time you stop and then start. ||| Private IP: mixed for EC2 instance ||| Elastic IP: allows you to attach a public IP (fixed) to an EC2 instance, which can be remapped, Dynamically assigned, Specific to a region, Useful for redundancy when Load Balancers are not an option
VPC → Virtual Private Cloud, deploy resources like EC2 instances. VPC CIDRs cannot be modified once created, but you can add more CIDRs. ||| VPC IP Address Manager (IPAM) feature can be leveraged to plan, track, and monitor IP addressing in AWS (Automate IP assignments, Avoid downtime with overlap detection, tracking IPs in compliance with security policy, and utilization trends, Faster troubleshooting and auditing)

- Subnets → allocated as a subset of the VPC CIDR range and span a specific AZ, You can have up to 200 subnets per VPC, default route table(Implicit route between all subnets within a VPC).
- Routing tables → Each subnet has an associated routing table, Routing tables can be associated with multiple subnets, You can have 50 routes per route table.
- Routing: Route Tables direct traffic towards Internet/NAT Gateway, VPC Peering, Transit Gateway, and Direct connect. ||| If you add 0.0.0.0/0 to the routing table it has to be associated with the public subnet.
- Internet Gateway(VPC to Internet) → VPC connect to the internet. 1 Internet gateway per VPC. It must be referenced on the route table. It performs 1:1 NAT between public and private IP addresses. Horizontally scaled, redundant, highly available VPC component.
- Public IP addressing: Elastic IP Address: •Static, Public IPv4 address, associated with your AWS account •Dynamically assigned •Specific to a region •Can be associated with an instance or network interface •Can be remapped to another instance in your account • Useful for redundancy when Load Balancers are not an option •EXAMPLE: You have an EC2 instance with Private IP: 10.0.0.1 and Elastic IP: 198.51.100.2. The elastic IP address remapped to another instance so this EC2 instance have no longer that Elastic IP address but the other EC2 instance would have that Elastic IP: 198.51.100.2. and Private IP: 10.0.0.2
- NAT Gateway(Outbound only traffic) → Enable outbound connection to the internet. No incoming connection(useful for OS/packages updates, public web services access). Supports TCP, UDP, and ICMP protocols. Network ACLs(Access Control List) apply to NAT gateway traffic.

VPC Security:

- Network Access Control List(ACL) → •Inbound and Outbound •Subnet level inspection (IP inspection - IP Rules only) •Optional level of security •By default, allow all traffic • Stateless •IP and TCP/UDP port based •Supports allow and deny rules • Deny all at the end. → IP Rules only. EXAMPLE: ACL can be set for the private subnet which contains MySQL DB Amazon Aurora such that only the EC2 instance under a public subnet can connect through TCP 3306 but other IPs and other Ports are denied for connection.
- Security Group (Resource FW) → • Stateful firewall •Inbound and Outbound customer defined rules •Instance/Interface level inspection (Micro segmentation. Mandatory, all instances have an associated SG) •Can be cross referenced (Works across VPC Peering) • Only supports Allow rules •Implicit Deny all if not allowed • AWS Verified Access adds a new logging functionality to improve troubleshooting

VPC Load Balancing:

- Elastic Load Balancer (Horizontal Scaling) ELB → • Distribute traffic to multiple targets(EC2 instances, Containers, IP addresses, lambda functions) •Multiple Availability Zones • ELB Scales automatically • Support Auto Scaling Groups (Automatically (de)register instances to the ELB based on health checks) • health checks • Two types of ELB: NLB/ALB
- Application load balancer(ALB) → •Layer 7 Load Balancing •Content-Based Routing (host and path based) •HTTP/2 Support (HTTP/HTTPS only) •Request Tracing •Web Application Firewall (WAF) integration •Application Load Balancer now supports TLS 1.3 •Cross-zone load balancing •WebSocket Support •Deletion Protection. • Used for microservice calls and web.
- Network Load Balancer (NLB) → • Layer 4 Load Balancing • Connection-based Load Balancing • High Throughput • Low Latency • Preserve source IP address • Static IPs • Long-lived TCP Connections • Supports Security groups ⇒ Ultra high performance, TCP/UDP, used when there is traffic of API requests, data streaming, different networks created, high throughput.
- Gateway Load Balancer →(Third-party) Deploy and manage fleets of virtual appliances in elastic, fault-tolerant manner. Available to customers and technology partners (Third-party appliance fleet). Used for Security Analysis, Intrusion detection, Route Traffic to Firewalls.
- Both ALB and NLB have websocket support, deletion protection, and cross-zone load balancing.

VPC Connectivity Options

- VPC Endpoint (Stay on AWS network) → •Connect your VPC to: Supported AWS services, VPC endpoint services powered by PrivateLink (such as connect EC2 instance in a subnet to NLB in another VPC through VPC endpoint) • One VPC Endpoint is required for each connection, so you may have several VPC endpoints •Doesn't require public IPs or Internet connectivity (connection through private network) •Horizontally scaled, redundant, and highly available •Robust access control •Metrics for traffic visibility. → Connect to AWS services through a private network. Doesn't require internet connectivity.
- VPC Peering (Connect Multiple VPCs) → connect between VPCs in a private manner, behave as if they are in the same network. Must not have overlapping IP ranges otherwise they won't be able to communicate. Connection not transitive. • Scalable and high available •Supported between AWS accounts •Supported across AWS Regions •Bi-directional traffic •Remote Security groups can be referenced •Routing policy with Route Tables (Not all subnets need to connect to each other) •No overlapping IP addresses • No transitive routing (vpc1 <--> vpc2 <--> vpc3 ⇒ although vpc1&vpc2 and vpc2&vpc3 are connected vpc1&vpc3 are not connected.)
- Transit Gateway(Multiple VPCs access models) → thousands of VPC peerings & connecting on premise infrastructure. Centralize VPN and AWS Direct Connect connections. Control segmentation and data flow with Route Tables. Hub and Spoke Design

Connecting to On- premises

- Virtual Private Gateway(VGW) → Fully managed VPN endpoint device. One VGW per VPC. Default 10 Site-to-Site VPN connections per VGW, Scalable, Dynamic(BGP) or Static Routing. Redundant IPSec VPN Tunnels Terminating in different AZs. IPSec AES 256-bit encryption SHA-2 hashing
- AWS Direct Connect (DX): Dedicated network connection from your premises to AWS. AWS Dedicated Local Zones. Dedicated Connection (1,10 or 100Gbps, Supports multiple VIFs). AWS Partner Hosted Connection (50Mbps to 10Gbps, Single VIF). Consistent network performance (dedicated bandwidth, low latency). Reduced data charges. Connect to 108 Direct Connection Locations across the globe. Use Cases: working with a large dataset as it increases bandwidth throughput and low cost as it doesn't go through the public internet. Real time data feeds as it provides more consistent network experience. Suitable for hybrid environments. Dedicated links to AWS Direct Connect: For redundancy, DX can deployed with single or multiples of circuits, providers, customer gateways, direct connect locations, customer data centers.
- Direct connect gateway → If you want to setup a Direct Connect to one or more VPC in many different regions (same account), you must use Direct Connect Gateway. It connects two regions. • Transit VIF (Connects to a AWS Transit Gateway) •Simplify your network architecture and management overhead • Create a hub-and-spoke model that spans multiple VPCs, Regions, AWS accounts
- Site to Site VPN + Transit Gateway connect datacenter with multiple VPCs.

Global Accelerator → Uses AWS Global Network from Edge to Region, Route client to closest healthy endpoint. Uses AWS Global Network from Edge to Region. Client traffic ingresses via the closest available Edge location. Route client to closest healthy endpoint. No DNS switchover required, same IP address globally (Static IP Anycast).

STORAGE

Three types of storage: Block Storage, File Storage, Object Storage
Block Storage: Raw data, array organization (array of unrelated data blocks), data on disk (physical structıre - hard disk), EBS, Instance Storage, Storage Area Network (SAN) Storage Arrays ||| File Storage: Unrelated data blocks managed by file (serving) system, data on disk (e.g., EFS, FSx, FileCache, Network Attached Storage (NAS) Appliances, Windows File Servers, NetApp OnTap). ||| Object Storage: Objects stored as objects (put image, get image). Like dropbox. To reach data, I need an ID. Policy-based, metadata-driven. Stores virtual containers that encapsulate the data, data attributes, metadata, and object IDs. e.g., S3. ||| Storage Characteristics: Durability (expected data loss), Availability (expected downtime - time for reaching the data), Security, Cost, Scalability, Performance (bandwidth), Integration. Durability calculation 365.25*24*60 (min) is considered as %100. To calculate the durability: 1-n/(24*60*365.25) ||| Availability vs Durability: 5 min 15 sec (A), 1 in 100,000 (D). Hence, availability is a time metric, durability represents data loss. |||

Block Storage (mounting on OS/VM/hosts): General Purpose SSD, Provisioned IOPS SSD, Throughput-Optimized HDD, Cold HDD. || File Storage: EFS, FSx for OnTap/Windows/Lustre/OpenZFS. || Backup: AWS Backup || Object Storage: S3 (family). ||| Re-host: Amazon EBS, EC2 || Re-Platform: EFS, AWS Storage Gateway Family, AWS FSx || Re-architect: Amazon S3 || Re-architect & Re-platform: Amazon FSx for Lustre. ||| BLOCK STORAGE PORTFOLIO: Instance Storage: Temporary, attached to host hardware (ssd or hdd),suitable for information that frequently changes or replicated across multiple instances, non-persistent (when your machine terminate your storage is gone), no snapshot support. | Amazon EBS: Even if your machine terminates, your data is in EBS. Easy to use, high-performance block storage service designed for using with EC2 for both throughput and transaction-intensive workloads. | Snapshots: Incremental, point-in-time copies of your EBS data that can be used to restore new volumes, expand the size of the volumes, move volumes across AZs. Snapshots are stored in S3. Snapshots are like zip files, you need a snapshot backup process to open it. ||| EBS is designed for a wide range of workloads on EC2 (enterprise applications, relational databases, non-relational / NoSQL databases, big data analysis, file/media). || EBS→ low latency, consistent, high IOPS (input/output operations per second) and high throughput, scalable without disruption. | EBS→easy to use, high reliability, virtually unlimited scale, secure, cost-effective. || Six different volume types for optimal use: SSD (gp2-gp3, io1-io2), HDD(st1, sc1). |||

