☀️ THE VACATION EDITION

The DevOps & Monitoring Summer Workbook

Keep your observability skills sharp between two ice creams. Six lessons, hands-on exercises, and one very serious personality quiz.

6

SIX LESSONS, ZERO SUNBURN

Golden signals, observability, percentiles, alerting, SLOs, and incident response — the fundamentals every team relies on, refreshed with schemas you can actually reuse and exercises you can actually finish.

This workbook belongs to: _____

HOW TO USE THIS WORKBOOK

Summer is the perfect time to step back from the alert stream and revisit the fundamentals. This workbook turns the core ideas of modern monitoring into six short lessons — each with a schema to **look at**, a key idea to **remember**, and an exercise to **try**. No stack to install, no laptop required. A deck chair will do.

Work through it in order or cherry-pick the topics you want to sharpen. Every exercise has a worked solution in the **Answer Key** on page 11 — but no peeking until you've had a go.

The Legend

EXERCISE A hands-on drill to test yourself.

EASY MEDIUM HARD Difficulty, from warm-up to expert.

📐 **Schema** — a diagram you may reuse freely.

💡 **Key idea** — the one thing to remember.

⏱️ **Time** — a rough estimate per exercise.

🏖️ **Quiz** — the fun one. No wrong answers.

What's Inside

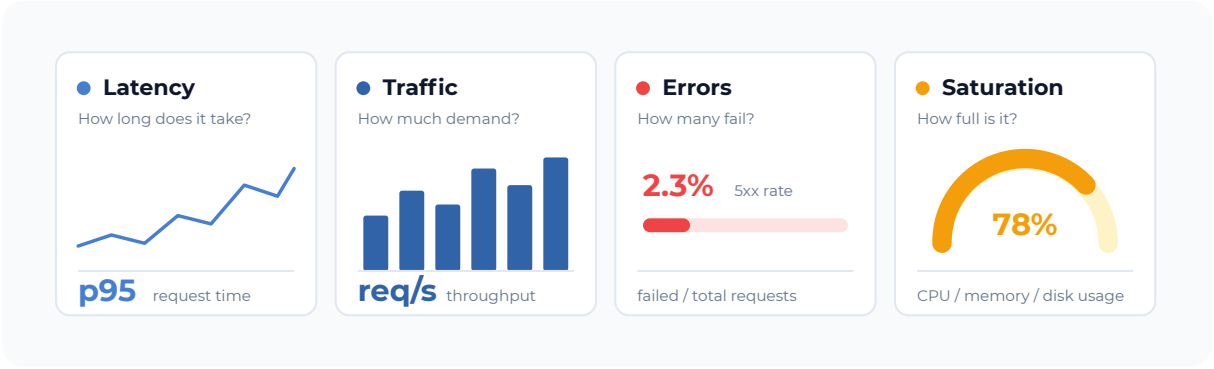
> **The only rule:** be honest about the ones you get wrong. A missed exercise here is a saved incident later.

LESSON 1

The Four Golden Signals

The four numbers that tell you, at a glance, whether a service is healthy.

Google's SRE book distilled service health into four signals. Watch these first; everything else is detail. **Latency** is how long requests take, **Traffic** is how much demand you're serving, **Errors** is how often requests fail, and **Saturation** is how full your resources are. Master these and you can triage almost any service — even one you've never seen before.



Schema 1 — The four Golden Signals as a single dashboard row.

> 💡 **Key idea:** if you can only build four graphs for a service, build these four. Latency and Errors describe what **users feel**; Traffic and Saturation describe what your **system is doing**.

EXERCISE **Match the symptom to the signal** EASY ⏱ 4 min

Write which Golden Signal each symptom points to — **Latency, Traffic, Errors,** or **Saturation**.

