SRE Maturity Assessment Guide

Instructions for Interactive and Offline Assessment

Bot Army Engineering | Assessment Toolkit v1.0

15	75	450	30
DOMAINS	QUESTIONS	MAX POINTS	MINUTES

5 MATURITY LEVELS

LEVEL	NAME	SCORE	AVG/DOMAIN
1	**Ad-hoc**	0-90	0-6
2	**Foundational**	91-180	7-12
3	**Standardized**	181-270	13-18
4	**Advanced**	271-360	19-24
5	**Optimized**	361-450	25-30

INTERACTIVE ASSESSMENT

1. → **Prepare:** Gather 2-4 team members, block 30-45 min
2. → **Start:** Open Interactive Assessment, enter team name
3. → **Answer:** Score each question 0-6 honestly
4. → **Review:** Check radar chart and domain breakdown
5. → **Export:** Save JSON for historical tracking

OFFLINE PDF ASSESSMENT

1. → **Print:** Scoring worksheet + domain rubrics
2. → **Gather:** Same team prep as interactive
3. → **Score:** Use rubrics, record on worksheet
4. → **Calculate:** Sum domain totals (max 450)
5. → **Identify:** Circle domains below 13 points

SCORING GUIDE

SCORE	MEANING
0	Not practiced at all
2	Minimal/ad-hoc practice
3	Partial implementation
5	Strong implementation
6	Exemplary/industry-leading

AFTER THE ASSESSMENT

1. → **Share:** Results with team and stakeholders
2. → **Identify:** Top 3 gaps (lowest scoring domains)
3. → **Review:** Rubrics for those domains
4. → **Read:** Improvement playbooks for guidance
5. → **Plan:** Create action items with owners

ASSESSMENT CADENCE

ACTIVITY	FREQUENCY
Full assessment	Quarterly
Progress review	Monthly
Action tracking	Weekly
Stakeholder report	Quarterly

COMMON MISTAKES

→ Scoring aspirations instead of reality
→ Ignoring evidence for high scores
→ Rushing without team discussion
→ Skipping domains that "don't apply"

BOT ARMY OWNERSHIP

TEAM	DOMAINS
SRE Bot	1, 6, 7, 8, 9, 11, 15
Ops Bot	4, 5, 12
Observability	2, 3
Security Bot	13
All Teams	10, 14

Measure to Improve

Target Level 3+ for all critical services.

SRE Maturity Assessment Questions

Complete Question Reference for Offline Assessment

0 Not Practiced | 2 Basic/Reactive | 3 Defined | 5 Measured | 6 Optimized

01 SLOs & Error Budgets

Q1. How well-defined are your Service Level Indicators (SLIs)?
0No SLIs defined | 2Informal metrics tracked ad-hoc | 3SLIs defined for some services | 5Comprehensive SLIs for all critical services | 6User-journey based SLIs with clear measurement methodology

Q2. How do you track and enforce error budgets?
0No error budget concept | 2Error budgets calculated but not enforced | 3Error budgets tracked with manual reviews | 5Automated burn rate alerts with policy enforcement | 6Multi-window burn rates with automated feature freezes

Q3. How aligned are stakeholders on SLO targets?
0No stakeholder awareness of SLOs | 2Engineering aware, business not involved | 3SLOs documented and shared with stakeholders | 5Business and engineering co-own SLO targets | 6SLOs embedded in contracts and product decisions

Q4. What happens when error budget is exhausted?
0Nothing, we don't track error budgets | 2Manual discussions, no formal process | 3Documented escalation process | 5Automatic feature freeze, reliability focus | 6Proactive budget management prevents exhaustion

Q5. How do you review and iterate on SLOs?
0SLOs never reviewed | 2Reviewed when issues occur | 3Quarterly reviews scheduled | 5Regular reviews with customer feedback integration | 6Continuous refinement based on user journey analysis

02 Observability

Q1. How comprehensive is your metrics coverage?
0Ad-hoc metrics; no consistent approach across services | 2System metrics (CPU/mem/disk) covered; app metrics ad-hoc | 3RED/USE methods adopted for critical services | 5Comprehensive coverage; consistent standards across services | 6All services with golden signals; consistent naming/labels

Q2. How mature is your logging infrastructure?
0Logs only on local disk, grep to debug | 2Some centralized logging, unstructured | 3Centralized structured logging with search | 5Structured logs with correlation IDs and retention policies | 6Intelligent log analysis with anomaly detection

Q3. How well do you implement distributed tracing?
0No distributed tracing | 2Tracing in some services, not correlated | 3End-to-end tracing for critical paths | 5Full tracing with service maps and latency analysis | 6Continuous profiling integrated with tracing

