AWS Well-Architected Framework

Version Framework, November 6, 2024 · verified 2026-11-09

The six Well-Architected pillars, their principles and question IDs, availability and DR math, how reviews and HRIs work, the WA Tool and lenses.

Contents

Mental model 3

Vocabulary 3

The six pillars and trade-offs 5

General design principles 7

Operational excellence and security 8

- Operational excellence (OPS 1–11) 8
- Security (SEC 1–11) 9

Reliability 10

- Disaster recovery 11

Performance efficiency and cost optimization 12
Performance efficiency (PERF 1–5) 12
Cost optimization (COST 1–11) 13
Sustainability 14
Reviews, risks and the Tool 16
Lenses 18
Further reading 19

Mental model

The Well-Architected Framework is not a standard you pass; it is a structured way to surface risk. Every pillar is a set of questions, every question has best practices, and each best practice you don't follow becomes a documented, conscious trade-off or an item in an improvement plan. A review is a conversation at a milestone, not an audit, and its output is a prioritized list of actions.

Vocabulary

Term	Meaning in the Framework
Component	Code, config and AWS resources meeting one requirement
Workload	Components that together deliver business value
Architecture	How components work together in a workload

Term	Meaning in the Framework
Milestone	A key change: design, test, go-live, production
Technology portfolio	All workloads the business needs to operate

The **workload** is the unit of review. It is the level at which business and technology leaders talk, so questions are answered per workload, not per account or per service.

The docs also grade every improvement by **level of effort**: *low* is hours to days, *medium* is days to weeks, *high* is weeks to months. Use it with risk to order the improvement plan.

> **Gotcha:** A component is “often the unit of technical ownership”. Reviewing one microservice in isolation answers the questions for a component, and misses the cross-component failure and cost risks the workload-level review exists to find.

The six pillars and trade-offs

Pillar	Asks, in one line
Operational excellence	Can we run, observe and improve it?
Security	Are data, systems and assets protected?
Reliability	Does it work correctly when expected?
Performance efficiency	Do resources meet demand as it changes?
Cost optimization	Is value delivered at the lowest price?

Pillar	Asks, in one line
Sustainability	Is resource and energy use minimized?

Sustainability was added as the sixth pillar in November 2021; older material that says “five pillars” is out of date. Each pillar has its own **design principles**, **best practice areas** and numbered **questions** (`OPS 1`, `SEC 1`, `REL 1` …).

You trade pillars against each other based on business context: a dev environment may trade reliability for cost, and an e-commerce site may pay for performance because latency costs revenue.

> **Warning:** The Framework states that security and operational excellence are generally **not** traded off against the other pillars. “We’ll add security later” is not a pillar trade-off.

General design principles

Principle	What it replaces
Stop guessing your capacity needs	Fixed upfront capacity
Test systems at production scale	Shared, scaled-down test rigs
Automate with experimentation in mind	Manual, untracked changes
Consider evolutionary architectures	Static, one-time designs
Drive architectures using data	Choices made by opinion
Improve through game days	Untested runbooks

These six apply across every pillar. They all rely on the cloud's ability to create and destroy resources cheaply: you can provision a full-size test environment for an hour, measure, and throw it away.

> **Tip:** A game day is a scheduled simulation of a real event, such as a lost Availability Zone or a traffic spike. It tests the runbooks and the people, not only the architecture.

Operational excellence and security

Operational excellence (OPS 1–11)

Area	Questions
Organization	OPS 1–3: priorities, structure, culture
Prepare	OPS 4–7: observability, flow, deploy risk
Operate	OPS 8–10: use telemetry, health, events
Evolve	OPS 11: how you evolve operations

Its eight design principles: organize teams around business outcomes; implement observability for actionable insights; safely automate where possible; make frequent, small, reversible changes; refine operations procedures frequently; anticipate failure;

learn from all operational events and metrics; use managed services.

Security (SEC 1–11)

Seven areas: security foundations (SEC 1), identity and access management (SEC 2–3), detection (SEC 4), infrastructure protection (SEC 5–6), data protection (SEC 7–9), incident response (SEC 10) and application security (SEC 11).

Its seven design principles: implement a strong identity foundation; maintain traceability; apply security at all layers; automate security best practices; protect data in transit and at rest; keep people away from data; prepare for security events.

> **Gotcha:** SEC 7, “How do you classify your data?”, comes before the questions on encryption at rest and in transit. You can’t pick controls for data you haven’t classified, so an encrypt-everything answer with no classification still leaves SEC 7 open.

Reliability

Availability	Max downtime per year
99%	3 days 15 hours
99.9%	8 hours 45 minutes
99.95%	4 hours 22 minutes
99.99%	52 minutes
99.999%	5 minutes

Availability is the percentage of time a workload performs its agreed function when required. Hard dependencies **multiply**; redundant copies multiply their **failure** rates instead.

```
Serial, three hard deps at 99.99%:
  0.9999 x 0.9999 x 0.9999 = 99.97%
Redundant, two copies at 99.9%:
  1 - (0.001 x 0.001)      = 99.9999%
```

The 13 questions span four areas: foundations (REL 1–2: quotas, network topology), workload architecture (REL

3–5), change management (REL 6–8) and failure management (REL 9–13: backup, fault isolation, withstanding failure, testing, DR). The five principles: automatically recover from failure, test recovery procedures, scale horizontally, stop guessing capacity, manage change through automation.

