Technical Track: Speaker Notes

Reliability Unleashed - The Engineering Playbook

Presenters (Technical Track)   February 2026

Technical Track: Speaker Notes

Presentation: Reliability Unleashed - The Engineering Playbook **Duration:** ~8.5 hours (full workshop) or modular by part **Audience:** Engineers, SRE practitioners, technical architects **Slides:** 93 across 9 parts **One-Pagers:** 34 detailed reference documents

How to Use These Notes

Each slide includes:

- **[SLIDE CONTENT]** - What the audience sees on screen
- **[SPEAKER SCRIPT]** - The actual words to deliver (crafted talk, not bullets)
- **[STAGE DIRECTIONS]** - Pauses, gestures, animations, transitions

Timing: Aim for 2-3 minutes per content slide, 30 seconds for transitions.

PART 1: Foundation & Vision

Duration: 45-60 minutes **Reference One-Pagers:** reliability-unleashed, sre-foundations, dora-24-capabilities, sre-maturity-assessment

Slide 1: Title Slide

[SLIDE CONTENT]

```
┌──────────────────────────────────────────────────────┐
│                                                      │
│           RELIABILITY UNLEASHED                      │
│         From Chaos to Confidence                     │
│                                                      │
│           The Engineering Playbook                   │
│                                                      │
│        Technical Operations Excellence               │
│                                                      │
│   93 Slides | 9 Parts | 34 One-Pagers               │
│                                                      │
└──────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

Welcome to Reliability Unleashed.

[Pause, make eye contact with audience]

This isn't another abstract talk about DevOps philosophy. This is the engineering playbook - the specific practices, patterns, and tools that transform chaotic operations into confident, reliable systems.

We've distilled insights from Google's SRE practices, Netflix's chaos engineering, NASA's failure analysis methods, and high-reliability organizations like nuclear plants and aircraft carriers. But more importantly, we've implemented these practices ourselves while building an AI agent army that operates 24/7.

[Gesture to screen]

Over the next eight hours - or in whichever parts you choose - we'll cover 93 slides organized into 9 progressive parts. Each part maps to detailed one-pagers you can take home as reference material.

Let's begin the journey from chaos to confidence.

[STAGE DIRECTIONS]

- Arrive early, test slides and audio
- Stand centered, confident posture
- First pause after "engineering playbook" - let it land
- End with energy: "Let's begin..."

Slide 2: The Journey Ahead (Agenda)

[SLIDE CONTENT]

```
┌──────────────────────────────────────────┐
│  THE JOURNEY AHEAD                         │
│  ─────────────────                         │
│                                            │
│  ┌────────┐ ┌────────┐ ┌────────┐         │
│  │ Part 1 │ │ Part 2 │ │ Part 3 │         │
│  │Foundation│ │Observ. │ │Resilience│         │
│  └────────┘ └────────┘ └────────┘         │
│                                            │
│  ┌────────┐ ┌────────┐ ┌────────┐         │
│  │ Part 4 │ │ Part 5 │ │ Part 6 │         │
│  │Incidents│ │ Release │ │ Cloud  │         │
│  └────────┘ └────────┘ └────────┘         │
│                                            │
│  ┌────────┐ ┌────────┐ ┌────────┐         │
│  │ Part 7 │ │ Part 8 │ │ Part 9 │         │
│  │ AI/ML  │ │ Culture │ │ Roadmap │         │
│  └────────┘ └────────┘ └────────┘         │
│                                            │
└──────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

Here's our roadmap. Nine parts, each building on the previous.

[Point to grid systematically]

Parts 1 through 4 are your foundation. We start with the "why" - SLIs, SLOs, error budgets, and DORA metrics. Then we build observability - because you can't improve what you can't measure. We cover resilience patterns - circuit breakers, retries, bulkheads - the building blocks of fault tolerance. And we master incident response - because failures will happen, and your response determines your reputation.

[Transition gesture]

Parts 5 and 6 tackle infrastructure. Release engineering, testing strategies, capacity planning. Cloud-native patterns and Kubernetes reliability. This is where theory meets deployment.

