AWS Cloud Solution Architect Fieldbook

Simple explanations, enterprise patterns, and interview practice

AWS Cloud Solution Architect Fieldbook

> Independent learning material. AWS and related marks are trademarks of Amazon.com, Inc. or its affiliates.

00 · The Cloud Solution Architect role

A Cloud Solution Architect turns business goals into a safe, useful cloud design.

Explain it like I am five

Imagine a company is a busy city. SAP is the warehouse, Salesforce is the sales office, Workday is the people office, and the customer website is the shop. The architect does not personally build every room. The architect decides where the roads go, who gets keys, what happens when a road closes, and how much the city can afford.

The role in one loop

1. **Discover** the business outcome, users, data, rules, scale, and deadlines.
2. **Design** a few viable options, including failure and security behavior.
3. **Decide** with stakeholders by making trade-offs visible.
4. **Enable** delivery with diagrams, guardrails, examples, and a roadmap.
5. **Verify** the result with tests, telemetry, cost data, and reviews.
6. **Improve** after learning from production.

What this role owns

Responsibility	The practical question
Architecture	What parts exist, and how do they communicate?
Integration	How do SAP, Salesforce, Workday, data, and channels exchange truth?
Security	Who may do what, and how will we know?
Reliability	What fails, how far does failure spread, and how do we recover?
Scalability	What happens when demand becomes ten times larger?
Cost	What business value do we buy with each major cost?
Sustainability	Can we do the same work with fewer resources?
Communication	Can engineers and executives understand the decision?

What the role does not mean

It is not memorizing hundreds of service names. It is not drawing a diagram and leaving. It is not choosing a fashionable technology before discovering the problem. A strong architect can say: “Here are the constraints, here are three options, here is the decision, and here is how we will prove it works.”

Role evidence to build

- A one-page context diagram and a deployment diagram.
- An architecture decision record with alternatives and consequences.
- A threat model and a least-privilege access model.
- A failure-mode table with RTO and RPO.
- A monthly cost estimate with the largest cost drivers.
- An observability plan with service-level objectives.
- A migration roadmap with reversible steps.

Your learning target

By the end of this fieldbook you should be able to take a scenario, ask the missing questions, select a small set of AWS services, explain the trade-offs, and defend the design against the six AWS Well-Architected pillars.

01 · Cloud foundations and global infrastructure

AWS rents computing building blocks over a network. You combine those blocks into a workload and pay for what you use.

The map

- A **Region** is a separate geographic area.
- An **Availability Zone (AZ)** is an isolated location inside a Region, made from one or more data centers with separate power and networking.
- An **edge location** puts content and network services close to users.
- An **AWS account** is a strong boundary for billing, permissions, quotas, and blast radius.
- A **resource** is something you create: a bucket, instance, database, queue, function, key, or network.

The simple design rule

Use multiple AZs for high availability inside one Region. Use multiple Regions only when business needs justify the extra complexity: strict disaster recovery, data residency, or globally low latency.

Shared responsibility

AWS secures **the cloud**: physical facilities, hardware, and the managed service platform. The customer secures what is placed **in the cloud**: identities, data, configuration, application code, and many operating-system duties. The exact line changes by service. With EC2 you manage more. With Lambda or DynamoDB AWS manages more of the underlying stack.

Managed services first

Choose the highest-level managed service that meets the requirement. It usually removes undifferentiated work such as patching, replication, and replacement. Do not choose it blindly: check service limits, portability, skills, latency, features, and cost at the expected scale.

Architect questions

1. Where are users and regulated data?
2. Which failures must the design survive?
3. What is the tolerated latency?
4. What must remain on premises?
5. Which teams operate the solution?
6. How will accounts and environments be separated?

Common trap

“The cloud is automatically highly available” is false. AWS supplies resilient building blocks. You must place and configure them so the workload survives the failures that matter.

02 · The six Well-Architected pillars

The AWS Well-Architected Framework is the architect's repeatable checklist. It helps expose risk; it is a conversation, not a punishment.

