Logging in the Age of Cost-Cutting: Smart Strategies to Reduce Bills

In an era where cloud costs face intense scrutiny, logging has emerged as one of the biggest cost drivers for engineering organizations. Today, we'll explore how industry leaders like Uber, Airbnb, and Slack have achieved dramatic cost reductions in some cases up to 70% without sacrificing their ability to debug and monitor systems effectively.

By **Venkata Madhu Prateek Reddy Kambala**, Sr. DevOps Engineer

The Logging Cost Crisis

70%

Unused Logs

Of logs are never queried

$50K

Monthly Cost

Typical escalation from $10K in just 12 months

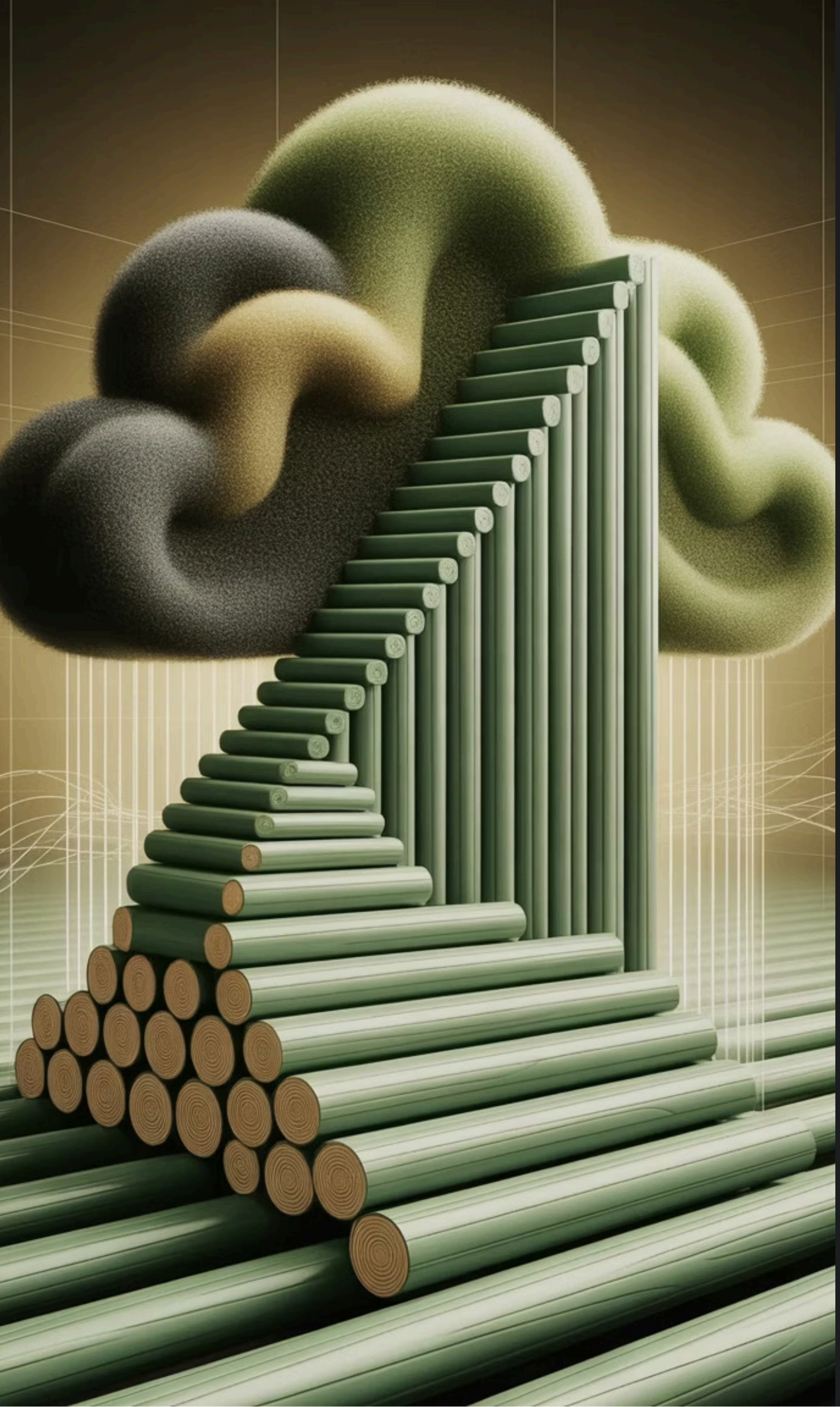
$1.2M

Wasted Annually

Airbnb's spend on unused debug logs

The logging cost crisis is real and growing. Research reveals that 70% of all logs ingested are never actually queried, meaning organizations are paying full price for data that provides zero value. This problem is compounded by microservices architecture, where each service generates its own log streams.

Optimize your resources



Why Costs Are Spiraling Out of Control

Microservices Explosion

Each service generates independent log streams, creating exponential volume growth

Cloud Vendor Pricing

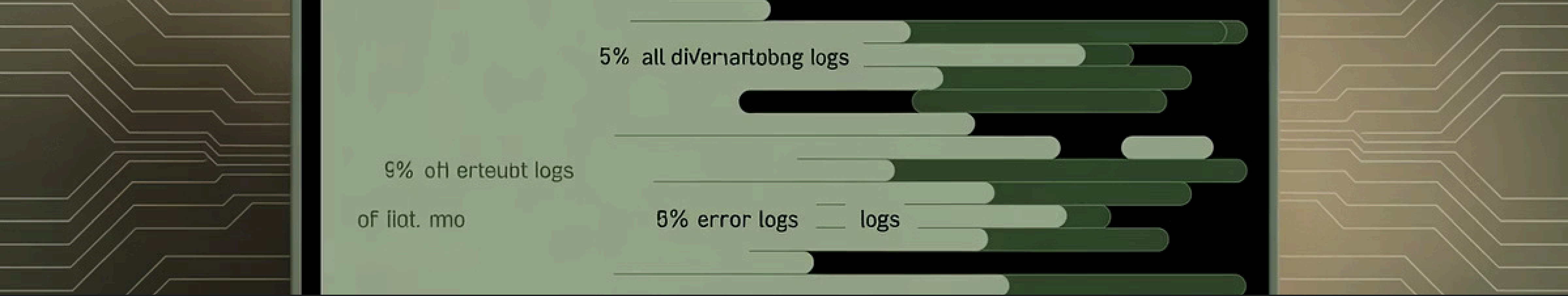
Per GB ingested plus retention costs create compound financial impact

Development Practices

Debug logs left enabled in production environments generate unnecessary data

Lack of Governance

No clear policies on what to log leads to inconsistent and excessive logging



Strategy 1 - Intelligent Sampling

Uber's Breakthrough

Sampled debug logs at 5%, kept 100% of errors

Annual savings: $8M (SRECon 2022)

Maintained full error visibility while dramatically reducing noise

Implementation Approaches

Head-based sampling: Decision made at log generation

Tail-based sampling: Decision made after trace completion

Level-based sampling: Different rates for different log levels

Best Practices

Always preserve ERROR and FATAL level logs

Sample at 1-10% for DEBUG and INFO levels

Use consistent sampling across service boundaries

Strategy 2 - Noise Filtering

Slack's Success Story

Problem: /healthcheck endpoints generated 2TB/month of useless logs

Solution: Dropped health checks with Fluent Bit filters

Result: 15% cost reduction with zero impact on debugging capabilities

Common Noise Sources

- Health check endpoints (/ping, /health, /status)
- Repeated connection timeouts (if not critical)
- Successful authentication logs
- Internal service-to-service communication

Filtering represents the most straightforward approach to cost reduction—simply don't log what you don't need. The key is identifying systematic noise sources that can be filtered out at the ingestion layer.