Q4. How effective are your dashboards?
0No dashboards or ad-hoc only | 2Basic dashboards, often outdated | 3Service-level dashboards maintained | 5Golden signals dashboard per service with SLO tracking | 6Self-service dashboard platform with templates

Q5. Can you correlate signals across metrics, logs, and traces?
0Signals completely siloed | 2Manual correlation via timestamps | 3Some tooling for correlation | 5Unified observability platform with correlation | 6AI-assisted root cause analysis across signals

03 Alerting Strategy

Q1. What percentage of your alerts are actionable?
0Unknown or mostly noise | 2Less than 50% actionable | 350-80% actionable | 580-95% actionable, regular tuning | 6>95% actionable, continuous improvement

Q2. How are alerts linked to runbooks?
0No runbooks exist | 2Some runbooks, not linked from alerts | 3Runbooks linked for critical alerts | 5All alerts link to runbooks, regularly updated | 6Runbooks with automation hooks and versioning

Q3. How do you tune alert thresholds?
0Set once, never tuned | 2Tuned reactively after complaints | 3Quarterly review of noisy alerts | 5Data-driven tuning with noise metrics | 6Automated threshold adjustment based on patterns

Q4. Do alerts correlate with SLO burn rates?
0No SLO-based alerting | 2Basic threshold alerts only | 3Single-window burn rate alerts | 5Multi-window burn rate alerts (fast + slow) | 6Predictive alerting before budget exhaustion

Q5. How do you manage alert escalation?
0No escalation process | 2Informal escalation via chat/phone | 3Documented escalation paths | 5Automated escalation with on-call integration | 6Intelligent routing based on context and expertise

04 Incident Response

Q1. How well-defined is your Incident Commander (IC) role?
0No IC role, whoever is available | 2Informal IC, not always assigned | 3IC role defined, rotation exists | 5Trained ICs, clear handoff procedures | 6IC certification program, regular drills

Q2. How do you track MTTD/MTTR?
0Not tracked | 2Ad-hoc calculations (spreadsheets, after-the-fact) | 3Incident system with manual timestamp entry | 5Automated capture from alerts and monitoring | 6Real-time dashboards with trend analysis

Q3. How do you conduct postmortems?
0No postmortems | 2Ad-hoc reviews for major incidents | 3Blameless postmortems with template | 5Postmortems with tracked action items | 6Learning reviews shared org-wide, patterns analyzed

Q4. How effective are your escalation paths?
0No defined escalation | 2Escalation exists but often unclear | 3Documented escalation matrix | 5Tested escalation with clear SLAs | 6Automated escalation with fallback procedures

Q5. How do you train incident responders?
0No training, learn by doing | 2Informal shadowing | 3Onboarding training exists | 5Regular game days and tabletop exercises | 6Certification program with continuous learning

05 On-Call Health

Q1. What percentage of time is spent on on-call work?
0>50% of time on reactive work | 235-50% reactive | 325-35% reactive | 5<25% reactive, rest proactive | 6<15% reactive, highly automated

Q2. How many pages require response per on-call shift?
0>10 pages per shift | 25-10 pages per shift | 32-5 pages per shift | 5<2 pages per shift | 6<1 page per shift; mostly proactive

Q3. How is on-call duty recognized and compensated?
0No policy; on-call expected without recognition | 2Informal; varies by manager or team | 3Documented policy with time-off or pay | 5Clear policy with pay, time-off, and flexibility | 6Competitive compensation; on-call valued

Q4. How do you track on-call health metrics?
0Not tracked | 2Anecdotal feedback only | 3Basic metrics (pages, hours) | 5Comprehensive dashboard with trends | 6Health metrics tied to improvement goals

Q5. How do you prevent burnout?
0Burnout is common, no prevention | 2React when people complain | 3Rotation policies, some flexibility | 5Proactive monitoring, load balancing | 6Sustainable by design, team satisfaction high

06 Reliability Patterns

Q1. How do you implement circuit breakers?
0No circuit breakers | 2Ad-hoc implementation in some services | 3Standard library used for critical paths | 5All external calls protected, monitored | 6Adaptive circuit breakers with auto-tuning

Q2. How standardized are timeouts and retries?
0No timeouts, or infinite waits | 2Inconsistent timeouts across services | 3Standard timeout policy documented | 5Exponential backoff with jitter everywhere | 6Context-aware adaptive timeouts

Q3. Do you isolate resources to prevent cascading failures?
0No resource isolation; shared everything | 2Some isolation, not systematic | 3Critical services have dedicated resources | 5Systematic isolation per dependency | 6Dynamic isolation that adjusts to load