Six lenses for every decision

Pillar	Child-simple question	Typical evidence
Operational excellence	Can we run and improve it?	automation, runbooks, telemetry
Security	Is it protected?	identities, encryption, detection
Reliability	Does it recover?	redundancy, tested recovery
Performance efficiency	Is the right tool doing the job?	load tests, right sizing
Cost optimization	Are we wasting money?	allocation, budgets, unit cost
Sustainability	Are we doing less work for the same result?	efficient code and resources

A useful review rhythm

At discovery, record assumptions and high-risk choices. Before production, review all six pillars and assign owners and dates to serious risks. In production, use real telemetry, incidents, and spend to update the review.

Trade-offs are normal

More redundancy improves reliability but usually raises cost. Encryption and inspection add small performance and operational costs but reduce security risk. A serverless design reduces infrastructure operations but can introduce quotas, startup latency, and tighter service coupling. The architect makes the trade-off explicit and links it to the business goal.

The answer pattern

For any architecture question:

1. State the requirement and constraint.
2. Name the failure or threat.
3. Propose the design.
4. Explain why it meets the requirement.
5. State the cost or limitation.
6. Explain how it will be tested and observed.

Review prompt

Do not ask only “Is it available?” Ask: “Which component can fail, what will the user see, how quickly will detection happen, who responds, and what proof shows recovery works?”

Official source: AWS Well-Architected Framework.

03 · Identity, accounts, and governance

Good enterprise architecture starts with boundaries, not servers.

Identity in simple terms

An identity says **who** is asking. A policy says **what** they may do. A role is a temporary set of permissions. Prefer temporary credentials and federation through IAM Identity Center. Avoid long-lived access keys.

The permission decision

AWS starts with implicit deny. An applicable allow is required. An explicit deny wins. Identity policies, resource policies, permission boundaries, session policies, and Organizations controls can all limit the final permission.

Multi-account landing zone

Use AWS Organizations to group accounts into organizational units (OUs). Use AWS Control Tower to establish and govern a landing zone. A practical layout:

- management account: organization administration only;
- security OU: audit and centralized log archive accounts;
- infrastructure OU: network and shared services;
- workloads OU: separate production and non-production accounts;
- sandbox OU: experimentation with spending and access limits.

Service control policies (SCPs) set the maximum available permissions. They do not grant permission. Controls can prevent, detect, or proactively reject unwanted configurations.

Security foundations

- Require MFA for people and protect the root user.
- Centralize workforce access with short-lived roles.
- Separate workloads and environments by account.
- Log organization activity centrally.
- Encrypt with KMS keys and control who can use each key.
- Store secrets in Secrets Manager or Parameter Store, never source code.
- Automate accounts and guardrails; do not handcraft each environment.

Architect trade-off

Too few accounts create a large blast radius and tangled billing. Too many accounts without automation create operational friction. Choose boundaries around ownership, environment, sensitivity, and failure containment.

Official source: AWS Control Tower.

04 · Networking and hybrid connectivity

A VPC is your private network boundary in one AWS Region.

The building blocks

- **Subnet:** an IP range in one AZ. A public subnet has a route to an internet gateway; a private subnet does not expose workloads directly.
- **Route table:** where network traffic should go.
- **Security group:** stateful firewall attached to a resource.
- **Network ACL:** stateless subnet-level rules; use as coarse guardrails.
- **NAT Gateway:** lets private IPv4 resources initiate internet connections.
- **VPC endpoint / PrivateLink:** private access to supported services without traversing the public internet.
- **Transit Gateway:** hub that connects many VPCs and on-premises networks.
- **Route 53:** DNS and health-aware routing.
- **CloudFront:** global content delivery and edge protection.

Connecting the enterprise

Use Site-to-Site VPN for encrypted connectivity over the internet and as a fast starting point or backup. Use Direct Connect for private, more predictable network connectivity. Direct Connect is not encryption by itself; add the appropriate encryption where required.

For many VPCs, use a hub-and-spoke design with Transit Gateway or Cloud WAN. Keep routing domains and responsibilities clear. Inspect traffic centrally only when the compliance value justifies the cost and dependency.

