MODULE 9 · STUDY NOTES

Cloud Architecture

The Well-Architected Framework and its six pillars, designing for reliability and high availability, and AWS Trusted Advisor

The first eight modules toured AWS service by service; this one asks how to put the pieces together well. Architecture is the art and science of designing and building large systems, and AWS distilled the lessons of thousands of customer reviews into the Well-Architected Framework — six pillars of design principles and best-practice questions that let you judge any architecture consistently. Because "everything fails, all the time" (Werner Vogels), the module then puts numbers on failure: reliability measured by MTBF, availability measured in 9s, and the three factors — fault tolerance, scalability, recoverability — that decide how much downtime your users see. It closes with AWS Trusted Advisor, the online tool that checks your live environment against best practices in real time.

LEARNING OBJECTIVES

- Describe the AWS Well-Architected Framework, including the six pillars
- Identify the design principles of the AWS Well-Architected Framework
- Explain the importance of reliability and high availability
- Identify how AWS Trusted Advisor helps customers
- Interpret AWS Trusted Advisor recommendations

9.1 The AWS Well-Architected Framework

Architecture is the art and science of designing and building large structures, and large systems need architects to manage their size and complexity. **Cloud architects** engage with decision makers to identify the business goal and the capabilities that need improvement, ensure alignment between a solution's technology deliverables and the business goals, and work with delivery teams so the technology features stay appropriate. Well-architected systems greatly increase the likelihood of business success.

The **AWS Well-Architected Framework** is a guide for building the most **secure, high-performing, resilient, and efficient** infrastructure possible for cloud applications and workloads. It provides a set of **foundational questions and best practices** for evaluating and implementing cloud architectures — a *consistent approach*, developed after AWS reviewed **thousands of customer architectures**. The framework is organized into six pillars; the first five have been there since its 2015 introduction, and **sustainability** was added as the sixth in **2021** to help organizations minimize the environmental impacts of running cloud workloads. This module works through the first five in depth.

TABLE 9.1 The six pillars of the Well-Architected Framework

PILLAR	FOCUS	KEY TOPICS
Operational excellence	Run and monitor systems to deliver business value, and continually improve supporting processes and procedures	Automating changes · responding to events · defining standards to manage daily operations
Security	Protect information, systems, and assets while delivering business value through risk assessments and mitigation strategies	Confidentiality and integrity of data · managing who can do what · protecting systems · detecting security events
Reliability	Ensure a workload performs its intended function correctly and consistently when it's expected to	Designing distributed systems · recovery planning · handling change
Performance efficiency	Use IT and computing resources efficiently to meet requirements, and keep that efficiency as demand changes and technologies evolve	Selecting right resource types and sizes · monitoring performance · informed decisions as business needs evolve
Cost optimization	Avoid unnecessary costs	Understanding and controlling spend · right type and number of resources · analyzing spend over time · scaling without overspending
Sustainability	Minimize the environmental impacts of running cloud workloads	Added 2021; covered in its own section of the framework documentation

How a pillar is organized. Each pillar contains a set of *design principles* and a set of *best practice areas*. Each best practice area aligns to **questions a reviewer should ask** when designing an architecture, and the full question set lives in the framework's Appendix. The anatomy, using the security pillar as the example: best practice area (*Identity and Access Management*) → question (**SEC 1:** *How do you manage credentials and authentication?*) → question context → best practices (define IAM requirements, secure the account root user, enforce MFA, automate enforcement of access controls, integrate a centralized federation provider, enforce password requirements, rotate and audit credentials).