Q4. How do you handle graceful degradation?
0All-or-nothing failures | 2Some fallbacks, not systematic | 3Degradation modes documented | 5Automatic degradation with user communication | 6Feature flags enable instant degradation

Q5. How do you prevent cascading failures?
0Cascading failures happen regularly | 2Some awareness, reactive fixes | 3Load shedding for critical services | 5Comprehensive protection at all layers | 6Automatic blast radius containment

07 Capacity & Performance

Q1. How do you monitor utilization and saturation?
0Not monitored | 2Basic CPU/memory only | 3USE method for critical resources | 5Comprehensive USE dashboards with alerts | 6Predictive capacity analysis

Q2. How mature is your autoscaling?
0No autoscaling, manual only | 2Basic CPU-based autoscaling | 3Custom metrics-based autoscaling | 5Predictive scaling with business signals | 6ML-driven proactive scaling

Q3. How often do you load test?
0Never or rarely | 2Ad-hoc; only when issues arise | 3Planned (scheduled or before releases) | 5Automated in CI/CD | 6Continuous with trend analysis

Q4. Do you have capacity models?
0No capacity planning | 2Gut feel; no documented approach | 3Documented models for critical services | 5Data-driven models updated regularly | 6Automated capacity forecasting

Q5. How do you forecast demand?
0No forecasting | 2Ad-hoc estimates | 3Historical trend analysis | 5Integrated with business planning | 6ML-based demand prediction

08 Release Engineering

Q1. How mature is your CI/CD pipeline?
0Manual builds and deployments | 2Basic CI, manual CD | 3Full CI/CD for most services | 5Standardized pipelines with quality gates | 6Self-service platform with guardrails

Q2. How do you implement canary releases?
0Big bang releases only | 2Gradual rollout; ad-hoc monitoring | 3Canary/baseline comparison; manual promotion | 5Automated promotion based on SLOs | 6Progressive delivery with auto-rollback

Q3. What are your DORA metrics?
0Not tracked | 2Low performer (monthly deploys, >6mo lead time) | 3Medium (weekly deploys, 1-6mo lead time) | 5High (daily deploys, <1 week lead time) | 6Elite (on-demand deploys, <1 day lead time)

Q4. How fast can you rollback?
0Rollback not possible or hours | 230-60 minutes | 310-30 minutes | 5<5 minutes, one-click | 6Automatic rollback on SLO breach

Q5. How do you use feature flags?
0No feature flags | 2Ad-hoc flags in code | 3Feature flag system for new features | 5Comprehensive flag management with targeting | 6Flags integrated with experiments and metrics

09 Toil & Automation

Q1. What percentage of time is spent on toil?
0>50% toil | 235-50% toil | 325-35% toil | 5<25% toil | 6<10% toil, mostly engineering

Q2. How mature is your Infrastructure as Code?
0ClickOps, manual provisioning | 2Some IaC, not comprehensive | 3IaC for new infrastructure | 5Full IaC with GitOps workflow | 6IaC with policy as code and drift detection

Q3. How much can developers do without waiting on ops?
0Tickets required for everything | 2Some self-service; most needs require tickets | 3Common daily tasks are self-service | 5Developer portal with golden paths | 6Full platform with self-healing

Q4. How do you track and prioritize toil reduction?
0Toil not tracked | 2Anecdotal awareness | 3Toil tracked, backlog exists | 5Dedicated time for automation (20%) | 6Toil elimination is a team OKR

Q5. How automated are routine operations?
0Mostly manual | 2Scripts exist, not maintained | 3Key operations automated | 5Comprehensive automation platform | 6AI-assisted autonomous operations

10 Culture & Organization

Q1. How blameless are your postmortems?
0Blame culture, people punished | 2Lip service to blameless | 3Genuinely blameless most times | 5Blameless culture, focus on systems | 6Failures celebrated as learning opportunities

Q2. Is psychological safety present?
0Fear of speaking up | 2Varies by team/manager | 3Generally safe to raise concerns | 5High safety, concerns welcomed | 6Proactive seeking of diverse perspectives

Q3. How well do teams collaborate?
0Siloed, competitive | 2Collaboration when forced | 3Regular cross-team interaction | 5Embedded SREs, shared ownership | 6Generative culture (Westrum)

Q4. How is knowledge shared?
0Tribal knowledge, silos | 2Documentation exists, outdated | 3Regular knowledge sharing sessions | 5Learning culture, communities of practice | 6Organization-wide learning system

Q5. How is reliability ownership distributed?
0Ops/SRE owns all reliability | 2Developers aware but not responsible | 3Shared on-call between dev and SRE | 5You build it, you run it | 6Reliability embedded in all teams