Design questions

- Can CIDR ranges overlap with acquisitions or partner networks?
- Which flows are actually required?
- Where does DNS resolve for hybrid systems?
- What happens if one link or AZ fails?
- Which egress paths are allowed and logged?
- Is latency suitable for chatty SAP or database protocols?

Common trap

A private subnet is not automatically secure. It still needs least-privilege security groups, controlled egress, patching, logging, and protected identities.

05 · Compute, serverless, and containers

Choose compute by operational responsibility and workload behavior.

Decision table

Need	Good starting point	Why
Full OS control or legacy software	EC2	maximum control
Container orchestration with AWS simplicity	ECS	managed AWS-native scheduler
Kubernetes ecosystem or portability need	EKS	managed Kubernetes control plane
Containers without managing hosts	Fargate	serverless container capacity
Short event-driven code	Lambda	pay per invocation, automatic scaling
Simple managed web deployment	App Runner or Elastic Beanstalk	less platform assembly
Batch jobs	AWS Batch	scheduling and compute orchestration

EC2 architecture

Put replaceable instances in an Auto Scaling group across multiple AZs, behind an Application or Network Load Balancer. Keep session state outside instances. Use launch templates, Systems Manager, and

immutable images or automated bootstrap. Choose Savings Plans or Reserved Instances only for predictable baseline use; let burst capacity remain flexible.

Lambda architecture

Functions should be small, idempotent where possible, and safe to retry. Control concurrency, set timeouts, send failed asynchronous events to a destination or dead-letter queue, and observe duration, errors, throttles, and downstream pressure.

Containers

Store images in ECR and scan them. Use ECS when the team wants containers without Kubernetes complexity. Use EKS when Kubernetes capabilities, ecosystem, or organizational standards provide real value. Fargate removes worker-node management but may cost more at steady high utilization.

Architect rule

Do not select the tool from fashion. Select it from runtime shape, control, team skill, scaling, compliance, portability, and total operating cost.

06 · Storage, databases, analytics, and data

The first data question is not “Which database?” It is “How will the data be used, protected, changed, and recovered?”

Storage

- **S3:** durable object storage for files, logs, backups, and data lakes.
- **EBS:** block volumes for EC2, tied to one AZ.
- **EFS:** shared regional Linux file system.
- **FSx:** managed file systems for Windows, Lustre, NetApp ONTAP, or OpenZFS.
- **AWS Backup:** policy-based backup across supported services.

Use S3 versioning, lifecycle rules, encryption, blocked public access, and replication only when the recovery or residency requirement needs it.

Databases

- **RDS/Aurora:** relational transactions and SQL.
- **DynamoDB:** serverless key-value/document access at predictable low latency.
- **ElastiCache:** in-memory cache; do not treat a cache as the only truth.

- **Redshift:** analytical data warehouse.
- **OpenSearch Service:** search, logs, and analytical search.
- **Neptune:** graph relationships.
- **DocumentDB:** document workloads needing MongoDB compatibility.

Decision questions

What are the access patterns? Do transactions span records? How much data and traffic? Which consistency is required? What is the RPO/RTO? Can the application handle retries? Is cross-Region replication necessary? How will schema changes be deployed?

Data platform

Land raw data in S3, catalog it with Glue, control access with Lake Formation, query with Athena or process with EMR/Glue, and warehouse curated analytics in Redshift when appropriate. Separate operational transactions from analytics so reports do not hurt customer traffic.

Common trap

“NoSQL scales better” is not a complete decision. DynamoDB is excellent when access patterns are known. Relational databases are excellent when relationships, transactions, and flexible queries matter. Use requirements, not slogans.

07 · Application and enterprise integration

Integration makes independent systems cooperate without turning them into one fragile machine.