[Energy increase]

Parts 7 through 9 are the frontier. AI/ML operations - because machine learning systems have unique reliability challenges. Culture and people - because tools without culture change nothing. And finally, your implementation roadmap - how to take these concepts back to your teams.

You can do this workshop end-to-end, or pick the parts most relevant to your current challenges.

[STAGE DIRECTIONS]

- Walk through grid top-to-bottom
- Brief pause between each trilogy (1-4, 5-6, 7-9)
- Increase energy on "frontier" section
- End with eye contact: "your current challenges"

Slide 3: Why SRE? Why Now?

[SLIDE CONTENT]

```
┌─────────────────────────────────────────────────────┐
│  WHY SRE? WHY NOW?                                   │
│  ─────────────────                                   │
│                                                      │
│    ┌──────────┐  ┌──────────┐  ┌──────────┐        │
│    │   4+     │  │  24/7    │  │  100s    │        │
│    │ AI Agents│  │Operations│  │ Commits  │        │
│    │production│  │never sleep│ │  daily   │        │
│    └──────────┘  └──────────┘  └──────────┘        │
│                                                      │
│   ┌─────────────────────────────────────────────┐  │
│   │ "Bots don't get tired. But they can fail." │    │
│   │                                             │    │
│   │  And when they do, who responds at 3 AM?   │    │
│   └─────────────────────────────────────────────┘  │
│                                                      │
└─────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

Let me tell you why this matters right now.

[Point to stats]

We have 4 AI agents running in production. They work 24/7 - no lunch breaks, no vacations. They generate hundreds of commits every day across multiple repositories.

[Pause, lower voice slightly]

Sounds amazing, right? But here's the thing.

[Pause for effect]

Bots don't get tired. But they can fail.

They can hit rate limits. They can get stuck in loops. They can corrupt state. They can produce subtly wrong outputs that compound over time.

[Look at audience directly]

So when Claude hits an API timeout at 3 AM on a Sunday... who responds?

[Longer pause]

That's the question SRE answers. Not just "how do we build reliable systems" - but "how do we operate them reliably at scale, around the clock, with finite human resources?"

This isn't theoretical. This is operational survival.

[STAGE DIRECTIONS]

- Stats appear immediately - gesture to each
- Drop tone on "But they can fail"
- The 3 AM question is the climax - pause, make eye contact
- End with gravity: "operational survival"

Slide 4: DevOps vs SRE

[SLIDE CONTENT]

```
┌──────────────────────────────────────────────────────┐
│  DevOps vs SRE                                       │
│  ─────────────                                       │
│                                                      │
│  ╔════════════════════════════════════════════════╗  │
│  ║  class SRE implements interface DevOps { }     ║  │
│  ╚════════════════════════════════════════════════╝  │
│                                                      │
│  DEVOPS                    SRE                       │
│  ──────                    ───                       │
│  Philosophy                Engineering Discipline    │
│  Break down silos          Error budgets             │
│  Continuous delivery       SLOs and SLIs             │
│  Automation culture        Toil reduction targets    │
│  Shared responsibility     On-call engineering       │
│                                                      │
│  DevOps is WHAT. SRE is HOW.                         │
│                                                      │
└──────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

Now, some of you are thinking - "isn't this just DevOps with a different name?"

[Pause, smile]

Fair question. Let me clarify.

[Gesture to code banner]

Google's Ben Treynor Sloss said it best: "class SRE implements interface DevOps."

If you're a programmer, this makes intuitive sense. DevOps is the interface - the contract, the philosophy, the "what you should do." Break down silos. Ship continuously. Automate everything. Share responsibility.

[Gesture to comparison]

SRE is one concrete implementation of that interface. It adds specific engineering practices:

- Instead of "prioritize reliability," SRE gives you error budgets - a mathematical formula for balancing innovation and stability.