11 Chaos Engineering

Q1. How often do you run automated fault injection?
0Never | 2Ad-hoc; only after major incidents | 3Planned experiments in staging | 5Regular in staging; periodic in production | 6Continuous automated chaos in production

Q2. How do you control blast radius?
0No controls, hope for the best | 2Manual safeguards | 3Documented blast radius limits | 5Automated abort conditions | 6Intelligent blast radius with auto-scaling

Q3. Do you run game days or wargaming exercises?
0No organizational exercises | 2Informal tabletop discussions | 3Annual scheduled exercises | 5Quarterly with cross-team scenarios | 6Regular full incident simulations

Q4. How do you apply learnings from chaos?
0Findings ignored | 2Ad-hoc follow-up | 3Findings tracked, some fixed | 5All findings tracked with SLAs | 6Continuous improvement from chaos insights

Q5. What chaos tooling do you use?
0No tooling | 2Manual scripts | 3Basic chaos tools (kill pods, etc.) | 5Comprehensive platform (Gremlin, LitmusChaos) | 6Custom platform integrated with observability

12 Disaster Recovery

Q1. Are your disaster recovery targets defined and validated?
0No defined recovery targets | 2Targets defined; never tested | 3Targets defined; tested occasionally | 5Regularly validated; meets targets | 6Continuously validated; exceeds targets

Q2. How do you verify backups are restorable?
0Never tested | 2Only tested when issues occur | 3Occasional restore tests | 5Regular tests with data verification | 6Automated continuous validation

Q3. How do you test failover?
0Failover never tested | 2Tested once, years ago | 3Annual DR drills | 5Quarterly failover tests | 6Regular active-active failover

Q4. Do you have multi-region capability?
0Single region only | 2Cold standby in another region | 3Warm standby with manual failover | 5Hot standby with automated failover | 6Active-active multi-region

Q5. How automated is recovery?
0Fully manual, tribal knowledge | 2Documented runbooks | 3Partially automated | 5Mostly automated with one-click recovery | 6Self-healing with automatic recovery

13 Security Reliability

Q1. How do you manage secrets?
0Secrets in source code or env vars | 2Basic secrets storage, manual rotation | 3Secrets vault with access control | 5Dynamic secrets with auto-rotation | 6Zero-trust secrets with audit logging

Q2. How are certificates managed?
0Manual renewal, outages from expiry | 2Calendar reminders for renewal | 3Automated monitoring of expiry | 5Auto-renewal (cert-manager, ACME) | 6Short-lived certs with continuous rotation

Q3. How do you scan for vulnerabilities?
0No scanning | 2Ad-hoc scans | 3Scheduled scans, manual remediation | 5CI/CD integrated scanning with blocking | 6Continuous scanning with auto-remediation

Q4. How often do you rotate credentials?
0Never or when compromised | 2Annually | 3Quarterly | 5Monthly or on-demand | 6Continuous rotation (short-lived)

Q5. How do you handle security incidents?
0No security incident process | 2Ad-hoc response | 3Security incident runbooks exist | 5Dedicated security incident response team | 6Automated detection and response (SOAR)

14 Documentation

Q1. How current is your architecture documentation?
0No architecture docs | 2Outdated diagrams | 3Docs exist, updated occasionally | 5Current docs, reviewed quarterly | 6Auto-generated from code/infra

Q2. Do runbooks exist for all alerts?
0No runbooks | 2Runbooks for some alerts | 3Runbooks for critical alerts | 5All alerts have runbooks | 6Executable runbooks with automation

Q3. How do you track architecture decisions?
0No record of decisions | 2Decisions in chat/email | 3Some ADRs written | 5ADR process followed consistently | 6ADRs linked to code and searchable

Q4. How do you keep docs up-to-date?
0Docs abandoned after creation | 2Updated when someone notices issues | 3Review cadence exists | 5Docs as code in PRs | 6Automated freshness checks

Q5. Can new team members onboard via docs?
0Heavy reliance on shadowing | 2Some docs, mostly tribal knowledge | 3Onboarding guide exists | 5Self-service onboarding possible | 6Comprehensive onboarding with exercises

15 Dependency Management

Q1. Do you have a complete service map?
0No service map | 2Partial, outdated map | 3Manual service map maintained | 5Auto-discovered service map | 6Real-time dependency graph with health

Q2. How do you track vendor SLAs?
0Vendor SLAs unknown | 2SLAs known but not monitored | 3Major vendors monitored | 5All vendors tracked with dashboards | 6SLA compliance automated with alerts

Q3. How do you monitor dependency health?
0No dependency monitoring | 2Manual checks or vendor status pages | 3Health endpoints monitored | 5Comprehensive dependency dashboard | 6Predictive dependency health analysis