General Purpose SSD - gp2: Use cases: Most workloads, relational and non-relational database workloads, boot volumes, development and test environments, virtual desktops.||| General Purpose SSD - gp3: Use cases: Latest generation general-purpose SSD-based EBS volumes. Provision performance independent of storage capacity, while lower pricing than existing gp2 volumes. ||| Provisioned ones are faster than General Purpose ones but for read/write IO operations. ||| Provisioned IOPS SSD io1: Use Cases: Large database workloads, mission-critical business applications requiring sustained high-performance. ||| Provisioned IOPS SSD io2: Use Cases: Latest generation of the Provisioned IOPS SSD volumes that is designed to provide 100X durability of 99.999% as well as a 10X higher IOPS to storage ratio of 500 IOPS for every provisioned GB –at the same price as the previous generation (io1) ||| Provisioned IOPS SSD io2 Block Express: Use Cases: io2 Block Express offers the highest performance block storage in the cloud with 4x higher throughput, IOPS, and capacity than io2 volumes, along with sub-millisecond latency. Block Express is the next generation of Amazon EBS storage server architecture purpose-built to meet the performance and latency requirements of the most demanding applications.||| Throughput Optimized HDD for frequently accessed, throughput-intensive workloads (st1): • Use cases: st1 is backed by hard disk drives (HDDs) and is ideal for frequently accessed, throughput-intensive workloads with large datasets and large I/O sizes. ||| Cold HDD for infrequently accessed workloads (sc1): • Use cases: sc1 is backed by hard disk drives (HDDs) and provides the lowest cost per GB of all EBS volume types. It is ideal for less frequently accessed workloads with large, cold datasets. ||| Cold HDD is the slowest, and cannot be used for live systems. They are used for backup. Throughput HDD (st1) can be used for live systems. |||

EC2 instance storage: local to instance, non-persistent data store, data is not replicated by default, no snapshot support, pricing included in instance cost, SSD. ||| Fully-managed backup with EBS Snapshots: Backup, Restore, Low cost, Protection, Agility (for quick restoration of volumes across AZ within a region). ||| Amazon Elastic File System (EFS): serverless, very durable, EFS automatically understands whether your data is hot, warm, or cold, so it manages accordingly. Elastic System so you pay what you use. Not connected to EC2. Shared storage. Mostly for microservices, 1 or more AZs. EFS is a shared system so you cannot use your OS directly. USE CASE: You have an analytics application. You have big data with small sizes. What would you use? EFS or EBS? Use EFS because it would be cheaper, EBS works in your instance which is more expensive. You can also reach EFS through other microservices like analytics. Put EFS in AZ, you need connection rules to reach them. ||| EFS → Migrate: lift and shift to AWS cloud without refactoring application, Optimize: enable cost efficiency, Modernize: build microservices into application with common data platform, Innovate: improve development efficiency, build new features, enter new markets ||| Shared storage in EFS: different apps using the same data. Live OS, it locks the data when updated by an app. ||| Use cases for Amazon EFS: 1) Low-latency: home directories, DevOps, Application dev & test. 2) Mid-latency: Enterprise apps, database backups, web serving & content management. 3) High throughput: Analytics, ML, media workflows. ||| Amazon File Cache: Keeps the most recent data, you can access your data from EC2 very fast, you should update your cache according to your files. Flexible data access across on-premises, AWS file and Amazon S3 datasets (cloud or on premises). Benefits: Fast (accelerate workloads), Agile (use anywhere - seamless data access), Simplified (unified datasets): link network file systems (NFS,) or S3 buckets for a single view of multiple datasets. ||| Amazon FSx for Windows File Server: Fully-managed file storage built on Windows Server, easy migration to AWS. Fully managed Windows FS means you no longer have to manage hardware & manage software. This service provides fully managed Windows file servers, offering compatibility with Microsoft Windows file system features and native Windows access control lists (ACLs). It's built on Windows Server and provides features such as user quotas, file-level restore, and integration with Microsoft Active Directory. FSx for Windows File Server is suitable for a wide range of use cases including enterprise applications, home directories, content management systems, and business applications that require shared file storage. It offers high availability and durability with automatic backups, data replication across multiple Availability Zones, and continuous monitoring for performance and availability. S3 → storage type flexibility (SSD/HDD), • Deployment type flexibility (Single-AZ / Multi-AZ) • Select throughput and storage independently • Choice of in-line snapshots and backups stored in S3 • Data deduplication and compression. |||

S3 vs EFS: EFS shared storage cannot be used directly in your OS. ||| Object Storage: Like dropbox or Google Drive. Web accessible object storage. You just put data & access it via a link. Bucket is a folder in AWS, each bucket is unique. You can run your static website S3 (html, js). 7 types of s3, Glacier part is the slowest. All S3 services are used with 3 or more AZs.When you create a bucket, it is public by default. We use S3 fro async processes. ||| Replication Time Control (RTC): set time to replicate your data. If you put your data and delete before the replication time you may not reach it. ||| S3 Access Management & Security: IAM integration, Access Control List (ACL), S3 bucket policies, S3 access points, server-side & client-side encryption, S3 object lack, private link for S3, VPC interface endpoints for S3.
How do customers benefit from the cloud? *Agility: Teams can experiment and innovate quickly and frequently. *Cost Savings: Only pay for what you use, leveraging economies of scale. *Elasticity: Easily scale up or down with the needs of the business. *Innovation: Focus on business differentiators, not infrastructure. *Global Footprint: Extensive, reliable, and secure global cloud infrastructure.

Abbreviations:

ECS → Elastic Container Service	ECR → Elastic Container Registry	FM → Foundational Model
EKS → Elastic Kubernetes Service	IAM → Identity and Access Management	SAN → Storage Area Network
KMS → Key Management Service	VPC → Virtual Private Cloud	NAS → Network Attached Storage
IaaS → Infrastructure as a Service	PaaS → Platform as a Service	ENI → Elastic Network Interface
SaaS → Software as a Service	EC2 → Elastic Compute	VPN → Virtual Private Network
vCPU → Virtual Central Processing Unit	EBS → Elastic Block Store	DX → Direct Connect
GiB → GibiBytes → 256 GiB = 275 GB	AMI → Amazon Machine Image	VIF → Virtual Interface
ADA → API Driven Architecture	S3 → Simple Storage Service	DNS → Domain Name Resolution
CVE → Common Vulnerabilities and Exposure	EDA → Event Driven Architecture	GWLB → Gateway Load Balancer
ALB → Application Load Balancer	ELB → Elastic Load Balancer	RTB → Routing Table
CIDR → Classless Inter-Domain Routing	NLB → Network Load Balancer	NAT → Network Access Translation
IPAM → IP Address Manager	NFS → Network File System	BPA → Block Public Access

Dictionary:
Pod → A thin wrapper around one or more container under worker node
Daemon Set → Implement a single instance of a pod on a worker node. It ensures there is one pod when a worker node is created.
Replica Set → Ensures defined number of pods are always running.
Job → Ensures a pod property runs.
Service → Maps a fixed IP address to a logical group of pods.
Label → Key/values pairs used for association and filtering
Kubectl → Command line interface
Deployment → Details how to roll out (or roll back) across versions of your applications
Namespace → Logically named group
Karpenter → Open-source, flexible and high performance Kubernetes cluster autoscaler
Hot data → Frequently used data
Warm data → Monthly used data
Cold data → not used much data (yearly)

AWS DATABASE

- Early days: on-premise / monolithic architecture → RDBMS
- Challenges with on-premise / monolithic architectures: Managing on-premise database environments - time consuming and complex ||| Scaling Limitations: Horizontally scaling/Sharding requires complex application logic. Cost of provisioning for peak and licensing ||| Availability Challenges: HA cluster setup and data replication for high availability. Security updates require downtime ||| High Operations: Engineers/DBAs need to invest significant effort in provisioning fine tuning database configurations, patching and upgrades. Deal with operational issues with database outages or availability drops
- Modern real-time applications require → performance, scalability and availability → Users (1M+), Data Volume (Terabytes-petabytes), Locality(Global), Performance (Microsecond latency), Request rate (Millions per second), Access (Mobile, IoT, devices), Scale (up-down-out-in), Economics (pay as you go), Developer access (Open API)
- Shifting to microservice architectures with purpose-built databases to keep up with requirements of modern applications.
- Common data models and use cases: Relational | Key-value | Document | In-memory | Graph | Wide-Column | Time-Series | Ledger
- Relational DB Use Cases: Divide data among tables | highly structured | relationships established via keys | Data accuracy and consistency enforced by the system.
- Amazon RDS: Managed relational database service with popular db engines (Amazon Aurora, MySQL, PostgreSQL, MariaDB, Microsoft SQL Server, Oracle) ||| Easy to Administrator(No need to provision infrastructure, install, and maintain DB software) ||| Available & Durable (Automatic Multi-AZ data replication; automated backup, snapshots, and failover) ||| High Scalable (Scale DB compute and storage with a few clicks; minimal downtime for your application) ||| Fast & Secure (SSD storage and guaranteed provisioned I/O; data encryption at rest and in transit)
- Amazon Aurora: MySQL and PostgreSQL-compatible relational database built for the cloud ||| Performance & scalability (5x throughput of standard MySQL and 3x of standard PostgreSQL; scale-out up to15 read replicas) ||| Availability & Durability (Fault-tolerant, self-healing storage; six copies of data across three AZs; continuous backup to S3) ||| Highly Secure (Network isolation, encryption at rest/transit) ||| Fully Manage (Managed by RDS: no server provisioning, software patching, setup, configuration, or backups) Faster & cheaper than RDS