Four patterns

Pattern	Use it when	AWS starting points
Request/response	caller needs an immediate answer	API Gateway, ALB, AppSync
Queue	work can wait and needs buffering	SQS, Amazon MQ
Publish/subscribe	many consumers need the same event	SNS, EventBridge
Stream	ordered continuous records must be processed	Kinesis, MSK

Step Functions coordinates multi-step workflows. EventBridge routes events by content. SQS absorbs bursts and isolates failures. SNS fans messages out. Amazon MQ supports familiar broker protocols. MSK provides managed Apache Kafka.

Integration rules

- Prefer contracts and business events over shared databases.
- Make consumers idempotent because delivery can repeat.
- Set timeouts, retries with backoff and jitter, and dead-letter handling.
- Add correlation IDs, schema versions, ownership, and observability.
- Do not retry permanent validation errors forever.
- Protect synchronous chains from cascading failure.

SAP, Salesforce, and Workday

Use AppFlow for supported managed data transfers, including Salesforce and SAP OData. Use APIs and events for operational integration. Use DMS for supported database migration/change data capture, Transfer Family or DataSync for managed file patterns, and an integration platform when enterprise transformation and governance needs justify it.

Workday integration commonly uses Workday-supported APIs, reports, or files through a controlled integration layer. Confirm the tenant's supported interfaces; do not invent a native connector that does not exist.

Architect decision

Separate systems of record from copies. Define which system owns customer, product, employee, price, and order facts. Every replicated field needs an owner, freshness target, failure path, and reconciliation process.

Official sources: Amazon AppFlow, Salesforce connector, and SAP OData connector.

08 · Security architecture

Security is a system of prevention, detection, response, and learning.

Protect identities

Federate people, use roles for workloads, grant least privilege, require MFA, rotate or remove long-lived credentials, and use Access Analyzer to find unintended access.

Protect data

Classify data. Encrypt in transit with TLS and at rest with service encryption and KMS. Put secrets in Secrets Manager. Restrict S3 public access. Use Macie to help discover sensitive data in S3 where useful. Define retention and deletion, not only encryption.

Protect workloads

Use security groups, private placement, Systems Manager instead of open administration ports, WAF for web exploits, Shield for DDoS protection, Inspector for vulnerability findings, and hardened deployment pipelines.

Detect

- CloudTrail records API activity.
- Config records resource configuration and evaluates rules.
- GuardDuty analyzes signals for threats.
- Security Hub aggregates and prioritizes security findings.
- CloudWatch handles metrics, logs, alarms, and operational events.

Centralize immutable logs in a dedicated account and protect them from workload administrators.

Respond

Predefine owners, severity, containment actions, evidence handling, and communication. Automate safe responses such as quarantining a resource or disabling a leaked key, but include rollback and human escalation.

Threat-model prompt

For each data flow ask: who can call it, how they authenticate, what data moves, where it is decrypted, what happens if a credential is stolen, and what signal would reveal misuse.

09 · Reliability, resilience, and disaster recovery

Availability keeps a service working. Disaster recovery restores it after a serious event. Backup is only one recovery ingredient.

Start with business numbers

- **RTO:** maximum acceptable time to restore service.
- **RPO:** maximum acceptable amount of lost data measured in time.
- **SLO:** measurable reliability target, such as successful requests.

Do not buy a multi-Region design before the business agrees it needs the complexity and cost.

Reliability patterns

- Distribute across AZs and remove single points of failure.
- Scale horizontally and keep compute replaceable.
- Buffer bursty work with queues.
- Use timeouts, bounded retries, exponential backoff, and jitter.
- Make operations idempotent.
- Use health checks, graceful degradation, and load shedding.
- Test backups and recovery, not only backup creation.
- Practice failure with game days.

DR strategies

Strategy	Relative cost	Recovery
Backup and restore	low	slowest
Pilot light	low-medium	core services always ready
Warm standby	medium-high	reduced-size environment running
Multi-site active/active	highest	fastest, most complex

Multi-AZ versus Multi-Region

Multi-AZ handles data-center/AZ failure and should be the normal availability baseline for important regional workloads. Multi-Region addresses larger regional or sovereignty needs but adds data replication, consistency, routing, deployment, testing, and operational complexity.