Q4. What's your strategy for vendor outages?
0No strategy, wait for vendor | 2Manual workarounds | 3Documented fallback procedures | 5Automated failover to alternatives | 6Multi-vendor redundancy by default

Q5. How do you manage library dependencies?
0Dependencies not tracked | 2Manual review occasionally | 3Automated vulnerability scanning | 5Automated updates with testing | 6Dependency governance with policies

SRE Maturity Scoring Worksheet

Offline Assessment Form - Print and Complete

Bot Army Engineering | Assessment Toolkit

Team: _____ **Date:** _____ **Assessors:** _____

SCORING GRID (5 QUESTIONS PER DOMAIN, 0-6 POINTS EACH)

#	DOMAIN	Q1	Q2	Q3	Q4	Q5	TOTAL	LEVEL	NOTES
1	SLOs & Error Budgets						/30		
2	Observability						/30		
3	Alerting Strategy						/30		
4	Incident Response						/30		
5	On-Call Health						/30		
6	Reliability Patterns						/30		
7	Capacity & Performance						/30		
8	Release Engineering						/30		
9	Toil & Automation						/30		
10	Culture & Organization						/30		
11	Chaos Engineering						/30		
12	Disaster Recovery						/30		
13	Security Reliability						/30		
14	Documentation						/30		
15	Dependency Management						/30		
						TOTAL SCORE:	/450	Maturity Level:	

MATURITY LEVELS

LEVEL	NAME	SCORE
1	Ad-hoc	0-90
2	Foundational	91-180
3	Standardized	181-270
4	Advanced	271-360
5	Optimized	361-450

TOP 3 PRIORITY GAPS

1. → Domain: _____ Score: ___ Action: _____
2. → Domain: _____ Score: ___ Action: _____
3. → Domain: _____ Score: ___ Action: _____

QUESTION SCORING

SCORE	MEANING
0	Not practiced
2	Minimal/ad-hoc
3	Partial implementation
5	Strong implementation
6	Exemplary

NEXT STEPS

- → Review rubrics for low-scoring domains
- → Read improvement playbooks
- → Create action items with owners
- → Schedule quarterly reassessment

Domain 1: SLOs & Error Budgets

Service Level Objectives and Error Budget Management

SRE Bot | Foundations | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	No formal SLOs; availability discussed informally; no error budgets
2	Basic SLOs for some services; not consistently tracked; no budget enforcement
3	SLOs for critical services; error budgets calculated; basic burn rate monitoring
4	Comprehensive SLOs; budgets enforced; dev slowdowns when budget exhausted
5	SLOs drive all decisions; multi-window burn rates; automated freezes

ANTI-PATTERNS (RED FLAGS)

- → Setting 100% availability targets (impossible, expensive)
- → SLOs without error budget policies
- → Engineering-only SLOs, no business alignment
- → No consequence for budget violations
- → Static SLOs that never evolve

EVIDENCE CHECKLIST

- → SLO documentation exists and is up-to-date
- → Error budget dashboards visible to stakeholders
- → Historical SLO compliance data available
- → Error budget policy with escalation process
- → Evidence of SLO-driven prioritization decisions

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How well-defined are your SLIs?	6
2	How do you track/enforce error budgets?	6
3	How aligned are stakeholders on SLO targets?	6
4	What happens when error budget exhausted?	6
5	How do you review and iterate on SLOs?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Observability	SLIs require metrics/logs infrastructure
Alerting	Burn rate alerts drive incident response
Release Eng	Error budgets gate feature releases

FOCUS AREAS

- → **SLI Definition:** User-journey based indicators with clear measurement
- → **SLO Targets:** Realistic, stakeholder-aligned availability goals
- → **Error Budget Policy:** Clear consequences for budget violations
- → **Stakeholder Alignment:** Business and engineering co-ownership

Error Budgets Enable Velocity

Managed risk, not zero risk.