THE ANYCOMPANY REVIEW ACTIVITY

The module's running activity is an architecture review of **AnyCompany Corporation** — a cloud-native seller of 3D-printed cityscapes, founded 2008, now preparing for investment and asking you to review its platform for due diligence. Its three departments mirror its architecture: **Fly and Snap** (aircraft-mounted cameras → detachable storage array → FTP upload to an EC2 *Preprocessor* → imagery in S3, flight metadata in a self-managed RDBMS on EC2, tape backups held offsite), **Show and Sell** (website on Elastic Load Balancing with HTTPS + an Auto Scaling group running a CMS, static assets in S3, payments via a PCI-compliant third party, orders recorded in another RDBMS on EC2), and **Make and Ship** (a *Render* fleet of `g2.2xlarge` instances consuming a Production queue, 3D models and preview videos in S3, an on-premises *Print conductor* feeding four 3D printers, status flowing back through an Order-status queue). Details are planted for you to find — for example, the founder gave the **AWS account root user credentials to his whole team**. For each pillar you answer: what is the **current state**, what should the **future state** be, and what single top improvement would you make? There are no right or wrong answers — the framework questions exist to prompt discussion.

9.2 Operational Excellence pillar

Operational excellence is the ability to **run and monitor systems to deliver business value** and to **continually improve** supporting processes and procedures. Key topics: automating changes, responding to events, and defining standards to manage daily operations.

TABLE 9.2 Operational excellence – five design principles

PRINCIPLE	WHAT IT MEANS
Perform operations as code	Define the entire workload — applications *and* infrastructure — as code and update it with code; implement operations procedures as code that trigger automatically on events. Limits human error, gives consistent responses
Make frequent, small, reversible changes	Design workloads so components update regularly, in small increments that can be reversed if they fail — without affecting customers when possible
Refine operations procedures frequently	Evolve procedures as workloads evolve; run regular game days to review procedures, validate their effectiveness, and keep teams familiar with them
Anticipate failure	Identify potential failure sources so they can be removed or mitigated; test failure scenarios, validate their impact, and test response procedures
Learn from all operational failures	Drive improvement through lessons learned from all operational events and failures; share what's learned across teams and the whole organization

The pillar's foundational questions fall under four best practice areas: **organization, prepare, operate, and evolve**. Operations teams must understand business and customer needs to support business outcomes; they create and validate procedures for responding to operational events, collect metrics that measure achievement of desired outcomes, and — as priorities and customer needs change — design operations that evolve and incorporate lessons learned. The AnyCompany breakout uses OPS 4 (*how do you design your workload so that you can understand its state?*), OPS 6 (*how do you mitigate deployment risk?*), and OPS 7 (*how do you know you're ready to support a workload?*).

9.3 Security pillar

The security pillar is the ability to **protect information, systems, and assets** while delivering business value through **risk assessments and mitigation strategies**. Key topics: protecting the confidentiality and integrity of data, identifying and managing who can do what (privilege management), protecting systems, and establishing controls to detect security events.

TABLE 9.3 Security – seven design principles

PRINCIPLE	WHAT IT MEANS
Implement a strong identity foundation	Least privilege and separation of duties for every interaction with AWS resources; centralize privilege management; reduce or eliminate long-term credentials
Enable traceability	Monitor, alert, and audit actions and changes in real time; integrate logs and metrics with systems that automatically respond
Apply security at all layers	Defense in depth: controls at edge network, VPC, subnet, load balancer, and every instance, operating system, and application
Automate security best practices	Secure architectures with controls defined and managed **as code** in version-controlled templates — scale securely, rapidly, cost-effectively
Protect data in transit and at rest	Classify data into sensitivity levels; apply encryption, tokenization, and access control where appropriate
Keep people away from data	Reduce or eliminate direct access and manual processing of data to cut the risk of loss or modification from human error
Prepare for security events	Have an incident management process aligned to organizational requirements; run incident response simulations; use automated tooling to speed detection, investigation, and recovery

The foundational questions fall under six best practice areas: **security, identity and access management, detection, infrastructure protection, data protection, and incident response**. Security practices belong in place *before* you architect any system: control who can do what, identify incidents, protect systems and services, maintain data confidentiality and integrity, and keep a well-practiced incident response process — objectives like preventing financial loss and meeting regulatory obligations depend on them. The AnyCompany breakout uses SEC 1, SEC 4, and SEC 6.