A. Users say the app "feels slow", but nothing is actually down. _____

B. The checkout API returns HTTP 500 for 3% of calls. _____

C. Disk usage sits at 94% and keeps climbing. _____

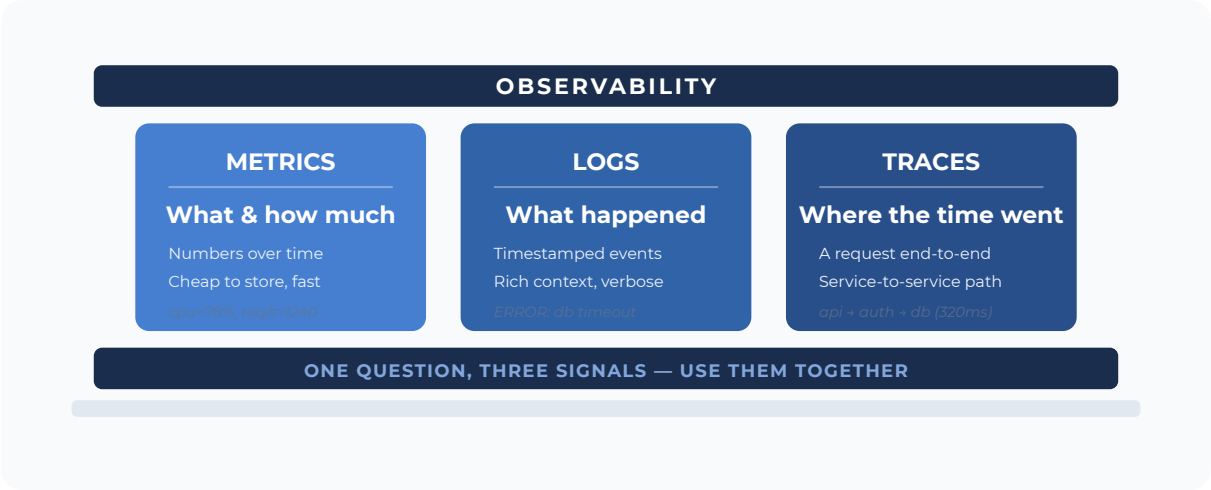
D. Requests jump from 1k to 12k per minute after a newsletter blast. _____

LESSON 2

The Three Pillars of Observability

Metrics, logs, and traces — three angles on the same question: "what is happening?"

Monitoring tells you **when** something is wrong; observability helps you understand **why**. It rests on three data types. **Metrics** are cheap numbers over time, great for dashboards and alerts. **Logs** are timestamped events with rich detail, great for "what exactly happened". **Traces** follow a single request across services, great for "where did the time go". You need all three — each answers questions the others can't.



Schema 2 — The three pillars stand on one foundation: answering "why?".

💡 **Key idea:** metrics tell you **that** something broke, logs tell you **what** broke, and traces tell you **where** it broke. Start at the metric, drill into the trace, confirm in the logs.

EXERCISE Which pillar answers it?

EASY ⏱ 4 min

For each question, write **M** (metrics), **L** (logs), or **T** (traces).

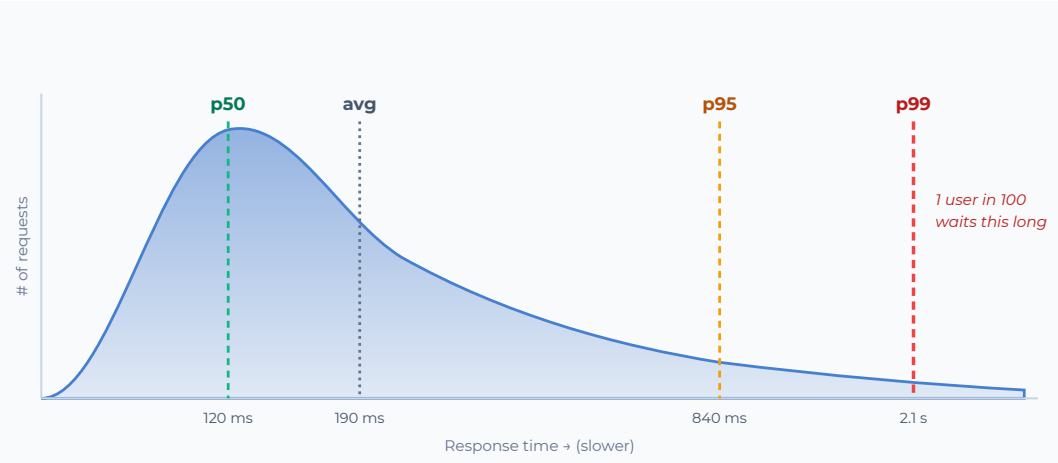
1. Which line of code threw the exception? _____
2. Is CPU usage trending up this week? _____
3. Why did this one checkout take 4 seconds? _____
4. How many requests per second right now? _____
5. What did the auth service log at 14:03? _____
6. Which downstream service added the most delay? _____