- Instead of "measure what matters," SRE defines SLIs and SLOs - precise indicators tied to user experience.
- Instead of "reduce toil," SRE sets a 50% cap - spend no more than half your time on repetitive tasks.

[Emphasize]

DevOps tells you WHAT to do. SRE tells you HOW to do it - with engineering rigor and measurable outcomes.

[STAGE DIRECTIONS]

- Acknowledge the skepticism with good humor
- Code banner appears first - pause for recognition
- Walk through comparison methodically
- Strong emphasis on "HOW" at the end

Slide 5: Three Pillars of Operations

[SLIDE CONTENT]

```
┌───────────────────────────────────────────────────────┐
│  THREE PILLARS OF OPERATIONS                          │
│  ───────────────────────────                          │
│                                                       │
│      REACTIVE          PROACTIVE          PREDICTIVE  │
│        (Red)           (Violet)           (Emerald)   │
│                                                       │
│    ┌────────────┐   ┌────────────┐   ┌────────────┐  │
│    │Alert triage│   │SLO monitor │   │Anomaly     │  │
│    │Runbooks    │   │Capacity    │   │detection   │  │
│    │Incident    │   │planning    │   │Chaos eng   │  │
│    │response    │   │Automation  │   │ML insights │  │
│    └────────────┘   └────────────┘   └────────────┘  │
│                                                       │
│   Emergency          Prevention         Intelligence  │
│   response           engineering        systems       │
│                                                       │
│         ◄──────── Maturity Journey ────────►         │
│                                                       │
└───────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

Operations exists on a maturity spectrum. Let me show you where most organizations are - and where they need to go.

[Point to Reactive pillar]

Reactive operations is where everyone starts. The pager goes off. You open a runbook. You follow the steps. Fire drill, fire drill, fire drill. It's necessary - you need good incident response. But it's exhausting, it doesn't scale, and it's always behind the problem.

[Point to Proactive pillar]

Proactive operations is where good teams get to. You're monitoring SLOs before customers complain. You're doing capacity planning before you run out of headroom. You're automating the toil before engineers burn out. This is the plateau most organizations target - and it's a worthy goal.

[Point to Predictive pillar, increase energy]

But the frontier - predictive operations - this is where the magic happens. Anomaly detection catches problems before alerts fire. Chaos engineering finds weaknesses before production does. Machine learning insights reveal patterns humans can't see.

[Gesture across spectrum]

The journey is from reactive to proactive to predictive. You can't skip steps - you need solid incident response before you can prevent incidents, and you need prevention before you can predict. But the goal is to shift your investment left, from fighting fires to preventing them to predicting them.

[STAGE DIRECTIONS]

- Color-code each pillar as discussed
- Reactive: empathetic tone (we've all been there)
- Proactive: respectful (this is solid work)
- Predictive: excited (this is the future)
- Arrow gesture at end

Slide 6: The Vision

[SLIDE CONTENT]

```
┌───────────────────────────────────────────────────────┐
│  THE VISION                                           │
│  ──────────                                           │
│                                                       │
│  "Bots that monitor, diagnose, remediate, and learn"│
│                                                       │
│  ┌─────────────────────────────────────────────────┐ │
│  │  TARGET STATE                                   │ │
│  │  ────────────                                   │ │
│  │                                                 │ │
│  │  70% auto-resolved incidents                    │ │
│  │  <15 min MTTR for remaining 30%               │ │
│  │  99.95% availability                          │ │
│  │  Zero-toil SRE team (engineering focus)       │ │
│  │                                                 │ │
│  └─────────────────────────────────────────────────┘ │
│                                                       │
│  Humans: Strategy, novel situations, creativity     │
│  Bots: Routine, repetitive, rapid response          │
│                                                       │
└───────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

So where are we heading? What's the end state we're building toward?

[Pause for weight]

The vision is simple: bots that monitor, diagnose, remediate, and learn.

Not bots that page humans. Bots that actually fix things. Bots that see a disk filling up and clear the cache. Bots that detect an anomaly and roll back a deployment. Bots that identify a memory leak and restart the affected pods.