Strategy 3 - Tiered Storage Architecture

Hot Storage (0-7 days)

Elasticsearch for fast querying - $$$

Warm Storage (7-30 days)

S3 Standard for balanced cost/performance - $$

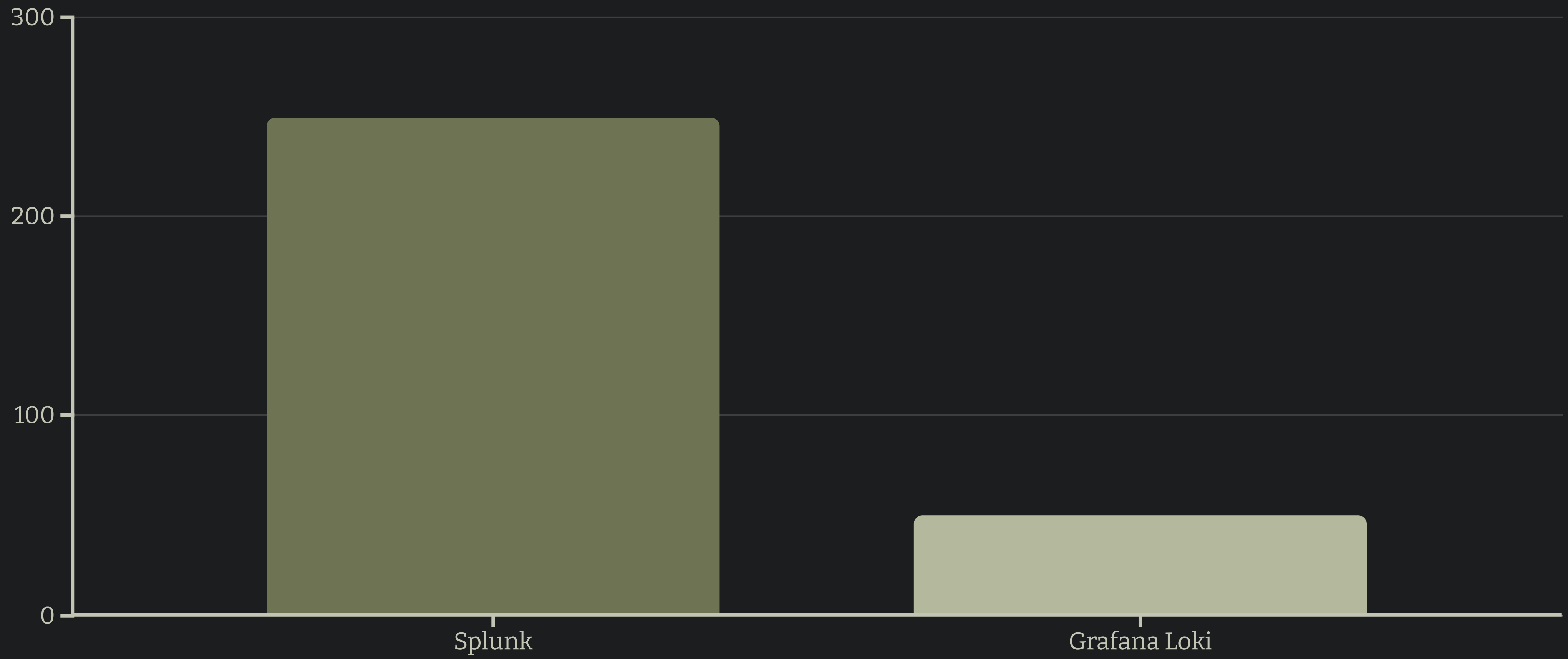
Cold Storage (30+ days)

S3 Glacier for archival needs - $

Shopify implemented this tiered approach and achieved 60% savings on retention costs. This strategy recognizes that log access patterns change dramatically over time, with the vast majority of queries focusing on recent data.