LESSON 3

Why Averages Lie

The average hides your worst experiences. Percentiles put them back in view.

If your average response time is 190 ms, are your users happy? You can't tell. An average blends the fast majority with the slow tail into one misleading number. **Percentiles** fix this: the **p95** is the value 95% of requests come in under, and the **p99** the value 99% come in under. The p99 is where your angriest users live — and where a single slow database query hides.



Schema 3 — Same requests, very different stories: p50 vs. the p99 tail.

💡 **Key idea:** alert and report on percentiles, not averages. "p95 latency under 300 ms" is a promise you can keep; "average latency" is a number you can hide behind.

EXERCISE Read the distribution

MEDIUM ⏱ 6 min

Use the numbers from Schema 3: **avg = 190 ms · p50 = 120 ms · p95 = 840 ms · p99 = 2.1 s.**

1. Half of all users get a response faster than _____

2. Out of 1,000 requests, how many are slower than 840 ms? _____

3. Your SLO is "95% of requests under 500 ms." Are you meeting it? _____

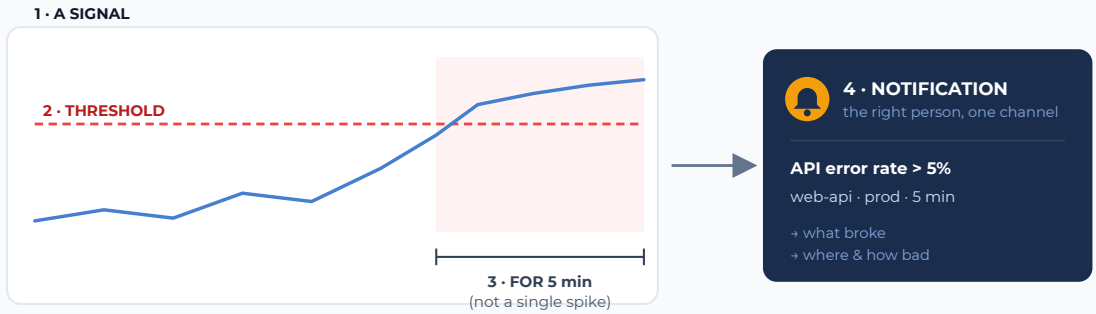
4. In one sentence, why is the 190 ms average misleading here?

LESSON 4

Anatomy of a Good Alert

An alert should wake the right person, at the right time, with the right context — and never otherwise.

The fastest way to be ignored is to alert too often. A good alert has four parts: it watches a **signal** that matters, compares it to a sensible **threshold**, waits long enough (**for a duration**) to avoid reacting to momentary spikes, and delivers an actionable **notification** that says what broke, where, and how bad. Anything else is noise — and noise trains your team to swipe alerts away.



Schema 4 — From a raw signal to an actionable page, in four parts.

> 💡 **Key idea:** alert on **symptoms users feel**, not every internal cause. "Checkout error rate > 5%" beats twenty separate CPU alerts that no one reads.

EXERCISE Good alert or just noise?