Common trap

Retries can cause an outage if every caller retries together. Bound retries, back off with jitter, make requests safe to repeat, and stop when the remaining time cannot support another attempt.

10 · Operations, cost, performance, and sustainability

A design is not finished until teams can see, run, and afford it.

Observability

Collect metrics, logs, traces, and business outcomes. CloudWatch provides metrics, logs, alarms, dashboards, and synthetics. X-Ray or OpenTelemetry traces requests across services. CloudTrail answers who changed AWS resources.

Alarm on symptoms users feel and on causes operators can act on. Every critical alarm needs an owner and a runbook.

Infrastructure and change

Use CloudFormation, CDK, Terraform, or another governed infrastructure-as-code tool. Review changes, scan them, deploy through environments, and detect drift. Use Systems Manager for fleet operations and Parameter Store where appropriate.

Cost

Tag and account-separate ownership. Use Cost Explorer, Budgets, Cost and Usage Reports, Savings Plans, Spot capacity, and Compute Optimizer. Track unit cost, such as cost per order, not only the monthly total.

The biggest wins usually come from architecture: deleting idle resources, right-sizing, using storage lifecycle policies, scheduling non-production, choosing efficient data transfer paths, and matching purchase models to demand.

Performance

Test with realistic load. Select the right compute, database, storage, and networking. Cache only with a clear freshness and invalidation policy. Use CloudFront close to global users and asynchronous work when a response need not wait.

Sustainability

Use fewer resources for the same outcome: scale with demand, eliminate idle capacity, use managed services, efficient processors such as Graviton where compatible, efficient code, sensible data retention, and Regions that meet both business and sustainability requirements.

11 · Migration and modernization

Migration is a business change carried by technology.

Discover first

Inventory applications, owners, dependencies, data, contracts, usage, costs, and risk. Use Migration Hub and discovery tooling where appropriate. Group systems into waves that produce value and keep dependencies manageable.

The migration choices

- **Retire:** remove what no longer provides value.
- **Retain:** keep it for now.
- **Rehost:** move with minimal change.
- **Relocate:** move a platform with little application change.
- **Replatform:** use some managed capabilities without rewriting everything.
- **Repurchase:** replace with a product or SaaS.
- **Refactor:** redesign for cloud-native capabilities.

Choose per workload. A company-wide “refactor everything” program usually creates too much simultaneous risk.

Useful services

Application Migration Service (MGN) supports rehosting servers. Database Migration Service (DMS) moves supported databases and can replicate changes. DataSync moves file/object data. Storage Gateway connects on-premises workflows to cloud storage. Snow Family supports some offline/edge transfer cases.

Safe wave pattern

1. Build the landing zone and connectivity.
2. Pilot a representative low-risk workload.
3. Prove security, operations, backup, and rollback.
4. Migrate bounded waves.
5. Validate business data and performance.
6. Cut over with explicit go/no-go criteria.
7. Decommission only after reconciliation and retention obligations.

Modernization rule

Modernize where it changes an important outcome: faster releases, lower operational burden, better scale, or lower risk. Do not rewrite a stable system only to make the diagram look newer.

12 · Real-life scenario: Global Sales Connect

This fictional scenario mirrors the responsibilities in the role description without claiming to describe Hilti's internal architecture.

The situation

BuildCo sells equipment in 60 countries. Customers use web and mobile channels. Sales teams use Salesforce, orders and product prices live in SAP, employees and territories come from Workday, and marketing needs trusted customer events. Peak campaigns create ten times normal traffic.

The goals are faster customer experiences, no duplicate orders, regional resilience, controlled personal data, and a platform that teams can extend.

Questions before services

- Which system owns customer, product, price, employee, consent, and order data?
- Which countries require data to stay in a particular location?
- What are the checkout RTO, RPO, and latency targets?
- Can SAP accept bursts, or must requests be buffered?
- Which integrations need immediate answers and which can be eventual?
- How will duplicate and out-of-order events be handled?