For compliance or deep investigation needs, older logs remain accessible but at much lower storage costs.

Strategy 4 - Open Source Migration



Razorpay's dramatic transformation shows the potential of open source solutions. Their switch from Splunk to Grafana Loki resulted in an 80% cost reduction, dropping from $250,000 to just $50,000 monthly.

Loki's architecture is particularly cost-effective because it doesn't index log content like traditional solutions, instead storing logs directly in object storage like S3. However, this strategy requires the highest implementation effort and ongoing operational overhead.

Real-World Case Study - Uber's CLP Revolution

169:1 Compression Ratio

Unprecedented efficiency

Search Without Decompression

Maintained full capabilities

$10K vs $1.8M Annual Cost

99% cost reduction

Uber faced a challenge with 250,000 Spark jobs daily generating 200TB of logs. Users demanded retention increase from 3 days to 1 month, which would have cost $1.8M annually in traditional storage.

Their Compressed Log Processor solution achieved a remarkable 169:1 compression ratio while maintaining search capabilities without decompression. This enabled them to extend retention periods by 10x and unlock new analytics capabilities.

Structured Logging Best Practices

Unstructured Logging

This is an example of unstructured logging:

2023-04-05 12:34:56 ERROR User signup failed: email already exists

This log entry provides some information, but it's difficult to parse and analyze at scale. There's no consistent structure or metadata.

Structured Logging

In contrast, here's an example of structured logging in JSON format:

```
{
  "timestamp": "2023-04-05T12:34:56.789Z",
  "level": "error",
  "message": "User signup failed",
  "details": {
    "email": "user@example.com",
    "error": "email already exists"
  }
}
```

The structured log entry includes consistent metadata like timestamp, log level, and relevant details. This makes it much easier to filter, analyze, and automate logging processes.

Enables Precise Filtering

Target specific log types for optimization

Improves Compression

Reduces storage requirements significantly

Enhances Query Performance

Find relevant information faster

Facilitates Automation

Enables automated analysis and alerting

Cost Optimization Success Metrics

Financial Metrics

- Monthly log ingestion costs
- Storage costs by tier
- Total cost per GB ingested

Operational Metrics

- Query response time
- Log availability
- Error detection effectiveness

Volume Metrics

- Daily log volume trends
- Percentage filtered vs. retained
- Storage utilization across tiers

Success in logging optimization requires comprehensive measurement across financial, operational, and volume metrics. The key is establishing baselines before implementing changes, then tracking improvements over time.