Domain 2: Observability

Metrics, Logs, Traces, and Dashboards

Observability Bot | Observability | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Minimal logging; no centralized metrics; debugging via SSH
2	Basic metrics/logs; some dashboards; siloed per team
3	Centralized observability stack; standard dashboards; basic tracing
4	Full pillars (metrics, logs, traces); correlation; self-service
5	Exemplars, continuous profiling; AI-assisted analysis

ANTI-PATTERNS (RED FLAGS)

- → Debugging production via SSH
- → Metrics without context (no labels/tags)
- → Logs without structured fields
- → Dashboard sprawl with no ownership
- → Observability as afterthought

EVIDENCE CHECKLIST

- → Centralized metrics platform (Prometheus, Datadog, etc.)
- → Log aggregation with search capability
- → Tracing enabled for critical paths
- → Service-level dashboards exist
- → Runbooks link to relevant dashboards

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How comprehensive is your metrics coverage?	6
2	How mature is your logging infrastructure?	6
3	How well do you implement distributed tracing?	6
4	How effective are your dashboards?	6
5	Can you correlate across signals?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
SLOs	SLIs derive from observability data
Alerting	Alerts query observability backend
Incidents	Dashboards critical for diagnosis

FOCUS AREAS

- → **Metrics:** RED/USE methods, cardinality management
- → **Logs:** Structured logging, centralized aggregation
- → **Traces:** Distributed tracing, context propagation
- → **Dashboards:** Service-oriented, actionable visualizations

Observe, Don't Guess

Data-driven debugging at scale.

Domain 3: Alerting Strategy

Actionable Alerts and Runbooks

Observability Bot | Observability | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Few alerts; mostly noisy; no runbooks; alert fatigue common
2	Basic alerts exist; high noise ratio; some documentation
3	SLO-based alerts; runbooks linked; regular tuning
4	Multi-window burn rates; <5% noise; automated tuning
5	Self-healing alerts; ML anomaly detection; proactive

ANTI-PATTERNS (RED FLAGS)

- → Alerting on causes, not symptoms
- → >20% non-actionable alerts
- → No runbooks or outdated runbooks
- → Alert storms during incidents
- → Alerts ignored due to fatigue

EVIDENCE CHECKLIST

- → Alert actionability metrics tracked
- → Runbooks exist for all critical alerts
- → Alert noise ratio <20%
- → Multi-window burn rate alerts configured
- → Regular alert review cadence

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	What % of alerts are actionable?	6
2	How are alerts linked to runbooks?	6
3	How do you tune alert thresholds?	6
4	Do alerts correlate with SLO burn rates?	6
5	How do you manage alert escalation?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
SLOs	Burn rate alerts derive from SLOs
Observability	Alerts query observability data
On-Call	Alert quality affects on-call health

FOCUS AREAS

- → **Actionability:** Every alert should have a clear action
- → **SLO-Based:** Alert on error budget burn, not thresholds
- → **Runbooks:** Documented response procedures
- → **Tuning:** Regular noise reduction reviews

Alert on Symptoms

Every page should require human action.

Domain 4: Incident Response

Incident Command, Escalation, and Resolution

Ops Bot | Operations | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Chaotic response; no IC role; hero culture; no learning
2	Basic severity levels; some escalation paths; informal IC
3	Defined IC role; runbooks used; postmortems written
4	Trained ICs; MTTD/MTTR tracked; blameless culture
5	Incident learning system; automated mitigation; chaos drills

ANTI-PATTERNS (RED FLAGS)

- → Hero culture (same person always responds)
- → Blame-focused incident reviews
- → No severity classification
- → Postmortem actions never completed
- → Escalation unclear or broken

EVIDENCE CHECKLIST

- → IC rotation schedule exists
- → Severity levels defined with examples
- → Escalation matrix documented
- → Postmortem template in use
- → MTTD/MTTR dashboards available

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How well-defined is your IC role?	6
2	How do you track MTTD/MTTR?	6
3	How do you conduct postmortems?	6
4	How effective are escalation paths?	6
5	How do you train incident responders?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
On-Call	On-call handles initial response
Alerting	Alerts trigger incident flow
Culture	Blameless culture enables learning

FOCUS AREAS

- → **IC Role:** Clear ownership during incidents
- → **Escalation:** Defined paths with contact info
- → **Metrics:** MTTD, MTTR, incident frequency
- → **Learning:** Blameless postmortems with actions

Incidents Are Learning Events

Every outage makes us stronger.

Domain 5: On-Call Health

Sustainable On-Call and Burnout Prevention

Ops Bot | Operations | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Burnout common; no rotation; >50% time reactive
2	Basic rotation; frequent paging; compensation unclear
3	Regular rotation; <25% time on-call; comp policy exists
4	<2 incidents/shift; health tracked; follow-the-sun
5	Proactive on-call; minimal paging; team satisfaction high

ANTI-PATTERNS (RED FLAGS)

- → Same people always on-call
- → >5 incidents per shift average
- → No compensation for pages
- → High turnover due to burnout
- → On-call seen as punishment

EVIDENCE CHECKLIST

- → On-call rotation schedule published
- → Incidents per shift tracked (<2 target)
- → Compensation policy documented
- → Team satisfaction surveys conducted
- → Handoff procedures documented

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	What % of time is spent on-call work?	6
2	How many incidents per on-call shift?	6
3	Is on-call compensation clear?	6
4	How do you track on-call health metrics?	6
5	How do you prevent burnout?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Alerting	Alert quality affects paging load
Incidents	Incident volume drives on-call stress
Culture	Healthy culture supports on-call

FOCUS AREAS

- → **Rotation:** Fair distribution, follow-the-sun if global
- → **Workload:** <25% time, <2 incidents/shift target
- → **Compensation:** Clear policy, time-off for pages
- → **Health:** Burnout tracking, satisfaction surveys

Sustainable Operations

<25% time, <2 incidents/shift.