[Point to metrics]

These aren't fantasy numbers. With proper automation:

- 70% of incidents can be auto-resolved without human intervention
- The remaining 30% hit a human who has context and tooling to resolve in under 15 minutes
- Sustained 99.95% availability becomes achievable

- And your SRE team spends zero time on toil - all engineering, all the time

[Contrast gesture]

The human role doesn't disappear - it elevates. Humans handle strategy. Novel situations that automation hasn't seen. Creative problem-solving. Architectural decisions.

Bots handle the routine. The repetitive. The 3 AM pages.

[Pause]

That's the division of labor we're building toward.

[STAGE DIRECTIONS]

- Quote appears first - let it resonate
- Metrics are aspirational but grounded - confident tone
- Human/bot contrast is the key insight
- End with conviction

Slide 7: SLIs - Measuring What Users Experience

[SLIDE CONTENT]

```
┌───────────────────────────────────────────────────────┐
│  Service Level INDICATORS                             │
│  ─────────────────────────                            │
│                                                       │
│  "SLIs quantify how users actually experience        │
│   your service, not how your servers feel"           │
│                                                       │
│  ┌──────────────────────────────────────────────────┐│
│  │  THE FOUR GOLDEN SIGNALS                         ││
│  │                                                  ││
│  │  LATENCY    Response time for requests         ││
│  │  TRAFFIC    Request volume / load              ││
│  │  ERRORS     Failed request rate                ││
│  │  SATURATION Resource utilization              ││
│  │                                                  ││
│  └──────────────────────────────────────────────────┘│
│                                                       │
│  SLI Formula: (good events / total events) × 100%   │
│                                                       │
└───────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

Let's get into specifics. SRE starts with measurement, and measurement starts with SLIs - Service Level Indicators.

[Pause for quote]

Here's the critical insight: SLIs measure how users experience your service, not how your servers feel.

Your servers could be running at 10% CPU, plenty of memory, no errors in the logs - and your users could still be having a terrible experience. Why? Maybe a network hop is slow. Maybe a third-party API is timing out. Maybe the request queue is backing up.

[Point to Golden Signals]

Google distilled decades of operational wisdom into four golden signals:

Latency - how long does it take to get a response? This is the metric users feel most directly. Milliseconds matter.

Traffic - how much load are you handling? This provides context for everything else and early warning of capacity issues.

Errors - what percentage of requests fail? Not all failures are equal - distinguish client errors from server errors.

Saturation - how full are your resources? CPU, memory, disk, connections - the signals that predict future problems.

[Point to formula]

And the formula is simple: good events divided by total events. If 9,950 out of 10,000 requests succeed, your SLI is 99.5%.

[STAGE DIRECTIONS]

- Quote is the key insight - pause after
- Walk through signals with examples from the audience's context
- Formula should feel simple and approachable

Slide 8: SLOs - Your Reliability Contract

[SLIDE CONTENT]

```
┌───────────────────────────────────────────────────────┐
│  Service Level OBJECTIVES                             │
│  ─────────────────────────                            │
│                                                       │
│  ┌─────────────────────────────────────────────────┐│
│  │  "SLOs are promises to your users"               ││
│  │                                                  ││
│  │  Too high → Can't innovate (always fixing)     ││
│  │  Too low  → Users leave (poor experience)      ││
│  │  Just right → Sustainable excellence           ││
│  └─────────────────────────────────────────────────┘│
│                                                       │
│  EXAMPLE SLOs:                                        │
│                                                       │
│  API Latency     p99 < 200ms      (99.9% of time)  │
│  Availability    Requests succeed (99.95% monthly) │
│  Error Rate      Failures < 0.1%  (per rolling hour│
│                                                       │
│  SLO = SLI + Target + Time Window                   │
│                                                       │
└───────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

SLOs - Service Level Objectives - are where strategy meets measurement.

[Point to quote]

Think of SLOs as promises to your users. Explicit commitments about the reliability they can expect.