Disaster recovery

Strategy	Recovery-Region state
Backup and restore	Data backed up; infra redeployed via IaC
Pilot light	Data replicated; core infra on, apps off
Warm standby	Scaled-down but fully working copy
Multi-site active/active	Serves traffic from every Region

Each step down the table costs more and recovers faster. **RTO** is the maximum acceptable delay between interruption and restoration; **RPO** is the maximum

acceptable time since the last recovery point. The business sets both, then you pick the cheapest strategy that meets them.

> **Gotcha:** RTO is not MTTR. MTTR is a mean across many events; RTO is the worst case allowed for a single disaster. DR covers one-time disasters, while availability measures resiliency to everyday component failures. Design for both separately.

Performance efficiency and cost optimization

Performance efficiency (PERF 1–5)

Area	Question
Architecture selection	PERF 1: resources and patterns
Compute and hardware	PERF 2: compute resources

Area	Question
Data management	PERF 3: store, manage, access data
Networking and content delivery	PERF 4: networking resources
Process and culture	PERF 5: process for efficiency

Its five principles: democratize advanced technologies (consume ML or NoSQL as a service), go global in minutes, use serverless architectures, experiment more often, and consider *mechanical sympathy* — pick the technology that fits your data access pattern.

Cost optimization (COST 1–11)

Pricing model	Saving vs On-Demand
On-Demand	Baseline, no minimum commitment
Savings Plans, Reserved Instances	Up to 75%

Pricing model	Saving vs On-Demand
Spot Instances	Up to 90%

Five areas: practice Cloud Financial Management, expenditure and usage awareness, cost-effective resources, manage demand and supply resources, optimize over time. The principles stress measuring business output per dollar and attributing spend to workload owners.

> **Tip:** The docs' "adopt a consumption model" example: stopping dev and test environments outside working hours runs them 40 of 168 weekly hours, cutting their cost by about 75%.

Sustainability

Responsibility	Who	Covers
Sustainability *of* the cloud	AWS	Efficient infra, water, renewables

Responsibility	Who	Covers
Sustainability *in* the cloud	You	Utilization, total resources deployed

The pillar has six areas, one question each: Region selection (SUS 1), alignment to demand (SUS 2), software and architecture (SUS 3), data management (SUS 4), hardware and services (SUS 5), and process and culture (SUS 6). Measure impact **per unit of work**, such as per user or per transaction, so growth doesn't mask gains.

The improvement process is a loop: identify targets, evaluate specific improvements, prioritize and plan, test and validate, deploy to production, measure results and replicate successes.

> **Note:** Most sustainability wins are also cost wins: removing idle resources and right-sizing improve both pillars at once.

Reviews, risks and the Tool

Concept	Meaning
Review	Blame-free conversation, hours not days
HRI	High risk issue: significant business impact
MRI	Medium risk issue: lesser impact
Improvement plan	The HRIs and MRIs, with guidance
Milestone (Tool)	Saved state of a workload at a point in time

Review early in design, to catch *one-way doors* that are hard or impossible to reverse, then again before go-live and after significant changes. Two-way doors get a lightweight process.

The AWS Well-Architected Tool runs the review in the console at no extra charge. Risks come from your answers: each unselected best practice can raise an HRI

or MRI. Save a milestone after the first full review and after every round of improvements to track progress.

```
aws wellarchitected create-workload \
  --workload-name checkout-api \
  --description "Checkout service" \
  --environment PRODUCTION \
  --aws-regions eu-west-1 \
  --review-owner team@example.com \
  --lenses serverless
aws wellarchitected list-lens-review-improvements
\
  --workload-id "$WORKLOAD_ID" \
  --lens-alias serverless \
  --pillar-id security
```

> **Warning:** If you won't implement a best practice, the docs say to record the business reason and approval in the workload notes, or mark the choice not applicable. An HRI left silently open looks identical to one nobody has looked at.

Lenses

Lens type	Source
Framework lens	Applied to every workload automatically
Lens Catalog	AWS-maintained, e.g. Serverless, SaaS
Custom lens	Your own pillars, questions, practices

A lens adds domain-specific questions on top of the Framework. The catalog includes Serverless Applications, SaaS, Generative AI, Machine Learning,

Data Analytics, IoT, Container Build, DevOps, Migration and several industry lenses.

A workload holds at most 20 lenses. A custom lens is authored as a JSON file of up to 500 KB, starts as a `DRAFT`, and must be published before you can apply or share it.

> **Gotcha:** Removing a lens from a workload doesn't delete its answers. Re-add it and the earlier review data comes back, so remove-and-re-add is not a reset.

Further reading

- AWS Well-Architected Framework — the full pillar text, best practices and questions
- Reliability Pillar whitepaper — availability, fault isolation and DR in depth
- Disaster recovery of workloads on AWS — the four DR strategies compared

- AWS Well-Architected Tool user guide — workloads, milestones, lenses and sharing
- AWS CLI wellarchitected reference — every review operation as a command