9.4 Reliability pillar

The reliability pillar ensures a workload **performs its intended function correctly and consistently when it's expected to**. A resilient workload quickly recovers from failures to meet business and customer demand. Key topics: designing distributed systems, recovery planning, and handling change.

TABLE 9.4 Reliability – five design principles

PRINCIPLE	WHAT IT MEANS
Automatically recover from failure	Monitor KPIs and trigger automated recovery when a threshold is breached — automatic notification, failure tracking, and recovery that works around or repairs the failure
Test recovery procedures	Test how systems fail and validate recovery; use automation to simulate failures and recreate past failure scenarios — expose failure pathways before a real event
Scale horizontally	Replace one large resource with multiple smaller ones and distribute requests across them, reducing the impact of any single point of failure
Stop guessing capacity	Monitor demand and usage; automate adding and removing resources to stay at the optimal level
Manage change in automation	Make infrastructure changes through automation — and manage the automation's own changes the same way

The foundational questions fall under four best practice areas: **foundations, workload architecture, change management, and failure management**. A reliable system has a well-planned foundation with monitoring in place, mechanisms for handling changes in demand or requirements, and a design that detects failure and **automatically heals itself**. The AnyCompany breakout uses REL 2 (*network topology*), REL 7 (*adapting to changes in demand*), and REL 9 (*backing up data*).

9.5 Performance Efficiency pillar

Performance efficiency is the ability to **use IT and computing resources efficiently** to meet system requirements — and to *maintain* that efficiency as demand changes and technologies evolve. Key topics: selecting the right resource types and sizes based on workload requirements, monitoring performance, and making informed decisions as business needs evolve.

TABLE 9.5 Performance efficiency – five design principles

PRINCIPLE	WHAT IT MEANS
Democratize advanced technologies	Consume hard technologies (NoSQL databases, media transcoding, machine learning) *as services*, so teams focus on product development instead of provisioning and management
Go global in minutes	Deploy systems in multiple AWS Regions for lower latency and a better customer experience at minimal cost
Use serverless architectures	Remove the operational burden of running and maintaining servers; managed services at cloud scale can also lower transactional costs
Experiment more often	Perform comparative testing of different instance types, storage, and configurations
Consider mechanical sympathy	Use the technology approach that aligns best with the goal — e.g. match database and storage choices to your data access patterns

The foundational questions fall under four best practice areas: **selection, review, monitoring, and tradeoffs**. Use *data* to design and build high-performance architectures — from high-level design down to resource types and configuration. Review choices periodically to exploit new AWS services, monitor so you catch deviations from expected performance,

and use tradeoffs to improve performance — such as **compression, caching, or relaxing consistency requirements**. The AnyCompany breakout uses PERF 1, PERF 2, and PERF 4.

9.6 Cost Optimization pillar

The cost optimization pillar is the ability to **avoid unnecessary costs**. Key topics: understanding and controlling where money is being spent, selecting the most appropriate and right number of resource types, analyzing spend over time, and scaling to meet business needs without overspending.

TABLE 9.6 Cost optimization – five design principles

PRINCIPLE	WHAT IT MEANS
Implement Cloud Financial Management	Invest in the capability — knowledge building, programs, resources, and processes — to become a cost-efficient organization
Adopt a consumption model	Pay only for the computing resources you require; increase and decrease usage with business requirements, not elaborate forecasting
Measure overall efficiency	Measure the business output of the workload against the cost of delivering it, to know the gains from raising output and cutting cost
Stop spending on undifferentiated heavy lifting	AWS racks, stacks, and powers the servers — focus on customers and business projects instead of IT infrastructure
Analyze and attribute expenditure	The cloud makes usage and cost easy to identify and attribute to individual workload owners — measure ROI and give owners the chance to optimize

The foundational questions fall under five best practice areas: **practice cloud financial management, expenditure and usage awareness, cost-effective resources, manage demand and supply resources, and optimize over time**. As with the other pillars, there are tradeoffs: you might prioritize speed — going to market, shipping features, meeting a deadline — over upfront cost optimization, and designing for *higher availability typically costs more*. Identify your true application needs, use empirical data to inform design decisions, and benchmark to establish the most cost-optimal workload over time. The AnyCompany breakout uses COST 2, COST 6, and COST 7.