[Explain the tension]

Here's the challenge: set your SLO too high - say, 99.99% availability - and you'll spend all your time preventing that one failure instead of building new features. You become defensive, risk-averse, paralyzed.

Set your SLO too low - say, 95% availability - and your users leave. Five percent failure rate means one in twenty requests fails. Nobody tolerates that for a critical service.

The art is finding the right level - the minimum reliability that keeps users happy, which gives you the maximum room to innovate.

[Walk through examples]

A typical SLO might be: "99.9% of requests complete in under 200 milliseconds, measured over each calendar month." Notice the components - the SLI (latency), the target (99.9% under 200ms), and the time window (monthly).

[Emphasize formula]

SLO equals SLI plus target plus time window. Get these right, and you have a shared language for discussing reliability across product, engineering, and leadership.

[STAGE DIRECTIONS]

- Too high/too low tension is key - use hand gestures
- Examples should feel realistic
- Formula at end ties it together

Slide 9: Error Budgets - Permission to Innovate

[SLIDE CONTENT]

```
┌──────────────────────────────────────────────────────┐
│  ERROR BUDGETS                                       │
│  ─────────────                                       │
│                                                      │
│  "100% reliability is impossible and undesirable"   │
│                                                      │
│  ┌──────────────────────────────────────────────────┐│
│  │  99.9% SLO = 0.1% Error Budget                   ││
│  │                                                  ││
│  │  43 minutes of downtime per month                ││
│  │                                                  ││
│  │  This is your INNOVATION BUDGET                  ││
│  │                                                  ││
│  └──────────────────────────────────────────────────┘│
│                                                      │
│  Budget remaining → Ship faster, take risks         │
│  Budget depleted  → Freeze features, fix stability  │
│                                                      │
│  Error budgets turn reliability from a battle       │
│  into a negotiation.                                │
│                                                      │
└──────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

Now for the concept that transforms everything: error budgets.

[Pause for quote]

"100% reliability is impossible and undesirable."

Impossible because physics. Networks fail. Disks die. Code has bugs. You cannot achieve perfection.

Undesirable because opportunity cost. Every hour spent preventing the last 0.01% of failures is an hour not spent building features users want.

[Point to calculation]

So if your SLO is 99.9%, you have a 0.1% error budget. For a month, that's 43 minutes of downtime you're allowed to use. Not that you want downtime - but you have permission to spend that budget on innovation risk.

[Explain dynamics]

When your error budget is healthy, you ship faster. You experiment. You take calculated risks. You deploy that new feature even though it's a little scary.

When your error budget is depleted, you freeze. No new features until you've paid back the reliability debt. All hands on stability.

[Key insight]

This turns reliability from a war between dev and ops into a negotiation. Both sides want the same thing - a healthy error budget. Dev wants it to ship features. Ops wants it to avoid pages. Alignment through shared incentives.

[STAGE DIRECTIONS]

- "Impossible and undesirable" - let it sink in
- 43 minutes should feel concrete and manageable
- Dev/Ops alignment is the transformation

Slide 10: DORA Metrics - Industry Benchmarks

[SLIDE CONTENT]

```
┌───────────────────────────────────────────────────────┐
│  DORA METRICS                                         │
│  ─────────────                                        │
│  DevOps Research & Assessment - 10 years of data     │
│                                                       │
│  THROUGHPUT                    STABILITY              │
│  ───────────                   ─────────              │
│  Deployment Frequency          Change Failure Rate    │
│  Lead Time for Changes         Time to Restore        │
│                                                       │
│  ┌─────────────────────────────────────────────────┐ │
│  │  ELITE PERFORMERS (2023 data)                   ││
│  │                                                 ││
│  │  Deploy: Multiple times per day                 ││
│  │  Lead time: Less than one day                   ││
│  │  Restore: Less than one hour                    ││
│  │  Change failure: <5%                            ││
│  │                                                 ││
│  └─────────────────────────────────────────────────┘ │
│                                                       │
│  Elite teams move FAST and STABLE. Not either/or.    │
│                                                       │
└───────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

How do you know if you're doing well? DORA gives you the answer.