Common Pitfalls and How to Avoid Them

Over-Aggressive Filtering

Risk: Losing critical debugging information

Solution: Start conservatively, increase filtering gradually

Safeguard: Always preserve error-level logs

Sampling Bias

Risk: Missing important but infrequent events

Solution: Use trace-aware sampling strategies

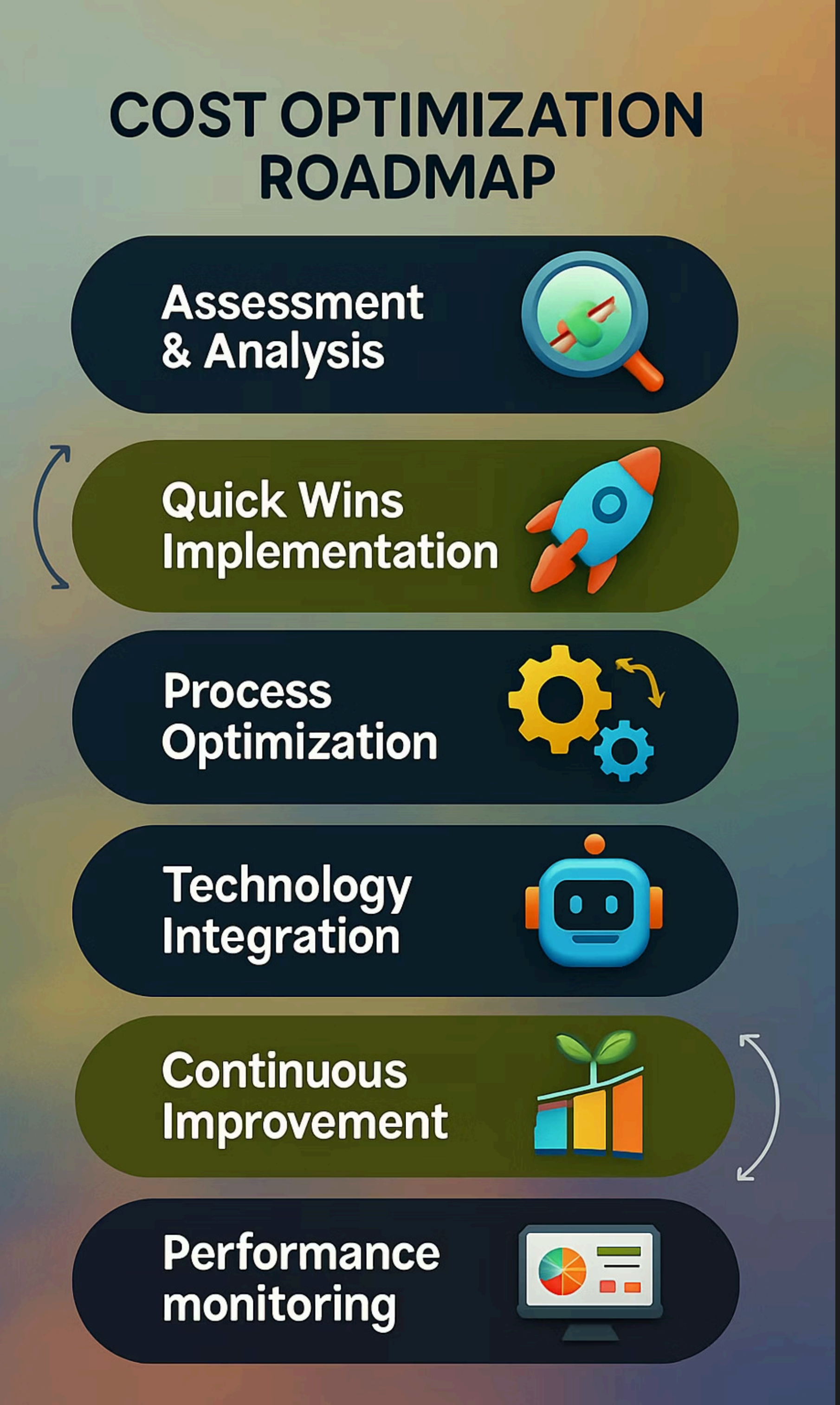
Safeguard: Implement alert-triggered full logging

Cold Storage Query Performance

Risk: Unacceptable query times for archived data

Solution: Design queries to minimize cold access

Safeguard: Implement log rehydration workflows



Action Plan and Call to Action

Immediate Actions (This Week)

- Run log volume audit using existing tools
- Identify top 5 noisiest log sources
- Calculate current monthly logging spend
- Set up log volume monitoring dashboards

30-Day Sprint

- Implement health check filtering
- Deploy debug log sampling (start at 10%)
- Design tiered storage architecture
- Establish structured logging standards

Strategic Initiatives (3-6 months)

- Evaluate open source alternatives
- Implement advanced compression techniques
- Develop log-based metrics for alerting
- Create cost optimization culture and training

Resources

Essential Tools

- Datadog Cost Management
- Fluent Bit, Logstash, Vector
- Grafana Loki, SigNoz, ELK Stack
- CLP (open source)

Further Learning

- Uber's CLP technical papers
- Datadog cost optimization docs
- Cloud provider best practices
- SRECon and LISA conference talks

Community Resources

- OpenTelemetry specifications
- CNCF logging projects
- Vendor user communities
- Internal optimization working groups