9.7 The AWS Well-Architected Tool

The pillar-by-pillar AnyCompany review you just performed is exactly how the **AWS Well-Architected Tool** works. Available in the AWS Management Console, it helps you review the state of your workloads and compare them against the latest AWS architectural best practices: you **define your workload, answer a series of questions** across operational excellence, security, reliability, performance efficiency, and cost optimization, and the tool **delivers an action plan** with step-by-step guidance on building better workloads for the cloud.

> **WHAT THE TOOL GIVES YOU**
>
> - A review of workloads against the **latest** AWS architectural best practices
> - Access to the knowledge and best practices used by AWS architects, whenever you need it
> - An **action plan** with step-by-step improvement guidance
> - A **consistent process** to review and measure architectures — feeding next steps, architectural decisions, and corporate governance

9.8 Reliability and high availability

"Everything fails, all the time." — Werner Vogels, CTO, Amazon.com. A Well-Architected best practice is to **plan for failure**: architect applications and workloads to withstand it. Two factors govern that design work — **reliability** and **availability**.

KEY TERMS

Reliability
A measure of the system's ability to provide functionality when the user wants it — the *probability* that the **entire system** (hardware, firmware, and software) functions as intended for a specified period. Think of a car: if the ignition fails, the car is unavailable; if it fails *repeatedly*, the car is unreliable.

MTBF (mean time between failures)
Total time in service ÷ number of failures. Equivalently MTTF + MTTR: mean time to failure (average running time before a failure) plus mean time to repair.

Availability
Normal operation time ÷ total time — the percentage of uptime (time online between failures) over a period, commonly one year. Reduced by *any* abnormal operation, scheduled or unscheduled.

High availability (HA)
The system withstands some measure of degradation while remaining available: downtime is minimized and minimal human intervention is required. A set of system-wide shared resources cooperates to guarantee essential services, restoring them rapidly on failure — often in under a minute.

> COMPUTING MTTF, MTTR, AND MTBF
>
> An application comes online **Monday at noon** and runs normally until it fails **Friday at noon** — time to failure: **96 hours**. You spend Friday noon to Monday noon diagnosing and repairing — time to repair: **72 hours** — then bring it back online, and the cycle repeats weekly. Averaging the cycles: **MTTF = 96 h**, **MTTR = 72 h**, and **MTBF = 96 + 72 = 168 h** (one week). Availability over the cycle is normal operation ÷ total time = 96 ÷ 168 ≈ **57%**.

Availability is quoted as a **number of 9s** — five 9s means 99.999% availability. Requirements vary: the length of acceptable disruption depends on the type of application, and common availability design goals range from 99% up to 99.999%. Each extra 9 shrinks the allowable downtime dramatically (as arithmetic: a year is 8,766 hours, so each tier's maximum yearly disruption follows directly).

TABLE 9.7 Availability tiers – maximum disruption per year (arithmetic on one year)

AVAILABILITY	9S	MAX DISRUPTION PER YEAR
99%	Two 9s	≈ 3.65 days
99.9%	Three 9s	≈ 8.8 hours
99.99%	Four 9s	≈ 53 minutes
99.999%	Five 9s	≈ 5 minutes

Disruptions can't always be predicted, but availability can be **designed in**. Three factors determine an application's overall availability:

TABLE 9.8 The three factors that influence availability

FACTOR	DEFINITION	WATCH OUT
Fault tolerance	Built-in redundancy of components: the app stays operational when components fail. Specialized hardware detects failure (processor, memory, power supply, I/O, storage) and switches instantly to a redundant component	The fault-tolerant model does **not** address software failures — the most common reason for downtime
Scalability	The app accommodates increases in capacity needs without changing design, remaining available and within required performance standards	Contributes to availability but does **not guarantee** it
Recoverability	The process, policies, and procedures for restoring service — quickly and without lost data — after a catastrophic event makes components unavailable or destroys data	It's about restoring *service and data*, not preventing the disaster

> **COMMON PITFALL**
>
> Improving availability usually **increases cost**. Balance the cost of the improvement against the benefit to your users — and be precise about the goal: do you need the application *always alive and reachable*, or *servicing requests within an acceptable level of performance*? Those are different targets with different price tags.

9.9 AWS Trusted Advisor

The Well-Architected Framework guides you while *designing* — understanding risk, finding areas to improve, driving architectural decisions. **AWS Trusted Advisor** picks up the moment you start *implementing*: it's an online tool that provides **real-time guidance** to help you provision resources following AWS best practices. It looks at your **entire AWS environment** and makes recommendations in five categories.

TABLE 9.9 Trusted Advisor's five categories

CATEGORY	WHAT IT RECOMMENDS
Cost Optimization	Examines resource use; recommends eliminating unused and idle resources or making commitments to reserved capacity
Performance	Improves service performance by checking service limits, ensuring you exploit provisioned throughput, and monitoring for overutilized instances
Security	Improves application security by closing gaps, enabling AWS security features, and examining permissions
Fault Tolerance	Increases availability and redundancy via automatic scaling, health checks, Multi-AZ deployments, and backup capabilities
Service Limits	Flags service usage above **80%** of the service limit. Values come from a snapshot, so current usage may differ; limit and usage data can take up to **24 hours** to reflect changes

HOW TO READ A RECOMMENDATION

- Every check carries a **description**, **alert criteria** (what turns the status yellow or red), and a **recommended action**
- The module's activity trains a five-question habit: what is the *status*? what is the *problem*? what *environment details* are given? what is the *best practice*? what is the *recommended action*?

TABLE 9.10 The five checks from the activity – worth knowing on their own

CHECK	FLAGS WHEN...	RECOMMENDED ACTION
MFA on Root Account	MFA is not enabled on the root account	Log in to the root account and activate an MFA device
IAM Password Policy	No password policy is enabled, or at least one content requirement is off (changes apply immediately to *new* users)	Enable the missing requirements; if there's no policy, create and configure one
Security Groups — Unrestricted Access	A rule's source IP has a **/0** suffix on ports other than 25, 80, or 443 — unrestricted access invites hacking, DoS, data loss	Restrict access to only IPs that require it (a specific address = **/32**); delete overly permissive rules after adding restrictive ones
Amazon EBS Snapshots	Yellow: newest snapshot 7–30 days old · Red: older than 30 days, or the volume has **no snapshot** (EBS volumes are replicated, but failures can occur; snapshots persist to S3 for point-in-time recovery)	Create weekly or monthly snapshots of your volumes
Amazon S3 Bucket Logging	Server access logging is not enabled — it's **off by default**; when on, detailed hourly access logs (request type, resources, time and date) go to a bucket you choose	Enable bucket logging for most buckets; needed for security audits and understanding usage patterns

COMMON PITFALL

Don't confuse the two review tools. The **Well-Architected Tool** is a *design-time* review of one workload: you answer the framework's questions and get an action plan. **Trusted Advisor** is a *run-time* watchdog over the whole account: continuous, real-time checks in five fixed categories. Exam wording like “real-time recommendations across your environment” points to Trusted Advisor.