[Explain context]

DORA - DevOps Research and Assessment - has studied thousands of organizations over ten years. They've identified the metrics that actually correlate with high performance.

[Point to quadrants]

Four metrics, two dimensions:

Throughput - how fast can you ship? Measured by deployment frequency and lead time for changes. How often does code reach production, and how long does it take from commit to deploy?

Stability - how reliably can you ship? Measured by change failure rate and time to restore. What percentage of deployments cause problems, and how fast do you fix them?

[Point to elite benchmarks]

Elite performers - the top tier - deploy multiple times a day, with lead times under one day, restore service in under an hour when things break, and have change failure rates under 5%.

[Key insight]

The critical insight: elite teams are both fast AND stable. They don't sacrifice one for the other. They've figured out how to move quickly while maintaining reliability. That's the goal. Not choosing between speed and stability - achieving both.

[STAGE DIRECTIONS]

- DORA gives credibility - mention the research
- Quadrant structure is important
- "Fast AND stable" is the insight - emphasize

[Continue for remaining 83 slides...]

PART 2: Observability Mastery

Duration: 60-75 minutes **Reference One-Pagers:** observability-mastery, multi-window-alerting, use-method-performance, observability-2.0, alert-tuning-playbook

---

Slide 11: Part 2 Divider

[SLIDE CONTENT]

```
┌───────────────────────────────────────────────────────┐
│                    Part 2 of 9                        │
│                                                       │
│            OBSERVABILITY MASTERY                      │
│                                                       │
│   The Three Pillars, High-Cardinality Events,        │
│   Multi-Window Alerting, USE/RED Methods             │
│                                                       │
│   One-Pagers: observability-mastery,                 │
│               multi-window-alerting,                  │
│               use-method-performance,                 │
│               observability-2.0,                      │
│               alert-tuning-playbook                   │
│                                                       │
└───────────────────────────────────────────────────────┘
```

[SPEAKER SCRIPT]

[Transition energy]

Part 2: Observability Mastery.

If Part 1 was about "why" - the philosophy and metrics of reliability - Part 2 is about "what you see."

You cannot improve what you cannot measure. You cannot debug what you cannot see. Observability is the foundation of operational excellence.

We'll cover the three pillars - logs, metrics, and traces - and why all three matter. We'll introduce high-cardinality events - the next evolution in observability. We'll master alerting - when to page, how to avoid fatigue. And we'll learn two powerful frameworks: USE for infrastructure and RED for services.

[Reference one-pagers]

You have five one-pagers for this section. Observability Mastery covers the three pillars. Multi-Window Alerting explains burn-rate alerting. USE Method Performance gives you the infrastructure framework. Observability 2.0 introduces high-cardinality events. And Alert Tuning Playbook helps you reduce noise.

Let's make your systems visible.

[STAGE DIRECTIONS]

- Part dividers are energy resets
- List one-pagers clearly

- End with invitation: "Let's make your systems visible"

[Notes continue for all 93 slides with the same format...]

Appendix: Quick Reference

Timing Guidelines

CONTENT TYPE	TIME
Part divider	30 seconds
Content slide	2-3 minutes
Demo slide	5-7 minutes
Discussion slide	5-10 minutes

Common Transitions

- "Let's move from WHY to HOW..."
- "Now that we understand X, let's see it in action..."
- "Before we go deeper, let me give you the big picture..."
- "This brings us to a critical question..."
- "Building on that foundation..."

Engagement Techniques

- **Pause for impact** - After key insights, count to 3
- **Eye contact** - Different sections of room
- **Questions** - "How many of you have experienced..."
- **Stories** - Real incidents, anonymized if needed
- **Humor** - Self-deprecating, not dismissive

One-Pager Distribution

At each Part divider, remind audience: "The one-pagers for this section are [list]. You can download them now or get them at the end."

Full speaker notes for all 93 slides available in the complete document.

Bot Army Engineering | Technical Operations Excellence