-Key-value DB Use Cases: simple key-value pairs | partitioned by keys | consistent performance at scale | example: DynamoDB ||| DynamoDB: Fast and flexible NoSQL database service for any scale ||| Performance at scale: Handles millions of requests per second | Delivers single-digit-millisecond latency | Automated global replication | Advanced streaming with Amazon Kinesis Data Streams for Amazon DynamoDB. ||| No servers to manage: Maintenance free | Auto-scaling | On-demand capacity mode | Change data capture for integration with AWS Lambda, Amazon Redshift, and Amazon OpenSearch Service ||| Enterprise ready: ACID transactions (Atomicity, Consistency, Isolation, and Durability). | Encryption at rest | Continuous backups (supporting point-in-time recovery), and on-demand backup and restore | PartiQL (a SQL-compatible query language) support.
- Document DB Use Case: Store, query, & index JSON data natively |||| Amazon DocumentDB: Decoupled compute Decoupled compute and storage support independent scaling, Scale to millions of reads using read replicas for instance-based cluster, Scale to over a million of writes and reads and 1 PB of storage with Elastic Clusters ||| Improve productivity and lower total cost of ownership by removing undifferentiated database management tasks ||| High availability and durability, Global Clusters provide local reads and disaster recovery from Region-wide outages, Built-in security best practices with encryption-in transit and encryption-at rest, Amazon VPC, and AWS KMS ||| Supports hundreds of APIs, operators, and data types, MongoDB APIs, drivers, and tools can be used with little to no change
- In Memory DB Use Case: Latency: Extremely low, microsecond to millisecond responses ||| Request Rate: Thousands to millions of reads and/or writes per second ||| Data Volume: Will scale up to over 100TB ||| Amazon ElastiCache: In-memory data store and cache for sub-millisecond response times (Extreme performance) ||| AWS manages all hardware and software setup, configuration, and monitoring ||| Read scaling with replicas, Write and memory scaling with sharding, Nondisruptive scaling ||| Amazon MemoryDB: Fully compatible with open source Redis Cluster ||| In-memory database for microsecond reads ||| Network isolation, encryption at rest/transit, multi AZ, and automatic failover ||| Scale writes and reads with sharding and replicas
- Graph Database Use Case: Social news feed, recommendations, Retail fraud detection ||| Amazon Neptune: Fast, reliable graph database built for the cloud. ||| Open(Supports Apache TinkerPop & W3C RDF graph models) ||| Fast(Query billions of relationships with millisecond latency) ||| Reliable(6 replicas of your data across 3 AZs with full backup and restore) ||| Easy(Build powerful queries easily with Gremlin and SPARQL)
-Wide-Column Cassandra: Amazon Keyspaces (for Apache Cassandra): Works with Apache 2.0-licensed Cassandra drivers and developer tools ||| Fast & Scalable:Consistent, single-digit millisecond read and write performance at any scale ||| Fully managed (serverless): No servers to provision, patch, or manage, no software to install, maintain, or operate ||| Highly available & Secure: Data is replicated 3 times across multiple AWS Availability Zones, Encryption at rest.
-Time-series Use Case: Application events, IoT Sensor Readings, Devops data - you need to keep data in order according to time, specific to handle slowness and fetch data in order according to time compared to RDS. |||
Amazon Timestream: serverless time series database service for IoT and operational applications ||| Collect fast moving time-series data from multiple sources at the rate of millions of inserts per second ||| Capable of processing trillions of events daily; the adaptive query processing engine maintains steady, predictable performance ||| Built-in analytics for interpolation, smoothing, and approximation to identify trends, patterns, and anomalies ||| Serverless: no servers to manage; time-consuming tasks such as hardware provisioning, software patching, setup, & configuration done for you
-Ledger Amazon QLDB: Knows next part of the chain, immutable: maintains a sequence record of all changes to your data, which cannot be deleted or modified; you have the ability to query and analyze the full history ||| Cryptographically verifiable: uses cryptography to generate a secure output file of your data's history ||| Highly scalable: executes 2-3X as many transactions as ledgers in common blockchain frameworks ||| Easy to use: letting you use familiar database capabilities like SQL APIs for querying the data. yoyo

AirBNB Case: User Search History (Amazon DynamoDB): Massive data volume, Need quick lookups for personalized search ||| Session state (Amazon ElastiCache): In-memory store for submillisecond site rendering ||| Relational Database (Amazon RDS): Referential integrity, Primary transactional database

When to Use?
Relational → Use when you're migrating or modernizing an on-premises relational workload or if your workload has less predictable query patterns.||| Optimized for structured data stored in tables, rows, and columns. They support complex queries through joins. ||| *Amazon Aurora, Amazon RDS*
Key-value → Use for workloads such as session stores or shopping carts. Key-value databases can scale to large amounts of data and extremely high throughput of requests, while servicing millions of simultaneous users through distributed processing and storage. ||| Optimized for consistent single-digit millisecond performance at any scale (meaning any number of writes and reads). ||| *Amazon DynamoDB*
In-memory → Use Amazon ElastiCache when you need a caching layer to improve read performance. Use Amazon MemoryDB for Redis when you need full data persistence, but still need sub-millisecond read latencies. ||| ElastiCache is optimized to support microsecond reads and sub-millisecond writes. MemoryDB supports microsecond reads and single-digit milliseconds writes. ElastiCache is an ephemeral cache while MemoryDB is an in-memory database. ||| *Amazon ElastiCache, Amazon MemoryDB for Redis*
Document → Use when you want to store JSON-like documents with rich querying abilities across the fields of the documents. ||| Optimized for storing semi-structured data as documents with multi-layered attributes. ||| *Amazon DocumentDB (with MongoDB compatibility)*
Wide column → Use when you need to migrate your on-premises Cassandra workloads, or when you need to process data at high speeds for applications that require single-digit-millisecond latency. ||| Optimized for workloads that require heavy reads/writes and high throughput coupled with low latency and linear scalability. ||| *Amazon Keyspaces (for Apache Cassandra)*
Graph → Use when you have to model complex networks of objects, such as social networks, fraud detection and recommendation engine use cases. ||| Optimized for traversing and evaluating large numbers of relationships, and identifying patterns with minimal latency. ||| *Amazon Neptune*
Time series → Use when you have a large amount of time series data, potentially from a number of sources, such as Internet of Things (IoT) data, application metrics, and asset tracking. ||| Optimized for storing and querying data that is associated with timestamps and trend lines. ||| *Amazon Timestream*
Ledger → Use when your organization has to communicate with other entities (businesses, customers) and you need a way to verify and trust each other, or when you not only need to retrieve the current state of data, but need to prove how data mutated into the current state. ||| Optimized for maintaining a complete, immutable, and verifiable history of database changes. ||| *Amazon Quantum Ledger Database (Amazon QLDB)*
Amazon Redshift: Amazon Redshift uses SQL to analyze structured and semi-structured data across data warehouses, operational databases, and data lakes, using AWS-designed hardware and machine learning to deliver the best price performance at any scale.

- Amazon Redshift is a fully managed, petabyte-scale data warehouse service that provides a scalable solution for analyzing large volumes of data using standard SQL and BI tools.

Can I put/use non-relational data in relational db? → yes, but it is expensive as some columns will have empty blocks ||| relational data in non relational db? → yes