Proposed architecture

CloudFront and WAF protect and accelerate the customer channel. Route 53 directs traffic. The regional application runs stateless services on ECS Fargate across three AZs behind an Application Load Balancer. API Gateway publishes partner and channel APIs.

Aurora stores channel transactions that require relational consistency. DynamoDB stores high-volume idempotency keys and session-like state. S3 stores documents, events, and the analytics lake.

The application emits business events to EventBridge. SQS queues isolate SAP order processing and absorb campaigns. Step Functions coordinates long-running order steps. Failed messages go to a dead-letter queue with alarms and a replay runbook. Every command carries a correlation and idempotency key.

Salesforce customer changes move through AppFlow for managed bulk/scheduled data movement and EventBridge/API patterns where timely operational events are needed. SAP exposes governed APIs or OData; AppFlow can serve supported OData flows, while orders use a controlled integration service and queues. Workday employee and territory changes enter through approved Workday APIs or encrypted files, then become normalized business events. No system reads another system's database directly.

Security and governance

Production, non-production, network, security, and log archive live in separate accounts governed by Control Tower. IAM Identity Center federates people. Workloads use roles. KMS encrypts data, Secrets Manager stores credentials, CloudTrail and Config feed the security account, GuardDuty and Security Hub raise findings, and Macie helps inspect sensitive S3 data.

Reliability and operations

Multi-AZ is the normal operating design. The business chooses warm standby in a second Region for checkout after comparing cost with the RTO/RPO. Aurora and S3 data follow tested replication/backup plans; Route 53 failover is exercised in game days. CloudWatch dashboards show checkout success, queue age, SAP failure rate, latency, error budget, and cost per order.

Trade-offs

- Asynchronous SAP writes protect SAP and improve scale, but order confirmation becomes “accepted” before final completion.
- Multi-Region improves recovery but doubles many operational paths.
- AppFlow reduces integration code for supported flows, but custom rules still belong in owned services.
- Fargate reduces host operations but should be compared with EC2 at steady, large scale.

Delivery roadmap

1. Establish landing zone, identity, logging, network, and cost allocation.
2. Deliver customer read journeys with cached product data.
3. Add the queued order path with reconciliation and failure replay.
4. Integrate Salesforce and Workday through owned contracts.
5. Add analytics, DR rehearsal, optimization, and regional expansion.

Evidence of success

Measure checkout availability, p95 latency, duplicate orders, reconciliation exceptions, mean recovery time, security findings age, deployment frequency, and cost per successful order.

13 · Architecture and interview playbook

Interviewers want to see structured judgment, not a service-name race.

Use REQUIRE

- **R — Requirements:** users, business outcome, scale, latency, RTO/RPO.
- **E — Environment:** current systems, skills, Regions, compliance.
- **Q — Qualities:** security, reliability, cost, operability, sustainability.
- **U — Unknowns:** state assumptions and ask clarifying questions.
- **I — Interfaces:** APIs, events, data ownership, failure behavior.
- **R — Recommendation:** show a simple end-to-end design.
- **E — Evidence:** testing, observability, cost, rollout, and alternatives.

A strong answer sounds like this

“I would first confirm whether an immediate SAP response is required. If it is not, I would accept the order through a regional API, commit a durable order record, and queue SAP work in SQS. This isolates campaign spikes and permits controlled retries. The trade-off is eventual confirmation, so the customer experience and reconciliation process must show pending and failed states. I would prove it with load tests, queue-age alarms, idempotency tests, and a game day where SAP is unavailable.”

Questions you should be ready for

1. Design a global customer-facing platform.
2. Connect AWS with SAP, Salesforce, and Workday.
3. Design a secure multi-account landing zone.
4. Explain Multi-AZ and Multi-Region.
5. Migrate a legacy application with little downtime.
6. Reduce cloud cost without weakening reliability.
7. Respond to a leaked credential or public bucket.
8. Choose Lambda, ECS, EKS, or EC2.
9. Choose RDS/Aurora, DynamoDB, and a cache.
10. Handle duplicated events and downstream outages.