Domain 6: Reliability Patterns

Circuit Breakers, Retries, Timeouts, Bulkheads

SRE Bot | Resilience | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	No defensive patterns; cascading failures common
2	Basic timeouts in some services; retry logic ad-hoc
3	Circuit breakers for critical paths; standardized timeouts
4	Bulkheads, load shedding; graceful degradation
5	Adaptive patterns; self-healing; antifragile design

ANTI-PATTERNS (RED FLAGS)

- → No timeouts (infinite waits)
- → Retry storms (no backoff)
- → All-or-nothing failures
- → Cascading failures across services
- → No graceful degradation paths

EVIDENCE CHECKLIST

- → Circuit breaker library in use (Hystrix, resilience4j)
- → Timeout policy documented
- → Retry strategy with backoff implemented
- → Load shedding mechanisms exist
- → Graceful degradation tested

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How do you implement circuit breakers?	6
2	How standardized are timeouts/retries?	6
3	Do you use bulkhead isolation?	6
4	How do you handle graceful degradation?	6
5	How do you prevent cascading failures?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Chaos Eng	Test patterns via chaos experiments
Dependencies	Patterns protect from dep failures
Capacity	Load shedding prevents overload

FOCUS AREAS

- → **Circuit Breakers:** Fail fast when dependencies unhealthy
- → **Timeouts:** Bounded wait times for all calls
- → **Retries:** Exponential backoff with jitter
- → **Bulkheads:** Isolate failure domains

Design for Failure

Assume everything will fail.

Domain 7: Capacity & Performance

USE Method, Load Testing, Autoscaling

SRE Bot | Resilience | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	No capacity planning; reactive scaling; no load testing
2	Basic monitoring; manual scaling; occasional load tests
3	USE method applied; autoscaling configured; regular tests
4	Capacity models; predictive scaling; continuous perf tests
5	ML-based forecasting; cost-optimized; real-time adaptation

ANTI-PATTERNS (RED FLAGS)

- → Scaling only when pages fire
- → No load testing before releases
- → Unknown system limits
- → Over-provisioned for "safety"
- → No performance budgets

EVIDENCE CHECKLIST

- → USE method dashboards for all services
- → Autoscaling policies configured
- → Load testing in CI/CD pipeline
- → Performance regression tests exist
- → Capacity planning documentation

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How do you monitor utilization/saturation?	6
2	How mature is your autoscaling?	6
3	How often do you load test?	6
4	Do you have capacity models?	6
5	How do you forecast demand?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Observability	USE metrics from observability
Reliability	Load shedding at capacity limits
Release Eng	Perf tests gate releases

FOCUS AREAS

- → **USE Method:** Utilization, Saturation, Errors
- → **Load Testing:** Regular stress tests in CI/CD
- → **Autoscaling:** Horizontal scaling with proper signals
- → **Forecasting:** Demand prediction for planning

Know Your Limits

Measure, model, scale proactively.

Domain 8: Release Engineering

CI/CD, Canary Deployments, DORA Metrics

SRE Bot | Release | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Manual deployments; release days are stressful; no rollback
2	Basic CI; some CD; deployments weekly/monthly
3	Full CI/CD; canary deployments; DORA metrics tracked
4	Elite DORA metrics; automated rollback; feature flags
5	Continuous deployment; zero-downtime; progressive delivery

ANTI-PATTERNS (RED FLAGS)

- → Manual deployments with scripts
- → Big bang releases
- → No rollback capability
- → Releases require downtime
- → DORA metrics unknown

EVIDENCE CHECKLIST

- → CI/CD pipeline fully automated
- → Canary or blue-green deployments
- → DORA metrics dashboard exists
- → Rollback tested and documented
- → Feature flag system in use

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How mature is your CI/CD pipeline?	6
2	How do you implement canary releases?	6
3	What are your DORA metrics?	6
4	How fast can you rollback?	6
5	How do you use feature flags?	6

DORA ELITE TARGETS

METRIC	ELITE TARGET
Deploy Frequency	Multiple/day
Lead Time	<1 hour
MTTR	<1 hour
Change Fail Rate	<15%

FOCUS AREAS

- → **DORA:** Frequency, lead time, MTTR, change fail rate
- → **Canary:** Progressive rollout with auto-rollback
- → **Feature Flags:** Decouple deploy from release
- → **Rollback:** <5 minute recovery capability

Deploy Boring

Releases should be non-events.