SECURITY

Foundational Security controls:Accurate account information • Protecting your Root User • Using IAM Users • Configuring alarms • Turning CloudTrail ON • Using AWS Config • Using Trusted Advisor
AWS Shared Responsibility: Infrastructure Services (EC2) → More Customizable, More Customer Responsibility than Container services (RDS) then Abstracted Services (S3, Lambda, DynamoDB) → Less Customizable, Less Customer Responsibility, More best practices built-in
Foundational Security Control → Accurate account information, Protecting your Root User, Using IAM Users, Configuring alarms, Turning CloudTrail ON, Using AWS Config, Using Trusted Advisor ||| Protect Root User → Multi-factor Authentication (MFA), Complex Password, Deleting access key after you have used it, Set up Alarms, You can separate password and MFA between two people so that you can login only if you are together.
AWS CloudTrail → Auditing, Governance and Compliance ||| Record Activity: actions taken by user, role, or AWS service ||| Simplify compliance → by automatically recording and storing activity logs ||| Gain Visibility: view, search, download, analyze, respond ||| Troubleshoot → who/what took which action and when ||| Build Security Automation: tracking and responding to threats |||
Features: CloutTrail Insights, CloudTrail Lake, Log Encryption, File Integrity Validation ||| Auditing: Capture and log user and resource activity across your AWS infrastructure and resources for governance and auditing, Enable compliance, operational and risk auditing. Store those events in S3 bucket. Act(Trigger actions when important events are detected). Review(Analyze recent events and logs with Amazon Athena or CloudWatch Logs Insights) ||| Components of CloudTrail: Audit Trails: Configure recording events across all your AWS accounts and regions • Centralize logging across your AWS Organization • Create additional trails as needed for operations, support, and security needs • Configure trail to select relevant events for delivery by choosing management events (all, read, write, exclude KMS), and/or data events (all or specific S3 buckets and Lambda functions) ||| Event Delivery: Deliver events to Amazon S3, Amazon CloudWatch Logs or Amazon EventBridge • Get SNS notifications when events are delivered • Enable encryption using SSE or KMS • Cryptographically validate whether log file was modified, deleted or unchanged ||| Search and Analytics: Lookup recent (90-day) event history on Console or API • Leverage integration with CloudWatch Logs, or Amazon Athena to query events • Import logs to Amazon Elasticsearch Service, or AWS Partner Solutions for deeper analysis ||| CloudTrail Insights: Identify unusual operational activity such as spikes in resource provisioning, or gaps in periodic maintenance activity • Enable automatic analysis of events to establish a baseline for normal behavior and detect anomalous patterns. • Remediate operational issues using actionable information in Insights events. ||| Pricing → Pay as you go, Insights and CloudTrail Lake priced separately
AWS Config → Assess, Audit and Evaluate ||| Central View: of Configurations of AWS resources ||| Relationships: View relationships between resources ||| History: How configurations changed over time ||| Notifications: Receive notifications upon resource modification ||| Compliance: Evaluate resource compliance against desired settings • Config Rules • Conformance Packs – FedRAMP, PCI, HIPAA, NIST, etc. ||| Customize: Add custom rules and conformance packs, Github repo for community sourced rules
What does AWS Config do: Configuration Auditor • Monitors configuration changes over time • Evaluates the configuration against policies defined using AWS Config rules • Alerts you if the configuration is noncompliant with your policies ||| AWS Config Flow: 1. Configuration change occurs in your AWS resources, 2. Config records and normalizes the change into a consistent format, 3A. Normalized changes are delivered to your S3 bucket, accessed through Config APIs, and optionally sent via SNS. 3B. Config automatically evaluates the recorded configurations against your desired configurations 4. Evaluations are displayed on a dashboard, accessed through Config APIs, and optionally sent via SNS. ||| Common Use Cases: Auditing configuration changes over time • Detecting misconfigurations • Verifying compliance (with internal and regulatory requirements) • Change management (third-party CMDB integration)
AWS Trusted Advisor → Analyze your AWS Resources for best practices for availability, cost, performance, and security. ||| AWS Organizations Integration: You can aggregate compliance data from all accounts in your organization • Aggregator can only be created in the master account • All features must be enabled in your organization • Authorization step is not needed in the member accounts • Aggregator automatically gets updated when accounts join or leave the organization
AWS Well Architectured Security Pillar → Detection, Infrastructure Protection, Data Protection, Incident Response, Detective Controls
1. Identity and Access Management →Define, enforce, and audit user permissions across AWS services, actions, and resources ||| AWS Identity and Access Management (IAM), Securely manage access to AWS services and resources ||| AWS IAM Identity Center, centrally manage SSO access to multiple AWS accounts and business apps ||| AWS Organizations, policy-based management for multiple AWS accounts ||| AWS Directory Service, managed microsoft active directory in AWS ||| Amazon Cognito, add user sign-up, sign-in, and access control to your web and mobile apps ||| AWS Resource Access Manager, simple secure service for sharing AWS resources ||| Amazon Verified Permissions, Fine-grained permissions and authorization for your applications
IAM Access Analyzer → Generate policies based on Access Activity, reviews your AWS CloudTrail logs and generates a policy template that contains permissions that have been used by the entity in your specified time frame. Example Diagram: IAM Access Analyzer → (Analyses) → AWS CloudTrail Trail → (Generates) → Scoped-down policy → (Retrieves for) → Further customization
2. Detective Controls → Gain the visibility you need to spot issues before they impact your business, improve your security posture, and reduce the risk profile of your environment ||| AWS Security Hub, automate AWS security checks and centralize security alerts ||| Amazon GuardDuty, protect your AWS accounts with intelligent threat detection ||| Amazon Security Lake, automatically centralize your security data in a few steps ||| Amazon Inspector, automated and continual vulnerability management at scale ||| Amazon CloudWatch, observe and monitor resources and applications on AWS, on premises, and on other clouds ||| AWS Config, asses, audit, and evaluate configurations of your resources ||| AWS CloudTrail, track user activity and API ||| VPC Flow Logs, capture info about IP traffic going to and from network interfaces in your VPC
Amazon GuardDuty → threat detection service that continuously monitors for malicious activity and unauthorized behavior to protect your AWS accounts and workloads. Threat detection using two methods: threat intelligence and machine learning ||| How it Works → Collects logs from data sources then using security analytics it analyzes those logs then if any security related data is found it passes it to Detective services like Amazon Detective, AWS Security Hub and Amazon EventBridge etc. Partner integration with GuardDuty is available. GuardDuty Treat list providers are the third party security firms. There are also customer managed threat IP lists. Example: CrowdStrike → GuardDuty → EventBridge → other apps
Amazon Inspector → automated vulnerability management service that continually scans AWS workloads for software vulnerabilities and unintended network exposure. (EC2, Lambda, ECR) ||| What it Checks for? → network exposure, List of resources analyzed: Security Groups • VPCs • Network interfaces • Subnets • Network ACLs • Route tables • Elastic load balancers • Application load balancers • Internet gateways • Virtual private gateways • Direct Connect • VPC peering connections
AWS Security Hub → Threat detection, monitoring, and response ||| Integrated with AWS Workloads in an AWS Account, along with identities and network activity ||| USE CASE: Guard Duty detects threats, Malice detects sensitive data, and Inspector detects vulnerabilities. These services passes their findings to Security Hub then it passes it to Amazon Detective to investigate the findings or to Amazon Security Lake to centralize, normalize, and analyze the findings
Amazon Detective → Analyze and visualize security data to rapidly get to the root cause of potential security issues
3. Infrastructure Protection →Reduce surface area to manage and increase privacy for and control of your overall infrastructure on AWS |||
AWS Firewall Manager, Centrally configure and manage firewall rules across your accounts |||
AWS Network Firewall, Deploy network firewall security across your VPCs |||
AWS Shield, Maximize application availability and responsiveness with managed DDoS protection. |||
AWS WAF, Protects your web applications from common exploits. |||
Amazon VPC, Define and launch AWS resources in a logically isolated virtual network. |||
AWS PrivateLink, Establish connectivity between VPCs and AWS services without exposing data to the internet. |||
AWS Systems Manager, Gain operational insights into AWS and on-premises resources. |||
AWS Verified Access, Provide secure access to corporate applications without a VPN.
AWS WAF (Layer 7 Protection) → WAF can be connected to Amazon CloudFront, API Gateway, ALB(Application Load Balancer), AWS AppSync, Amazon Cognito. It is pay-as-you-go. It is managed, elastic and integrated WAF.
DDoS protection with AWS Shield → Standard: Available to all AWS customers at no additional cost • Protection against the most common attacks (SYN/UDP Floods, Reflection Attacks, etc. Layer 3/4) • Automatic detection and mitigation ||| Advanced: Paid service that provides additional protection against sophisticated attacks ✚ Protection against advanced attacks (Layer 7) ✚ 24x7 DDoS Response Team (Proactive/Reactive) ✚ Cost protection ✚ Faster Mitigation/Better Visualization ✚ Includes WAF and Firewall Manager
AWS Network Firewall → Native Firewall ||| Automatically scale, managed AWS infrastructure ||| Highly flexible, high capacity rule engine with managed IPS rules ||| Centrally manage policies, real-time monitoring ||| There are no upfront commitments and you only pay for what you use
!!!!!!!!!!Edge location: A site that CloudFront uses to cache copies of your content for faster delivery to users at any location. ||| Layer 4 → Network ||| Layer 7 → Application Layer
4. Data Protection → A suite of services designed to automate and simplify many data protection and security tasks ranging from key management and storage to credential management. ||| Amazon Macie, Discover and protect your sensitive data at scale. |||
AWS Key Management Service (KMS), Create and control keys used to encrypt or digitally sign your data. |||
AWS CloudHSM, Manage single-tenant hardware security modules (HSMs) on AWS. |||
AWS Certificate Manager, Provision and manage SSL/TLS certificates with AWS services and connected resources. |||
AWS Private CA, Create private certificates to identify resources and protect data. |||
AWS Secrets Manager, Centrally manage the lifecycle of secrets. ||| AWS VPN, Connect your on-premises networks and remote workers to the cloud. ||| Server-Side Encryption, Flexible data encryption options using AWS service managed keys, AWS managed keys via AWS KMS, or customer managed keys.
Amazon Maice → Discover and protect your sensitive data at scale ||| Gain visibility and evaluate (bucket inventory, policies) ||| Discover sensitive data (Inspection jobs, Flexible scope) ||| Centrally manage at scale (AWS Organizations, Managed & custom data detections) ||| Automate and take actions (Detailed findings, Management APIs) ||| Maice-Data identifiers: Fully managed sensitive data types, maintains a growing list of sensitive data types that include common personally identifiable information (PII) and other sensitive data types as defined by data privacy regulations, such as GDPR, PCI-DSS, CCPA and HIPAA. Data Identifiers(Financial, personal, national, medical etc.) ||| Maice-Supported File Format: analyzes data in an S3 bucket, it performs a deep inspection that factors the file or storage format for the data. Macie can analyze and detect sensitive data in many different formats, including commonly used compression and archive formats (Big Data - Apache Avro object containers and Apache Parquet files Compression or archive - .gz, .gzip, .tar, .zip Document - .doc, .docx, .pdf, .xls, .xlsx Text - .csv, .htm, .html, .json, .tsv, .txt, .xml, and others (depending on the type of non-binary text file)).
5. Incident Response → During an incident, containing the event and returning to a known good state are important elements of a response plan. AWS provides the following tools to automate aspects of this best practice. Amazon Detective, Analysis and visualization of security data to get to the root cause of potential security issues quickly ||| Amazon EventBridge, Serverless event bus that makes it easier to build event-driven applications to scale your programmed, automated response to incidents ||| AWS Backup, Centrally manage and automate backups across AWS services to simplify data protection at scale ||| AWS Security Hub, Out-of-the-box integrations with ticketing, chat, SIEM, SOAR, threat investigation, incident management, and GRC tools to support your security operations workflows ||| AWS Elastic Disaster Recovery, Fast, automated, cost-effective disaster recovery
Automated Incident Response Example → 1. Potential instance compromise 2. GuardDuty generates finding 3. Finding Triggers EventBridge Event 4. Event triggers Lambda function 5. Isolates instance and sends notification to security team 6. Security team focus on root cause analysis
6. Compliance → AWS Artifact ||| AWS Audit Manager
- Authentication: who is who vs Authorization: who can access what