MEDIUM ⏱ 5 min

Tick **Good** or **Noise** for each.

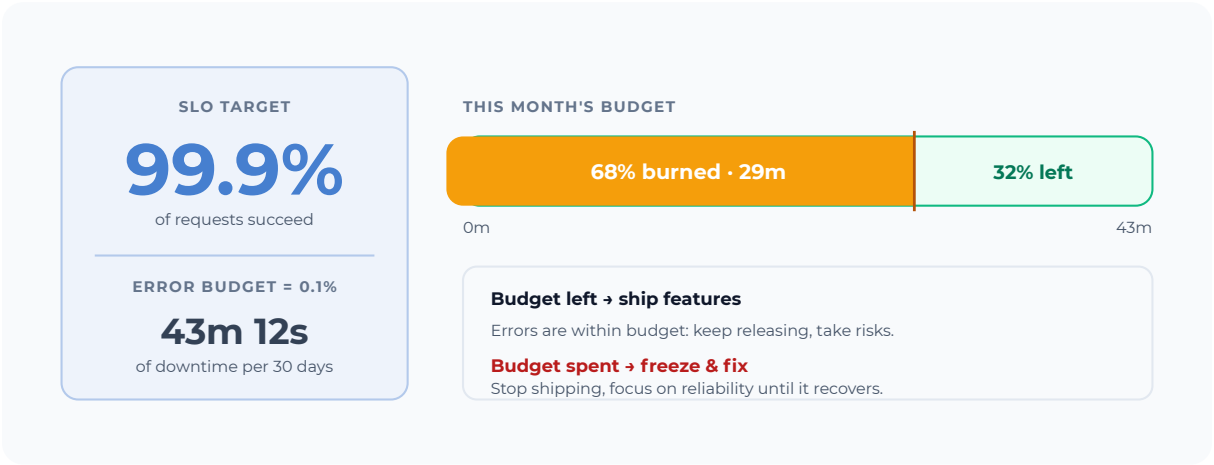
Alert	Good	Noise
a. CPU went above 90% for one 10-second spike at 3am.	☐	☐
b. The alert message reads, in full: "Something is wrong."	☐	☐
c. API 5xx rate > 5% for 5 min, paged to on-call with a dashboard link.	☐	☐
d. Disk will be full in ~4 hours at the current growth rate.	☐	☐
e. Every single failed request pages the entire team.	☐	☐

LESSON 5

SLIs, SLOs & the Error Budget

Turn “be reliable” into a number you can measure — and spend.

An **SLI** (Service Level Indicator) is a measurement, e.g. the percentage of successful requests. An **SLO** (Service Level Objective) is the target you set for it, e.g. 99.9% over 30 days. The gap between 100% and your SLO is your **error budget**: the amount of unreliability you’re allowed. Budget left means you can ship and take risks; budget spent means you stop and stabilise. It turns reliability from an argument into arithmetic.



Schema 5 — A 99.9% SLO buys you ~43 minutes of downtime a month. Spend it wisely.

💡 **Key idea:** an SLO of 100% is a trap — it forbids all change. A realistic SLO gives you a budget for risk, and a clear signal for when to slow down.

EXERCISE Do the budget maths

HARD ⏱ 7 min

Downtime allowed = **(1 − SLO) × time period.** A 30-day month has 43,200 minutes.