Module summary

- The Well-Architected Framework is a consistent approach to evaluating and implementing cloud architectures — foundational questions plus best practices, built from reviews of thousands of customer architectures.
- Six pillars: operational excellence, security, reliability, performance efficiency, cost optimization, and sustainability (added 2021). Each pillar has its own design principles and best practice areas, and each area maps to reviewer questions.
- Pillar focuses in one breath: run and improve (opex) · protect and monitor (security) · perform as intended, consistently (reliability) · use resources efficiently (performance) · avoid unnecessary costs (cost optimization).

- The AWS Well-Architected Tool operationalizes the review: define a workload, answer the questions, receive a step-by-step action plan — a consistent, measurable process.
- "Everything fails, all the time" — reliability is the probability the whole system (hardware, firmware, software) functions as intended for a specified period; MTBF = total time in service ÷ failures = MTTF + MTTR.
- Availability = normal operation time ÷ total time, counted in 9s (five 9s = 99.999%); a highly available system withstands degradation with minimal downtime and minimal human intervention.
- Fault tolerance (redundancy), scalability (growth without redesign), and recoverability (restoring after catastrophe) determine availability — and more availability generally costs more.
- AWS Trusted Advisor inspects your entire environment in real time and recommends improvements across five categories: cost optimization, performance, security, fault tolerance, and service limits (flagging usage over 80% of a limit).

Review questions

1. **What is the AWS Well-Architected Framework, and where did it come from?**

   Answer: A guide for designing secure, high-performing, resilient, and efficient cloud infrastructure. It packages foundational questions and best practices into a consistent approach for evaluating and implementing architectures, and AWS developed it after reviewing thousands of customer architectures.

2. **How is each pillar of the framework organized internally?**

   Answer: Each pillar contains design principles plus best practice areas; every area aligns to questions a reviewer should ask (e.g. SEC 1: How do you manage credentials and authentication?), each with context and a list of best practices. The full question set is in the framework's Appendix.

3. **Match the principle to its pillar: "stop guessing capacity," "go global in minutes," "adopt a consumption model," "perform operations as code."**

   Answer: Reliability, performance efficiency, cost optimization, and operational excellence respectively. Matching principles to pillars is a classic knowledge-check pattern — learn each pillar's list.

4. **Distinguish reliability from availability, including how each is measured.**

   Answer: Reliability is the probability that the entire system — hardware, firmware, and software — functions as intended for a specified period; it's measured statistically with MTBF (total time in service ÷ failures). Availability is the percentage of time the system operates normally: normal operation time ÷ total time, expressed in 9s.

5. **An app comes online Monday noon, fails Friday noon, and is repaired by the next Monday noon, every week. Compute MTTF, MTTR, and MTBF.**

   Answer: MTTF = 96 hours (Monday noon to Friday noon), MTTR = 72 hours (Friday noon to Monday noon), MTBF = MTTF + MTTR = 168 hours, i.e. one week.

6. **Name the three factors that influence availability and give the caveat attached to each.**

   Answer: Fault tolerance — built-in component redundancy, but the fault-tolerant model doesn't address software failures (the most common downtime cause). Scalability — absorbing capacity growth without redesign, but it contributes to availability rather than guaranteeing it. Recoverability — restoring service and data after a catastrophe; it's about recovery, not prevention. And overall: raising availability usually raises cost.

7. When would you reach for the Well-Architected Tool versus Trusted Advisor?

Answer: The Well-Architected Tool for a design-time review of a specific workload — you answer the framework's pillar questions in the console and get an action plan. Trusted Advisor for continuous, real-time checks of your whole live environment against best practices in five categories (cost optimization, performance, security, fault tolerance, service limits).

---

8. Trusted Advisor shows a red security group check: a rule allows source 0.0.0.0/0 on port 3306. What does it mean and what should you do?

Answer: The /0 suffix means unrestricted access on a port other than 25, 80, or 443 — an invitation for hacking, denial-of-service, and data loss. Restrict the rule to only the IP addresses that need access (a single address uses /32, e.g. 192.0.2.10/32), and delete the overly permissive rule once the restrictive ones exist.