AMAZON API GATEWAY

• What is API: In applications, an API simplifies programming by abstracting the underlying implementation and only exposing objects or actions the developer needs. Web-based companies and services offer APIs for developers to use(Social Networks – Facebook, Twitter, etc — Payment Processing – Amazon Pay, PayPal, etc).
API Gateway: Amazon API Gateway is a fully managed (serverless) service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. Endpoint types are Edge-Optimized, Regional, Private. API Gateway can be used for Lambda functions, public endpoints on EC2, private endpoints in VPC, AWS Direct Connect for on-premises. API Gateway also has Cache property. API has two types → RESTful APIs or WebSocket APIs.
WebSocket APIs: Serverless WebSocket, 2 way communication channel, long-lived communication, stateful.
RESTful APIs: Request / Response, HTTP Methods like GET, POST, etc., short-lived communication, stateless. There are two flavors/versions of RESTful APIs. REST API (v1) and HTTP API (v2). REST API is more feature rich, feature parity in HTTP API will take some time. HTTP API is built from the ground up: Faster(up to 60%), lower cost(up to 71%), easier to use.
Endpoint Types (Edge-Optimized | Regional | Private): Edge-Optimized (REST): Utilize CloudFront to reduce TLS connection overhead (reduces roundtrip time).Designed for a globally distributed set of clients. Regional (both): Recommended API type for general use cases. Designed for building APIs for clients in the same region. Private(REST): Only accessible from within VPC (and networks connected to VPC). Designed for building APIs used internally or by private microservices.
• Integrations: API Gateway can be connected to Lambda functions, Public HTTP endpoints (like EC2 endpoints), any other AWS service, Endpoints in VPC, AWS Direct Connect for connecting on-premises.
** For private integrations - VPC Link is required. API Gateway is connected to Network Load Balancer (NLB), Application Load Balancer (ALB)(HTTP property) and AWS Cloud Map(HTTP property) through VPC Link
The Request Cycle (WAF → Authenticate → Authorizer → Authorize → Integration): WAF Phase only appears when an AWS WAF web access control list (ACL) is configured for enhanced security. During this phase, AWS WAF rules are evaluated and a decision is made on whether to continue or cancel the request. Authenticate Phase only present when AWS Identity and Access Management (IAM) authorizers are used. During this phase, the credentials of the signed request are verified. Access is granted or denied based on the client's right to assume the access role. Authorizer Phase Only present when a Lambda, JWT, or Amazon Cognito authorizer is used. During this phase, the authorizer logic is processed to verify the user's right to access the resource. Authorize Phase only present when a Lambda or IAM authorizer is used. During this phase, the results from the authenticate and authorizer phase are evaluated and applied.
Client Certificates (REST API): Generate client-side SSL certificate using the API Gateway. It allows backend to verify requests coming from API Gateway using public key. Expires after 365 days.
• Protecting APIs:
Types of authorization → Resource Policy(REST), Mutual TLS, WAF(REST), Cognito User Pools(REST), JWT(HTTP), IAM, Lambda Authorizer ||| Resource Policy(REST): AWS Cloud Account and Corporate Data Center (IP Address) allowance. Mutual TLS: Requires two-way authentication between client and server. Clients must present X.509 certificates to verify identity. Often used in Internet of Things (IoT) and business to business applications. Only supported on Custom Domains. Web Application Firewall(WAF) (REST): Secure your app through WAF Rules like blocking an IP, or an area etc. JWT(HTTP)/Cognito(REST): OAuth2 compliant (part of OpenID Connect - OIDC). Allows or denies access based on token validity and optional scopes. Any required scopes for the route are validated in the token. Auth0 can be used. IAM Authorizer: Clients must use Signature Version 4 to sign their requests with AWS credentials. Authorization token is decoded. User is verified against the Identity & Access Management (IAM) service. Users must have execute-api access on the route to proceed. Lambda Authorizer: Your custom logic to validate the request. 2 payload options. Payload 1(HTTP): must return an IAM policy that allows or denies access to your API route. Payload 2: can return IAM policy or Boolean. Authorization can be cached.
Throttling and usage plans: Protect backend systems. Prevents one customer from consuming all your backend system's capacity. Let's you decide how to allocate capacity among your API consumers with quotas and request rates. Professional plan users:10 RPS, up to 100 calls / day. Enterprise plan users: 500 RPS, no limit on calls / day. Four levels of Throttling: Per Client & Method(REST - Requires API Keys and Usage Plans), Per Client(REST - Requires API Keys and Usage Plans), Per Method, Per account.
Usage Plans and API Keys: API Keys: Alphanumeric string values that you distribute to clients (per user/client). Generated by API Gateway or you can import from a CSV file. Use API keys together with Usage Plans or Lambda authorizers to control access to your APIs. Usage Plans: Specifies who can access API stages and methods. How much and how fast they can access the resource(rate limit, quota limit). Uses API keys to identify API clients.
• Validation & Transformation: Validation: Required request parameters in the URI, query string, and headers of an incoming request are included and non-blank. The applicable request payload adheres to the configured JSON schema request model of the method. Transformation: Mapping
• Stages and Versioning: API Gateway enables you to set stage variables, allowing the same API to point to different backends.
Your APIs are versioned and can be rolled back. APIs are deployed to staging environments. You choose what to name them. For example, these environments: Dev (e.g., example.com/dev), Beta (e.g., example.com/beta), Prod …
!!! REST API allows Canary Release Deployment. REST API support versioning allowing to easily roll back to a previous API version and deploy it.
• Custom Domain Names: Supports HTTP, REST, and WebSocket APIs. Without custom domains, endpoints would be like https://12345.execute-api.us-east-1.amazonaws.com/prod/catalog where 12345 indicates API ID, us-east-1 indicates region, prod indicates stage, catalog indicates resources. With custom domains, your endpoints would be like https://mydomain.com/api-one/products or https://mydomain.com/api-two/v2/catalog. The "api-one" and "api-two/v2" are base path mapping. Custom domains support multiple APIs through multi level base path mapping. SSL Certs manage through ACM.
• Observability: Built-in (REST, HTTP, WebSocket) and Custom metrics. REST: API Calls Count, Latency, 4XXs, 5XXs, Integration Latency, Cache Hit Count, Cache Miss Count. HTTP: API Calls Count, Latency, 4XXs, 5XXs, Integration Latency, DataProcessed. WebSocket: Connect Count, Message Count, Integration Error, Client Error, Execution Error, Integration Latency. Custom Metric: create via Metric Filter out of logs.
Logging(Execution Logs and Access Logs): Execution Logs: Two levels of logging → Error and INFO. Optionally log method request/body content. Set globally in stage, or override per method. Access Logs: Customizable format for machine parsable logs. CloudWatch Logs OR Kinesis Firehose
Enhanced Observability Variables: $context.<phase>.status, $context.<phase>.latency, $context.<phase>.error where <phase> are the phases in the request cycle like WAF, Authenticate, Authorizer, Authorize or Integration. These variables are available for each type of API for Authorizer and Integration Phase, but only available for REST API type for WAF, Authenticate, Authorize.
Troubleshooting Latency Example ⇒ Example of an IAM Authorized Endpoint with an AWS WAF in front of it→Total Latency is 48 ms. WAF(7ms), Authenticate(1ms), Integration(16ms), then API Gateway runs in 48-(7+1+16)=24 ms
!!! Tracing- X-Ray is available for REST API. AWS X-Ray provides trace information about any received responses and calls that an instrumented application makes, including to the following: Downstream AWS resources, microservices, databases, web APIs.
• Other Features: Caching(REST), OpenAPI for migrating APIs. Caching: Effective for quick responses and minimizes load to your backend. Leverage cache keys to optimize response (Path, headers, query strings). It can be set up per Stage or Method. OpenAPI for migrating APIs: Migrating third party API or Amazon API Gateway REST or HTTP APIs using OpenAPI JSON/YAML file.
• Pricing: HTTP API requests: $1.11 / Million (1M)* ||| REST API requests : $3.5 / Million (1M) *||| API cache: $0.2/hour for 0.5GB → $3.80/hour for 237 GB * ||| WebSocket requests: $1.14 / Million messages * (1M) ||| WebSocket connection minutes: $0.285 / Million minutes (750,000) ⇒ (* Tiered pricing)
• Best Practices: Use HTTP APIs if existing features are sufficient. Create an API per team/microservice and unify them with Custom Domain Name. The console is for experimenting, use Infrastructure as Code: SAM, CloudFormation, TerraForm etc.
3 Tier Apps: Frontend → Middleware → Backend + DB (Private area) ||| indirect API calls more secure (sent through middleware), direct API calls from frontend to backend can be hacked
Security Group is part of WAF, but WAF is a Firewall used outside only. ||| Transformation mapping can fix API calls of trusted users.

CI/CD

Deployment pipeline: Build->Test->Release->Monitor->Plan->Build ||| Important features: Lead time, Deploy frequency, Mean time between failure(MTBF), Mean time to recover(MTTR)
Development: Source code->Developer workstation->Dev AWS acc.->Code review = Works on *smallest* thing possible, explicitly defines *dependencies*(including those needed for testing), published *changes* for review, peer review *new* code, *style* checkers (all of these should be FAST)
Example pipeline (Development Phase): Code review->Dependencies->Package and build = Build and runs *unit testing*, Bundles code and run-time dependencies into a *combined artifact*, providence of *dependencies* is tracked (all should be THOROUGH) ||| (Mid Stages) : Beta = run destructive or any other testing, smoke testing, wild west -> Gamma = Isolated, "prod-like", integration testing full "end-to-end", load testing, security testing ||| (Production) : Start small and fan out, deploy in waves, give each wave time to "bake", always monitor (looking for any reason to FAIL, roll back on test failure)
Why CI/CD: Accelerate delivery of new and *high-quality* services, reduce *impact* of changes, gain *insight* across resources and applications, *protect* customers and the business
How CI/CD affects organizations: Low(weekly-monthly): failure rate(fr)=64% || Medium(weekly-monthly): fr=15% ||| High(Daily-weekly): fr=10% ||| Elite(on demand): fr=5% (Only 18% of organizations use elite)
Amazon Code Catalyst: Phases: Author->Source/Artifact->Build&Test->Deploy->Monitor ||| Managed, all-in-one, integrated, security-focused, flexible ||| Benefits: Accelerate project setup, automate daily workflows, automate, environments, fluidly collaborate
AUTHOR: Amazon CodeWhisperer, Cloud9, IDE Toolkits, SDKs ||| CodeWhisperer(Çakma copilot): Generate code suggestions in real time, scan code for hard-to-find vulnerabilities, flag code that resembles open-source training data or filter by default, provides 27% more complete task rate and 57% faster complete compare to non-users||
SOURCE/ARTIFACT: AWS CodeCommit, AWS CodeArtifact, Amazon ECR & ECR Public, Amazon CodeGuru ||| CodeCommit: Secure, highly scalable, managed source control service that hosts *private git repositories*, works with *existing git tools*, no *hardware* to provision and scale, highly *available* and *durable* ||| CodeArtifact: Fully managed *artifact repository* service, supports *npm, pip, yarn, twine, Maven, NuGet* and *SwiftPM*, access packages *within VPC*, no upfront *license cost* ||| Elastic Container Registry: Fully managed *Docker image registry*, managed and scalable *infrastructure*, security with *encrypted images* and *vulnerability scans*, container images, Helm charts, OCI artifacts ||| CodeGuru: Onboarding= *cloud-based API* allows *you to integrate flexibly* at various stages of the CI/CD pipeline, also *scales* with your workload so you dont need to *manage capacity* | Triaging/Prioritization= *Low false positives rate* improves your *operational efficiency* as you have fewer findings to review | Remediation= *Fewer false positives* means you get fewer tickets, your dev teams can now allocate their limited resources to *actual vulnerabilities* | Closure= Bug tracking automatically *detects* and *updates bug status*, providing current and accurate reporting on open vulnerabilities and *closure rate*
BUILD&TEST: AWS CodeBuild ||| CodeBuild: Fully managed build service that *compiles source code*, runs *tests*, and produces *software packages* | *Scales continuously* and processes multiple builds *concurrently* | *No build servers* to manage | Pay by the minute, only for the compute resources you use | monitor builds through *Amazon CloudWatch Events*
DEPLOY: AWS CodeDeploy ||| CodeDeploy: *Automates code deployments* to any instance and Lambda | Handles the *complexity* of updating your applications | *Avoids downtime* during application deployment | *Rolls back* automatically if failure detected | Deploys to *Amazon EC2, Lambda or on-premises* servers
Deployment Strategies (Definition): A deployment strategy is a way to *change* or *upgrade* an application. The aim is to make the change *without downtime* in a way that the users *barely notice* the changes.
In-place deployments:
All at once: Version A is terminated, then version B is rolled out || An in-place deployment is a deployment strategy that updates the application version without replacing any infrastructure components. In an in-place deployment, the previous version of the application on each compute resource is stopped, the latest application is installed, and the new version of the application is started and validated. This allows application deployments to proceed with minimal disturbance to underlying infrastructure. An in-place deployment allows you to deploy your application without creating new infrastructure; however, the availability of your application can be affected during these deployments. This approach also minimizes infrastructure costs and management overhead associated with creating new resources. Refer to Overview of an in-place deployment for more details on using in-place deployment strategies with CodeDeploy.|||
Rolling update: Version B is slowly rolled out in terms of microservices and replaces version A || A rolling deployment is a deployment strategy that slowly replaces previous versions of an application with new versions of an application by completely replacing the infrastructure on which the application is running. For example, in a rolling deployment in Amazon ECS, containers running previous versions of the application will be replaced one-by-one with containers running new versions of the application. A rolling deployment is generally faster than a blue/green deployment; however, unlike a blue/green deployment, in a rolling deployment there is no environment isolation between the old and new application versions. This allows rolling deployments to complete more quickly, but also increases risks and complicates the process of rollback if a deployment fails. Rolling deployment strategies can be used with most deployment solutions. Refer to CloudFormation Update Policies for more information on rolling deployments with CloudFormation; Rolling Updates with Amazon ECS for more details on rolling deployments with Amazon ECS; Elastic Beanstalk Rolling Environment Configuration Updates for more details on rolling deployments with Elastic Beanstalk; and Using a Rolling Deployment in AWS OpsWorks for more details on rolling deployments with OpsWorks.
Blue/green deployments: Version B is released alongside A, and then the traffic is switched to version B. || A blue/green deployment is a deployment strategy in which you create two separate, but identical environments. One environment (blue) is running the current application version and one environment (green) is running the new application version. Using a blue/green deployment strategy increases application availability and reduces deployment risk by simplifying the rollback process if a deployment fails. Once testing has been completed on the green environment, live application traffic is directed to the green environment and the blue environment is deprecated. A number of AWS deployment services support blue/green deployment strategies including Elastic Beanstalk, OpsWorks, CloudFormation, CodeDeploy, and Amazon ECS. Refer to Blue/Green Deployments on AWS for more details and strategies for implementing blue/green deployment processes for your application.
Canary deployments: The *slow rollout* in terms of users (by increasing the percentage of users) of a new version of an existing application.|| The purpose of a canary deployment is to reduce the risk of deploying a new version that impacts the workload. The method will incrementally deploy the new version, making it visible to new users in a slow fashion. As you gain confidence in the deployment, you will deploy it to replace the current version in its entirety. Steps: Use a router or load balancer that allows you to send a small percentage of users to the new version. Use a dimension on your KPIs to indicate which version is reporting the metrics. Use the metric to measure the success of the deployment; this indicates whether the deployment should continue or roll back. Increase the load on the new version until either all users are on the new version or you have fully rolled back.
AWS CodePipeline: Continuous delivery service for *fast* and *reliable* application updates | *Model* and *visualize* your software release process | *Builds, tests* and *deploys* your code every time there is a *code change* | Integrates with *third-party tools* and AWS
MONITOR: AWS X-Ray, Amazon CloudWatch, Amazon DevOps Guru ||| DevOps Guru: ML-powered cloud operations service to improve application availability | Select coverage: Select the entire account or specific CFN stacks | Generate insights: Guru will start analyzing metrics/logs to generate insights | Integrate with your workflow: Integrate with OpsCenter SNS/ServiceLens
Model: AWS CloudFormation, Cloud Development Kit (AWS CDK, CDK8s, CDK-terraform), AWS Serverless Application Model(SAM), AWS Amplify
Infrastructure as Code (IaC): What?: Writing *code* to create, configure and deploy *infrastructure* components | Infrastructure includes: *networking, compute, databases, security, management tools* etc. ||| Why?: Makes infrastructure changes *repeatable and predictable* | *Documents* your infrastructure | Automates the *provisioning* process | Eliminates *configuration drift* thorough automation
IaC with AWS CloudFormation: Code your template->Upload, test, review changes->Execution of changes creates a stack->Manage stacks and stack sets
IaC with AWS Cloud Development Kit(CDK): cdk init -> npm run build -> cdk synth -> cdk diff -> cdk deploy