Behavioral stories

Prepare concise STAR stories about: influencing without authority, reversing a decision after new evidence, handling an incident, explaining a trade-off to a non-technical stakeholder, reducing cost, and raising a security concern.

Whiteboard discipline

Draw users and external systems first, then boundaries, the normal request path, data stores, asynchronous paths, observability, and failure behavior. Add service names only after the responsibilities are clear.

14 · SAA-C03 certification checkpoint

The AWS Certified Solutions Architect – Associate exam is a useful checkpoint for service selection and architecture fundamentals. It does not by itself prove enterprise integration or stakeholder leadership.

Current official shape

- 65 questions total: 50 scored and 15 unscored.
- Multiple-choice and multiple-response questions.
- Scaled score from 100 to 1,000; minimum passing score 720.
- Domain 1, secure architectures: 30%.
- Domain 2, resilient architectures: 26%.
- Domain 3, high-performing architectures: 24%.
- Domain 4, cost-optimized architectures: 20%.

Unscored questions are not identified, so treat every question seriously. There is no penalty for guessing; unanswered questions are incorrect.

How to solve scenario questions

1. Underline the exact requirement: **most cost-effective**, **least operational effort**, **highest availability**, **near-real-time**, or **private**.
2. Remove answers that violate a hard constraint.
3. Prefer managed services when they meet the requirement.
4. Check scope: AZ, Region, global, account, or organization.
5. Check data and failure behavior.
6. Choose all required responses in multiple-response questions.

Study loop

Learn one concept, explain it without notes, compare close alternatives, answer scenario questions, and write why every distractor is wrong. Revisit missed ideas after one day, three days, one week, and two weeks.

Use the 65-question mock in this site under timed conditions. The questions are original practice items aligned to the official domains; they are not copied AWS exam questions.

Official source: AWS SAA-C03 exam guide.

15 · Glossary and official references

Use this compact glossary when a conversation becomes acronym-heavy.

A–Z glossary

Term	Plain meaning
ALB	Application Load Balancer for HTTP/HTTPS traffic
ARN	Unique AWS resource name
Aurora	AWS relational database compatible with MySQL/PostgreSQL
AZ	Isolated location inside an AWS Region
CAF	Cloud Adoption Framework for organizational cloud change
CDK	Define cloud infrastructure using programming languages
CIDR	Notation for an IP address range
CloudFront	Global content delivery network
CloudTrail	Record of AWS API activity
CloudWatch	Metrics, logs, alarms, dashboards, and operations
DMS	Database Migration Service
DR	Disaster recovery
EBS	Block storage for EC2
EC2	Resizable virtual machines
ECR	Container image registry
ECS	AWS container orchestration
EFS	Shared managed Linux file system
EKS	Managed Kubernetes
EventBridge	Event bus and event routing
Fargate	Serverless compute for containers
IAM	Identity and Access Management
IaC	Infrastructure as code
IGW	Internet gateway
KMS	Managed encryption keys
Lambda	Event-driven serverless functions
NACL	Stateless subnet network rules

Term	Plain meaning
NAT	Outbound address translation for private resources
NLB	Network Load Balancer for high-performance L4 traffic
OU	Organizational unit containing AWS accounts
PrivateLink	Private service connectivity through VPC endpoints
RDS	Managed relational databases
Region	Separate AWS geographic area
RPO	Maximum acceptable data loss measured in time
RTO	Maximum acceptable recovery time
S3	Durable object storage
SCP	Organization policy that limits maximum permissions
Security group	Stateful resource firewall
SLO	Measurable service reliability target
SNS	Publish/subscribe notifications
SQS	Durable managed message queue
VPC	Private regional network
WAF	Web application firewall

Official starting points

- AWS Well-Architected Framework
- AWS Architecture Center
- AWS Prescriptive Guidance
- AWS Control Tower
- AWS Security Documentation
- AWS Disaster Recovery guidance
- AWS SAA-C03 Exam Guide

Accuracy rule

AWS services, limits, features, and exam guides change. Use this fieldbook to understand decisions, then verify implementation details in current official documentation before production work.