1. How many minutes of downtime does a **99.9%** monthly SLO allow? _____

2. And a stricter **99.99%** SLO? _____

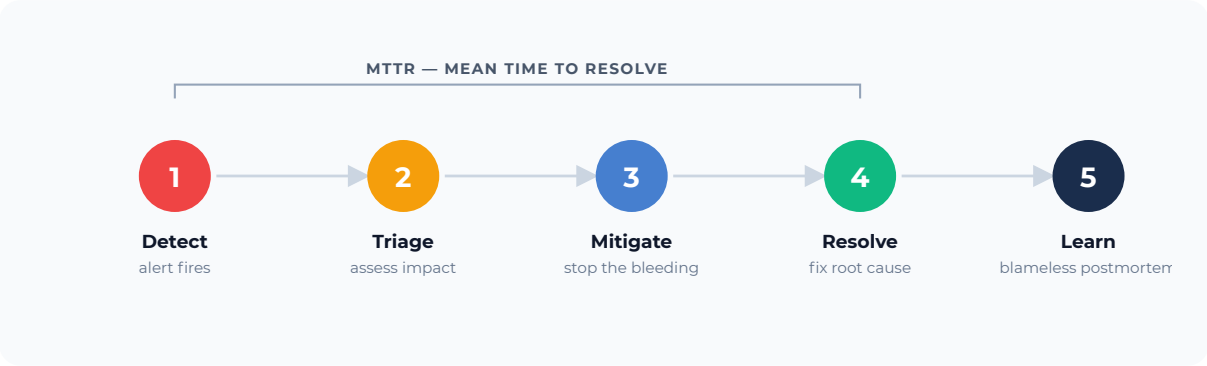
3. It’s day 20 of 30 and you’ve already burned 30 of your 43 minutes. Do you ship the risky release today, or wait? Why?

LESSON 6

The Incident Lifecycle

Every incident follows the same arc. Knowing the steps keeps a bad night from getting worse.

When something breaks at 2am, a clear sequence beats heroics. First you **detect** (an alert fires). You **triage** (who and what is affected, how badly). You **mitigate** to stop customer pain — often a rollback, long before you understand the cause. Then you **resolve** the true root cause, and finally you **learn** with a blameless postmortem. Two numbers keep you honest: **MTTA** (time to acknowledge) and **MTTR** (time to resolve).



Schema 6 — Detect, triage, mitigate, resolve, learn.

> 💡 **Key idea:** mitigate before you diagnose. Stopping the bleeding (a rollback) comes **first**; the perfect root-cause fix can wait until customers are safe.

EXERCISE **Put the incident in order** MEDIUM ⏱ 4 min

Number these 1 to 5 in the order they should happen.

_____ Write a blameless postmortem so it can't happen the same way twice.

_____ A PagerDuty alert wakes the on-call engineer.

_____ Roll back the bad deploy to stop customer impact.

_____ Assess which services are affected and how many users.

_____ Identify and fix the root cause in the code.

🏖️ THE BIG QUIZ

What Kind of SRE Are You?

Six questions, four possible answers each. Circle one per question, then tally your symbols. No wrong answers — but your on-call rotation may disagree.

1. It's 2am and your phone buzzes. Your first move?

- 🚒 Grab the laptop — I live for a good fire.
- 📊 Open the dashboard and study the graphs.
- 🧘 Check whether auto-remediation already fixed it.
- 🛡️ Open the runbook and follow it step by step.

2. A brand-new service goes live. You first…

- 📊 build it a beautiful dashboard.
- 🛡️ define its SLOs and alert rules.
- 🧘 automate its deploy and self-healing.
- 🚒 wait for it to break, then learn it.

3. Your ideal Friday-afternoon task?

- 🧘 Deleting a manual process forever.
- 📊 Polishing a Grafana dashboard.
- 🛡️ Auditing alert rules for noise.
- 🚒 Nothing — Fridays are when things break.

4. "It works on my machine." You reply…

- 🛡️ "Then let's add a test and an alert."
- 🚒 "Cool — let's watch it fall over in prod."
- 📊 "Show me the metrics."
- 🧘 "The pipeline will tell us in 3 minutes."

5. Pick your monitoring philosophy:

- 🛡️ Prevent every incident before it happens.
- 🚒 You can't prevent chaos, only respond fast.
- 📊 If it isn't measured, it doesn't exist.
- 🧘 The best incident is the one that self-heals.

6. Your idea of the perfect alert…

- 🧘 resolves itself before a human sees it.
- 🛡️ is rare, actionable, and documented.
- 📊 links straight to the right dashboard.
- 🚒 means it's game time.