GENERATIVE AI

Generative AI is powered by foundation models. Pretrained on vast amounts of unstructured data. Contains a large number of parameters that make them capable of learning complex concepts. Can be applied in a wide range of contexts. Customize FMs using your data for domain specific tasks. |||
Enhance Customer Experiences: Chatbots, Virtual Assistants, Conversation Analytics, Personalization || Boost employee productivity & creativity: Conversational Search, Summarization, Content Creation, Code Generation, Data To Insights || Optimize business processes: Document Processing, Data Augmentation, Cybersecurity, Process Optimization ||| Healthcare & Life Sciences: Ambient digital scribe, Medical imaging, Drug discovery, Enhance clinical trials, Research reporting || Industrial & Manufacturing: Product design, Operational efficiency, Maintenance Assistants, Supply chain optimization, Equipment diagnostics || Financial Services: Portfolio management, Financial documentation, Intelligent advisory, Fraud detection, Compliance assistant || Retail: Pricing optimization, Virtual try-ons review, Marketing Optimization, Product descriptions, Pers. Recommendations || Media & Entertainment: HQ content at scale, Enrich broadcast content, Automated content tagging, Optimize subscriber exper., Automated highlights gen. |||
❖ Generative AI Stack: INFRASTRUCTURE FOR FM TRAINING AND INFERENCE
GPUs || Trainium || Inferentia || SageMaker || UltraClusters || EFA || EC2 Capacity Blocks || Nitro || Neuron
CG1 NVIDIA Tesla M2050 "Fermi" GPUs || G2 NVIDIA GRID GK104 "Kepler" K80 || P2 NVIDIA K80 GPUs || G3 NVIDIA Tesla M60 GPUs || P3 NVIDIA V100 Tensor Core GPUs || G4 NVIDIA T4 Tensor Core GPUs || P4 NVIDIA A100 Tensor Core GPUs || G5 NVIDIA A10G Tensor Core GPUs || G5g NVIDIA T4G Tensor Core GPUs || P5 NVIDIA H100 Tensor Core GPUs||| 4x Higher Throughput || 10x Lower Latency
Amazon Sagemaker: Build, train, and deploy ML models at scale || Automatic model fine-tuning & distributed training || Flexible model deployment options || Tools for ML operations || Built-in features for responsible AI || Support for Hugging Face models and Hugging Face AWS Deep Learning Containers
❖ Generative AI Stack: TOOLS TO BUILD WITH LLMs AND OTHER FMs |||
Amazon Bedrock (Guardrails || Agents || Customization Capabilities)
Amazon Bedrock: The easiest way to build and scale generative AI applications with LLMs and other FMs. || Choice of industry-leading FMs from AI21 Labs (JURASSIC-2), Amazon (AMAZON TITAN), Anthropic (CLAUDE - Turkish support), Cohere (COMMAND + EMBED), Meta (LLAMA 2), Mistral AI (Mistral7B, Mixtral 8x7B), and Stability AI (STABLE DIFFUSION XL) || Customize FMs using your organization's data ||Enterprise-grade security and privacy || Simplifies Choice, Customization, Integration|| None of the customer's data is used to train the underlying model. || All data is encrypted in transit and at rest. || Data used to customize models remains within your VPC. || Support for standards, including GDPR & HIPAA || Amazon Bedrock simplifies: Choice, Customization, Integration.
Bedrock Recently added security capabilities : CloudWatch integration (Track usage metrics and build customized dashboards), CloudTrail integration (Monitor API activity and troubleshoot issues), SOC (System and Organization Controls) 1, 2, 3 compliance (It examines and audits service organizations to ensure that controls and processes are in place to protect client data that they have access to)|||

- Foundation Model ⇒ Customizations (Fine tuning, Retrieval Augmented Generation (RAG), Continued Pre-training) ⇒ Your Data ⇒ FM ⇒… (loop)
- Agents for Amazon Bedrock: Execute multi-step tasks across company systems and data sources || Enables generative AI applications to take action in just a few clicks || Breaks down and orchestrates tasks and executes API calls on your behalf || Securely accesses and retrieves company data
- Guardrails for Amazon Bedrock : Safeguard your generative AI applications with your responsible AI policies || Easily configure harmful content filtering based on your responsible AI policies || Apply Guardrails to any FM or agent || Redact PII (Personally Identifiable Information) in FM responses (coming soon)

❖ Generative AI Stack: APPLICATIONS THAT LEVERAGE LLMs AND OTHER FMs
Amazon Q || Amazon Q in Amazon Quicksight || Amazon Q in Amazon Connect || Amazon CodeWhisperer

- Amazon CodeWhisperer: AI-powered code suggestions in the IDE and the command line.|| Customization capability: Better and more relevant code suggestions || Your data is never used to train the underlying model, only Amazon code and publicly available code are used || Onboard developers faster ||| Can customize/personalize CodeWhisperer by finetuning it if you have 100MB code
- Amazon Q: A generative AI-powered assistant for work that is tailored to your business. || Provides interactive answers, solves problems, generates content, and takes action. || Understands your company information, code, and systems. || Personalizes interactions based on your role and permissions. || Built to be secure and private. || Trained on 17 years of AWS knowledge. || Assists you everywhere you work with AWS – in the console, IDE, and documentation || Converses with you to explore new AWS capabilities, learn unfamiliar technologies, and architect solutions. || Works with you to troubleshoot, build new features, and upgrade languages on AWS|||

- Amazon Q knocks down obstacles for developers : Find Information, Planning (In The Console & In The Ide), Understand A New Codebase (Chat In The Ide), Write Code (CodeWhisperer/feature Development), Manage Resources (In The Command Line), Upgrade Old Code (Code Transformati On), Debug & Optimize Code (Chat In The Ide), Troubleshoot Errors (In The Console), Generate Unit Tests (In The Ide), Find Security Vulnerabilities (Code Scanning), Remediate Code (Automated Code Remediation). |||
- Amazon Q as business expert: Delivers quick, accurate, and relevant answers to your business questions, securely and privately || Connects to over 40 popular data sources including S3, Salesforce, Google Drive, Microsoft 365, ServiceNow, Gmail, Slack, Atlassian, and Zendesk || Respects existing access controls - only returns info you're authorized to see based on your role
- Amazon Q in Amazon QuickSight: Generative dashboard authoring || Visually compelling data stories || Reimagined Q&A experience|||
- Amazon Q is : Your AWS Expert, Business Expert || Amazon Q in Amazon Quicksight is : Your Business Intelligence Expert || Amazon Q in Amazon Connect is : Your Contact Center Expert

AWS GLUE DATABREW

AWS Glue is a fully managed serverless environment where you can extract, transform, and load (ETL) data at scale. With AWS Glue, you can categorize data, clean it, enrich it, and move it reliably across various data stores and streams in a cost-effective manner. (In the scope of Analytics)
-The problem: Data grows exponentially| From new resources| Increasingly diverse| Used by people with diverse skillsets| Accessed by many applications ||| Data preparation requires huge amount o time (80%). | It is time consuming to extract, clean, normalize, and load data at scale & needs right tools to be integrated. | Costly user licenses & siloed tools causing rework & moving large data out of VPCs. | Hard to operationalize and build repeatable workflows & needs a lot of code-based heavy-lifting for it to work at scale.
AWS Glue DataBrew – DataBrew is a no-code data preparation tool that you can use to visually explore, clean, and transform data. You can choose from more than 250 prebuilt transformations to automate data preparation tasks without writing any code (Serverless). | Built for data analysts & data scientists. ||| Data preparation made easy: Understands data quality (patterns & anomalies)| Cleans and normalizes data | Visually maps data lineage (understand steps that data has been through) | Automates at scale (prepared for new data).
-Functionality: Connect to data sources (S3, Redshift, RDS Aurora, Glue Data Catalog)| 250+ built-in transformations | For users of all technical levels (visuals) | Advanced data profiling(to get the statistics) | Visual data lineage | Integrate with data pipelines | serverless usage at scale.
-ETL: Develop complex data transformations (Glue Studio - Visual ETL IDE for non-Spark experts & Dev Endpoints: interactive development for Glue jobs for Spark Experts) | Streaming: Process IoT streams in real time (Glue Studio - Develop streaming jobs visually & Glue Streaming - processes real time data stream from IoT devices) | Data Replication: Replicate data across purpose built data stores (Glue Elastic View- replicate data across multiple data stores without code) | Data prep: Transform data without code (Glue Data Brew build data without coding).
- Use Cases: (One-time data analysis for business reporting)- Local file ⇒ S3 & data catalog data sources ⇒ AWS Glue DataBrew ⇒ S3 output bucket ⇒ Amazon QuickSight||| (Set up data quality rules with AWS Lambda) Recurring raw data feed ⇒S3 (⇒AWS Lambda) ⇔ AWS Glue DataBrew⇒ Amazon EventBridge⇒ AWS Lambda ⇒Amazon SNS ⇒Email notification ||| Data preprocessing for ML ||| Orchestrating data preparation in workflows. ||| Highlighted features: Enrich data with unions and joins | update schema | generate data profiles | feature engineering-plugin in Notebooks | remove outliers | analyze aggregated data |

AMAZON CLOUDWATCH

Amazon CloudWatch Benefits: Observability on a single platform applications and infrastructure | Easiest way to collect metrics in AWS and on-premises | Improve operational performance and resource optimization | Get operational visibility and insight | Derive actionable insights from los ||| How it works: Collect⇒Monitor⇒Act⇒Analyze | Collect: Metric and logs from all your AWS resources, applications and services that run on AWS and on-premises servers | Monitor: Visualize applications and infrastructure with CloudWatch dashboards; correlate logs and metrics side by side to troubleshoot and set alerts with CloudWatch Alarms | Act: Automate response to operational changes with CloudWatch Events and Auto Scaling | Analyze: Up to 1-second metrics, extended data retention(15 months) and real-time analysis with CloudWatch Metric Math ||| Used For: Application Monitoring, System-wide Visibility, Resource Optimization, Unified Operational Health
Collect: Collect logs from; EC2 instances, on-premises servers, VPC Flow Logs, CloudTrail, Lambda, other AWS Services ||| Log data can be stored and accessed indefinitely in highly durable, low-cost storage so you dont have to worry about filling up hard drives. ||| Built-in metrics: Collecting metrics is time consuming. CloudWatch allows you to collect default metrics from more than 70 AWS services, such as; EC2, DynamoDB, S3, ECS, Lambda, API Gateway | No action is required on your part. For example, EC2 instances automatically publish CPU utilization, data transfer and disk usage metrics to help you understand the changes in state. ||| Custom metrics: Collect custom metrics from your own applications to monitor operational performance ,troubleshoot issues, and spot trends. User activity is an example of a custom metrics you can collect and monitor over a period of time. Publish metrics using the AWS CLI or an API | Standard resolution, with a one-minute granularity | High resolution, with a granularity of one second | Aggregate data before you publish to CloudWatch | StatsD and collectd support via CloudWatch Agent ||| Collect and aggregate container metrics and logs: Use CloudWatch Container Insights to collect, aggregate and summarize metrics and logs from your containerized applications and microservices. Collects metrics from each container; CPU, Memory, Disk, Network | Automatically generated dashboards | Set alarms on metrics
Monitor: Amazon CloudWatch dashboards enable you to create re-usable graphs and visualize your cloud resources and applications in a unified view. | A single view for selected metrics and alarms | Multiple AWS accounts and multiple Regions | An operational playbook | A common view of critical resource and application measurements that can be shared ||| High resolution alarms: Amazon CloudWatch alarms allow you to set a threshold on metrics and trigger an action. | Watch a single metric or the result of a math expression | Add alarms to dashboards to visualize them | Perform actions based on the value of metrics (Send a notification to an SNS topic, Auto Scaling action, EC2 Action (Stop, Terminate, Reboot or Recover)) ||| Logs and metric correlation: Amazon CloudWatch also makes it easy to correlate metrics and logs. | Manage logs and metrics in a single platform | Use metric filters to convert logs to metrics ||| Application insights: Easily monitor .NET and SQL Server applications, so you can get visibility into the health of such applications. | Automatic Set Up of Monitors for Application Resources | Problem Detection and Notification | Automatic dashboards | Insights that point to potential root causes ||| Anomaly Detection: When you enable anomaly detection for a metric, CloudWatch applies machine learning algorithms to the metric's past data to create a model of the metric's expected values. | Create alarms that auto-adjust thresholds based on natural metric patterns | Alarm when the metric value is above or below the band, or both | Visualize metrics with anomaly detection bands on dashboards ||| ServiceLens: You can use Amazon CloudWatch ServiceLens to visualize and analyze the health, performance, and availability of your applications in a single place. | Integrates CloudWatch with AWS X-Ray to provide an end-to-end view of your application | A service map displays your service endpoints and resources as "nodes" and highlights the traffic, latency, and errors for each node and its connections | You can choose a node to see detailed insights about the correlated metrics, logs, and traces associated with that part of the service ||| Synthetics: Run tests on your endpoints every minute, 24x7, and alerts you as soon as your application endpoints don't behave as expected. | • View of your customers' experiences • Configurable scripts • Run once • Run on a schedule • Check availability and latency • Store load time data • Store screenshots
Act: Auto Scaling: Auto Scaling helps you automate capacity and resource planning. | • Set a threshold to alarm on a key metric and trigger an automated Auto Scaling action • For example, you could set up an Auto Scaling workflow based on queue depth • Configure policies to scale in or scale out • Allows you to set 1 scaling policy and trigger with multiple alarms ||| Response automation: CloudWatch Events provides a near real-time stream of system events that describe changes to your AWS resources. | • Respond quickly • Take corrective action | Write rules to indicate which events are of interest to your application and what automated actions to take when a rule matches an event. | • Invoke a Lambda Function • Notify an SNS Topic • Create an Ops Item in Systems Manager
Analyze: Granular data and extended retention: Amazon CloudWatch allows you to monitor trends and seasonality with 15 months of metric data (storage and retention). • Historical analysis to fine-tune resource utilization • Collect metrics with a granularity of 1 second • Granular real-time data enables better visualization • Spot and monitor trends to optimize applications. ||| Custom operations on metrics: Metric Math enables you to perform calculations across multiple metrics for real-time analysis. • Visualize computed metrics in the Console • Add them to CloudWatch dashboards • Retrieve them using the GetMetricData API action. Metric Math supports arithmetic operations such as +, -, /, *, and mathematical functions such as Sum, Average, Min, Max, and Standard Deviation. Using AWS Lambda metrics as an example, you could divide the Errors metric by the Invocations metric to get an error rate. ||| Log analytics: CloudWatch Logs Insights enables you to drive actionable intelligence from your logs to address operational issues without needing to provision servers or manage software. • You can instantly begin writing queries with aggregations, filters, and regular expressions • In addition, you can: • Visualize timeseries data • Drill down into individual log events • Export query results to CloudWatch Dashboards • You only pay for the queries you run. ||| Contributor Insights: Analyzes time-series data to provide a view of the top contributors influencing system performance. • Runs continuously without needing user intervention • Understand who or what is impacting your system • Evaluate patterns in structured log events • Display on CloudWatch dashboards • Add to CloudWatch alarms.

AMAZON FOR GAMES

AWS for Games solution areas and use cases: Build, Run, Grow. ⇒ BUILD: Cloud game development (Workstations, Build pipelines, Version control, 3D world building) RUN: Game servers(Hosting session- based games, Global game infrastructure) ||| Game security (Defend against DDOS attacks, Protect against data breaches) GROW: LiveOps (Game Services for LiveOps) ||| Game analytics (Centralized game analytics) ||| AI & ML (Community health/toxicity, Smart acquisition and retention)
-Aws For Games Provides Solutions For Games Customers Wherever They Are In The Builder–Buyer Continuum: Native AWS services, Game-specific AWS services, AWS solutions & Quick Starts, AWS ProServe & GSIs, AWS Partner Solutions & AWS Marketplace
- AWS for Games solution library: AWS SERVICES (Purpose-built cloud products) ||| AWS SOLUTIONS (Ready-to-deploy solutions assembling AWS Services, code, and configurations) ||| PARTNER SOLUTIONS (Software, SaaS, of managed services from AWS Partners) ||| GUIDANCE (Prescriptive architectural diagrams, sample code, and technical support)
•••Amazon and AWS Game Services: Amazon GameLift(Dedicated game server hosting solution for multiplayer games) ||| Open 3D Engine (O3DE)(AAA-capable, cross-platform, open-source game engine available under an Apache 2.0 license) ||| Amazon Luna (Game streaming service that allows you to play games on devices you already own) ||| GameLift FlexMatch (Matchmaking for multi-player games) ||| Amazon Twitch ||| Amazon Prime Gaming (Activate 150M+ Prime members in your game)
!!! Amazon GameLift is for multiple player games but you can also create single player games. (This is from lecture notes)
•• Amazon GameLift: Deploy, operate, and scale dedicated, low-cost servers for session-based, multiplayer games. Low-cost game servers(Scale game servers automatically based on player demand and leverage low cost Spot Instances for short-lived game sessions). Low latency (Deploy a single fleet globally and leverage built-in latency-based matchmaking). High availability (fully managed game servers hosted across AWS Regions on multiple Availability Zone, as well as Local Zones, providing availability and scalability). Flexibility (Select the features you need. Use the built-in matchmaking or build your own. Select between fully managed and self-managed 11 hosting).
GameLift Overview: Multiplayer game server entry points → Fully-managed GameLift (Fully managed, vertically integrated game server stack for dedicated game servers) ||| GameLift Realtime (Lightweight, ready- to-go game servers well-suited for mobile games. It can understand your location, servers.) ||| GameLift FleetIQ (Server management layer to help developers optimize costs on Amazon EC2 with Spot. It keeps load same for the all servers) ||| GameLift Anywhere (Session management layer to help developers looking for compute flexibility - Beta. You can understand if there is a cheater or there is an attack to a server and then a close related user). |||||||| Select the features you need → 31 Regions (Fully managed game servers hosted across AWS Regions on multiple Availability and Local Zones ensuring availability and scalability) ||| Instances (Scale automatically based on player demand with 60+ instance types supported on Windows or Linux) ||| SDKs(C++ and C# SDKs to develop GameLift-enabled multiplayer game servers, game clients and game services) ||| Game Engines (Support for Unity and Unreal Engines with regular SDK updates for engine versions, binary plugins, and CloudFormation integrations)
Amazon GameLift Infrastructure Management: • Auto-scaling • High-availability (Multi-AZ) • Automatic cross-region failover
Amazon GameLift Session Management: • Player & Game sessions • Worldwide placement • Matchmaking (FlexMatch)
!!! Two service options for hosting:

- Fully-managed Amazon GameLift : Fully managed game server hosting || Upload your build once, deploy globally || Managed latency-based session placement || Supports on-demand and spot fleets for cost optimization with spot viability algorithm || Supports C++, C#, Unity & Unreal || Native integration with GameLift FlexMatch for matchmaking
- Amazon GameLift FleetIQ : More Flexible: Hosted on EC2 on your AWS account || Simple API layer for game session management || Use your existing tools and software for deploying and running game servers || Access GameLift Spot viability algorithm independent of other GameLift features || AWS SDK on the server side: use the programming language of your choice