Tally your symbols, then find your type:

🚒 **Mostly fire trucks — The Firefighter.** Calm in chaos, unbeatable under pressure. Your gift is response; your growth is prevention. Automate one recurring fire away.

🧘 **Mostly lotus — The Zen Master.** You'd rather delete a problem than fix it twice. Automation is your love language — just remember to monitor the automation too.

📊 **Mostly charts — The Dashboard Artist.** Every metric has a home and a colour. Beautiful visibility — just make sure the prettiest graph is also the one that pages you.

🛡️ **Mostly shields — The Guardian.** SLOs, runbooks, and quiet nights. Prevention-first and proud. Watch that your caution still leaves room to ship.

GRADUATION CHALLENGE

Final Exam

Eight questions pulled from all six lessons. No schema to peek at this time. Answers are on the next page — score yourself out of 8.

EXERCISE **The whole workbook, in eight questions** HARD 12 min

1. True or false: the average response time is the best single number for judging user experience. _____

2. Name the four Golden Signals.

3. Metrics, logs, and _____ are the three pillars of observability.

4. A 99.9% monthly SLO allows roughly how many minutes of downtime per 30 days? _____

5. True or false: a good alert should fire on every single failed request. _____

6. Put the incident stages in order — **Mitigate, Detect, Learn, Triage, Resolve:**

7. Your p99 latency is 2 s but your p50 is 120 ms. In one line, what does that tell you?

8. True or false: when the error budget is exhausted, keep shipping risky features. _____

ANSWER KEY

No cheating credit for answers filled in after reading this page.

Lesson 1 — Golden Signals

A. Latency · **B.** Errors · **C.** Saturation · **D.** Traffic.

Lesson 2 — Three Pillars

1. L · **2.** M · **3.** T · **4.** M · **5.** L · **6.** T.

Lesson 3 — Percentiles

1. 120 ms (the p50).
2. 50 requests (5% are slower than the p95).
3. No — your p95 is 840 ms, well over the 500 ms target.
4. The average blends a fast majority with a slow tail, so it hides the slow p95/p99 that real users actually feel.

Lesson 4 — Alerts

a. Noise (a momentary spike, no sustained impact).
b. Noise (no context, not actionable).
c. Good (symptom-based, sustained, actionable).
d. Good (predictive and actionable).
e. Noise (per-request paging is alert fatigue).

Lesson 5 — Error Budget

1. ~43 min (0.1% × 43,200 = 43.2 min).
2. ~4.3 min (0.01% × 43,200 = 4.32 min).
3. Wait. At day 20 of 30 you'd expect ~29 min spent if burning evenly; you're at 30 and nearly out. Hold the risky release and protect what's left.

Lesson 6 — Incident Order

1. Alert wakes on-call (Detect) · **2.** Assess impact (Triage) · **3.** Roll back (Mitigate) · **4.** Fix root cause (Resolve) · **5.** Postmortem (Learn).

Final Exam

1. False.
2. Latency, Traffic, Errors, Saturation.
3. Traces.
4. ~43 minutes.
5. False.
6. Detect ⟶ Triage ⟶ Mitigate ⟶ Resolve ⟶ Learn.
7. Most requests are fast, but a small fraction (~1%) are very slow — a long tail worth investigating.
8. False — freeze risky changes and focus on reliability.

> **7–8 correct:** you're ready for on-call. **4–6:** solid — revisit the ones you missed. **0–3:** perfect, that's what the summer's for.

CONGRATULATIONS

You've refreshed the fundamentals that every reliable team leans on — golden signals, the three pillars, percentiles, alerting, error budgets, and incident response. The reward for finishing your homework? A monitoring stack that applies all of this **for you**, without the maintenance.

🎓 Your monitoring badge

Finished every exercise? You've earned it. Share your SRE type from the Big Quiz with your team — and settle, once and for all, who the Firefighter really is.