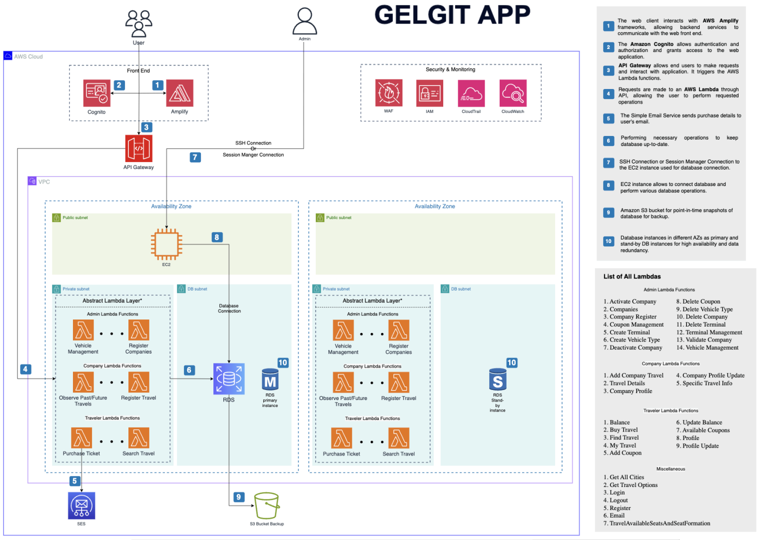
••• End-to-End High Level Integration Flow; Game Clients → Game Backend Services (client communicate with your game backend) → GameLift FlexMatch (use the free build-in matchmaking based on your needs) → GameLift Queue (Queues route traffic to the lowest latency region) → Fleet (A single Fleet can span across up to 23 different AWS Regions)
Amazon GameLift Components : Builds || Fleets || Aliases || Queues || Clients || FlexMatch

- Builds:Upload via AWS CLI or SDK | Store multiple builds | Define Install script for custom instance configuration | Supported Operating Systems:• Amazon Linux • Windows | Server SDKs:• C++ • C# • Unreal Plugin (SDK) • Unity Plugin (Fully-featured) ||| Build Dashboard : Access key information about builds stored in Amazon GameLift: • Status • Version • OS • Size • Number of fleets using this build
- Fleets: Fleets represent the deployed state of your game build. They enable you to define how your build should be hosted within Amazon GameLift including: • Instance Type • Home Region and additional Regions(multi-region Fleets) • Scaling Policies • Capacity Limits • Number of processes per instance (up to 50) ||| Fleet Metrics: • Game/Player Sessions • Number of Instances • Game Server Processes • Instance Performance • Scaling Limits ||| Target tracking auto-scaling : Amazon GameLift can automatically maintain a defined level of session availability for your game servers.GameLift determines the exact number of server instances to add or remove as it tracks toward a target buffer percentage. When player demand for a game is rising, new server instances are automatically provisioned. When demand subsides, server instances are automatically scaled down (with protection on running game sessions, when enabled on Fleet level) ||| Fleet Events : See events that take place in the fleet at the game session or instance level. Events reported include: • Scaling • Information • Warnings • Errors • Crashes ||| Fleet – Game Sessions: Track running game sessions running in a fleet in real time.Drill down to see player session information. ||| Multi-region Fleets enable global deployments.|||
  Amazon GameLift Spot Fleets: Spot Fleets allow you to run game servers in Amazon GameLift at a lower hourly rate. Optimal for shorter game sessions. •Prices adjust based on demand for instance capacity.•Built-in algorithm estimates the viability to host game servers on Spot. •Linux and Windows instances available.• Instance interruptions can be managed via a notification mechanism (onProcessTerminate). •On-demand Fleets can be used as a failover when spot availability is low.|||
  Cost optimization with Fully-Managed GameLift: 1)Target-based scaling (Customer obsession!!!): Your fleet will always run exactly the amount of instances you need | Define a single configuration for available game sessions | Pay for only the resources you use. ||| 2)Fully-utilized servers: Run up to 50 game sessions on a single fleet instance | GameLift will automatically maximize the use of your instances. ||| 3)Spot Instances: Leverage unused EC2 capacity for a high discount on compute | Built-in algorithm checks Spot viability to minimize interruptions | Failover to on-demand fleets when spot availability is low | Optimal for shorter session length. |||
- Aliases: Aliases allow you to redirect game clients to a fleet that you specify or to notify the clients that a fleet is out of service. Two types of aliases: 1) Simple: A simple redirect points to an associated fleet, the fleet an Alias resolves to can be updated at any time. 2) Terminal: This does not resolve to a fleet, instead it passes back a specific message back to the client.
- Queues:Queues are important for earning money(this is from lecture notes). Queues automate the process of efficiently allocating new game sessions on any fleet in a group. Queues are configurable and allow you to determine several options that are used when Amazon GameLift places a session: Queue timeout, member fleets, fleet priority, latency policy.
  Queues-Member Fleets: Place Fleets in AWS Regions close to your players for the best experience. | You can use a single multi-region Fleet to cover all Regions. | 23 Regions supported by GameLift | Both Fleets and Aliases can be configured for use by a Queue | Queues can combine on-demand and spot instances from multiple regions.
  Queues- Fleet Priority: Prioritize your destinations (Fleets or Aliases) to indicate the preferred order. | Amazon GameLift uses the priority to locate the best available session, starting with the first priority and moving down. | When using Spot Instances set this Fleet to a higher priority, then set an On-Demand Fleet below for cases where Spot capacity is not available.
  Queues -Player Latency Policy: Set a maximum acceptable player latency for new game sessions| Specify the amount of time to enforce the policy | multiple policies can be specified | FlexMatch also does this but Queue latency configuration allows you to control overflow to other Regions in case of no available sessions.
  Queues -Best Practices : Use Multi-region Fleets registered to a Queue to deploy globally with a single build upload and Fleet configuration | Use a Spot Fleet on high priority, and prepare a failover on- demand Fleet | For disaster recovery, have a backup Queue in a separate Region and plan for failover | A queue cannot have fleets with different certificate configurations. All fleets in the queue must have TLS certificate generation either enabled or disabled.
- Clients: Amazon GameLift supports any game client or game service that is able to make use of one of the AWS supported SDKs, languages include:• C++ • C# • Go • Python • JavaScript/Node.js • Java… | Always build a backend service that calls GameLift APIs instead of the game client!
  Clients – Best Practices: Accept game client requests via a central game backend | Integrate with Amazon GameLift from the Game backend to help isolate clients from change | Authenticate game clients upon request to the Game Backend | Capture latency information from Clients by pinging AWS endpoints. This can be for example requests to DynamoDB endpoints available across the regions

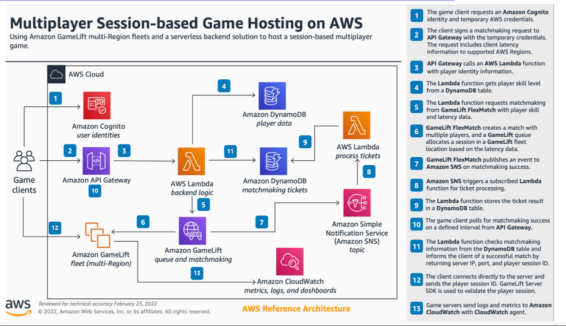
GameLift FleetIQ benefits: • More flexibility: Game Server Group hosted on EC2 on your AWS account • Simple API layer for game session management • Manages Fleet scaling for you • Use your existing tools and software to deploy and manage game server processes • Access GameLift Spot viability algorithm independent of other GameLift features • AWS SDK on the server side: use the programming language of your choice • It keeps load same for the all servers (this note from lecture) |||
How GameLift FleetIQ works: Use AWS SDK, CLI or CloudFormation to create a Game Server Group | An EC2 Auto Scaling Group is created with your configuration of instances. FleetIQ automatically scales this and manages spot and on-demand balance based on spot viability | You control how the game server processes are run on the instances | Game Servers report their state to FleetIQ. Your Backend calls ClaimGameServer() to request a free session|||
Matchmaking with Amazon GameLift FlexMatch: Fully managed customizable matchmaking | Works natively with GameLift Hosting, as well as standalone with any game server hosting solution | Player teams support | Latency-based matching | Rule relaxing | Match acceptance support | Best region placement | Player drop in/out support with FlexMatch backfill

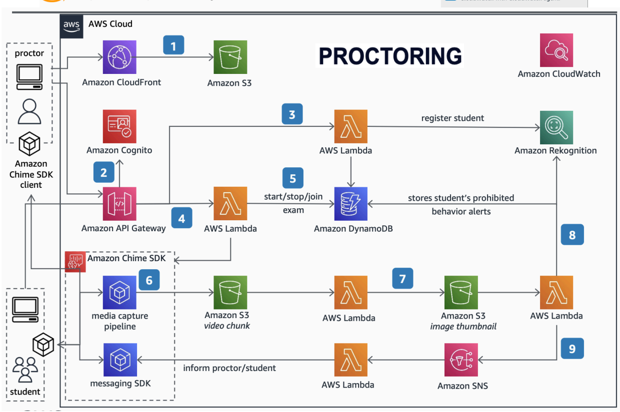
- GameLift Realtime: The Realtime servers have a managed server logic you can extend with scripts | Full network stack for game client/server interaction (encrypted TCP and UDP) | Integrated with the GameLift service | Live updates to Realtime configurations and server logic | Flexible control of hosting resources | Supports Unity/C# on the client side | Node.js scripts on the server side

Amazon GameLift Best Practices (Architecture): Don't call GameLift APIs from the client, use your backend instead and authenticate the players | Don't use any of the Describe or Search APIs of GameLift control plane in production -> they are not designed for production use. Instead, use Queue events and FlexMatch events | Use the Fleet IAM Role to access any AWS services (has to be assumed separately) | Review the checklists for different development phases | When using Spot fleets, always have a backup On-demand fleet as well behind the queue | Terminate game server processes after each game to reduce any memory leaks etc. Issues. Call ProcessEnding() before terminating to immediately get a replacement process | Only use Amazon Linux 2 OS for maximum performance and security |
Amazon GameLift Best Practices (Operations): Use CloudWatch Agent or 3rd party tools to collect process metrics, custom metrics and logs ⇒ Build relevant global dashboards for multi-region fleets | For game launch day, pre-warm the fleets and keep a high minimum, and switch to automatic scaling down after successful launch day | Make sure to request increases to all utilized GameLift APIs prior to launch | StartMatchmaking, StartGameSessionPlacement, CreateGameSession, CreatePlayerSession etc. | Make sure to request increases to all instance limits for all utilized instance types and regions | Monitor critical CloudWatch metrics | AvailableGameServers, TicketsFailed, AverageWaitTime, PlacementsFailed/TimedOut …



Src_2 (Slayttan)
Src_3
(gaming slide)





Integration