Domain 9: Toil & Automation

Toil Reduction, Self-Service, Infrastructure as Code

SRE Bot | Release | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	>50% toil; manual everything; ticket-driven ops
2	Some automation; toil not measured; ad-hoc scripts
3	Toil <50%; IaC for infra; some self-service
4	Toil <30%; full IaC; developer self-service
5	Toil minimal; platform engineering; autonomous ops

ANTI-PATTERNS (RED FLAGS)

- → Tickets for everything (ops as bottleneck)
- → ClickOps in production
- → Undocumented tribal knowledge
- → SRE team is ticket queue
- → No time allocated for automation

EVIDENCE CHECKLIST

- → Toil % tracked (<50% target)
- → Infrastructure managed via IaC
- → Self-service portal for common tasks
- → Automation backlog exists
- → Time explicitly allocated for automation

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	What % of time is spent on toil?	6
2	How mature is your IaC?	6
3	What self-service exists for developers?	6
4	How do you track/prioritize toil reduction?	6
5	How automated are routine operations?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
On-Call	Reduce pages via automation
Release Eng	CI/CD reduces deploy toil
Documentation	Automate runbook execution

FOCUS AREAS

- → **Toil:** Manual, repetitive, automatable work
- → **IaC:** Infrastructure defined in code (Terraform, Pulumi)
- → **Self-Service:** Developer portals, golden paths
- → **Automation:** Script → tool → platform progression

Automate the Boring

<50% toil, or push back.

Domain 10: Culture & Organization

Blameless Culture, Psychological Safety, Learning

All Teams | Culture | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Blame culture; heroes celebrated; silos; fear of speaking up
2	Some awareness; lip service to blameless; inconsistent
3	Blameless postmortems; cross-team collaboration; learning
4	Generative culture (Westrum); psychological safety; innovation
5	Learning organization; failure celebrated; continuous growth

ANTI-PATTERNS (RED FLAGS)

- → Looking for "who" not "what" failed
- → Punishing people for incidents
- → Information hoarding
- → Fear of asking questions
- → Reliability is "ops problem"

EVIDENCE CHECKLIST

- → Blameless postmortem template in use
- → Psychological safety surveys conducted
- → Cross-team collaboration examples
- → Knowledge sharing sessions regular
- → SRE embedded with dev teams

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How blameless are your postmortems?	6
2	Is psychological safety present?	6
3	How well do teams collaborate?	6
4	How is knowledge shared?	6
5	How is reliability ownership distributed?	6

WESTRUM CULTURE TYPES

TYPE	CHARACTERISTICS
Pathological	Blame, silos, fear
Bureaucratic	Rules, turf, tolerance
Generative	Learning, sharing, inquiry

FOCUS AREAS

- → **Blameless:** Focus on systems, not individuals
- → **Safety:** Safe to report errors, ask questions
- → **Westrum:** Generative vs pathological culture
- → **Learning:** Continuous improvement mindset

Blame Systems, Not People

Psychological safety enables excellence.

Domain 11: Chaos Engineering

Game Days, Blast Radius Control, Failure Injection

SRE Bot | Resilience | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	No chaos practice; only learn from real outages
2	Occasional game days; manual failure injection
3	Regular chaos experiments; blast radius controlled
4	Continuous chaos in staging; production game days
5	Chaos in production daily; antifragile systems

ANTI-PATTERNS (RED FLAGS)

- → Chaos without hypothesis
- → No blast radius controls
- → Chaos findings ignored
- → Only chaos in staging
- → Chaos as one-time event

EVIDENCE CHECKLIST

- → Chaos experiment runbooks exist
- → Game day schedule published
- → Blast radius controls documented
- → Chaos findings tracked and fixed
- → Production chaos (with controls)

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How often do you run chaos experiments?	6
2	How do you control blast radius?	6
3	Do you run game days?	6
4	How do you apply learnings from chaos?	6
5	What chaos tooling do you use?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Reliability	Validate patterns via chaos
DR	Test DR via chaos experiments
Incidents	Build muscle memory for response

FOCUS AREAS

- → **Experiments:** Hypothesis-driven failure injection
- → **Blast Radius:** Start small, expand gradually
- → **Game Days:** Scheduled team resilience exercises
- → **Tooling:** Chaos Monkey, Gremlin, Litmus

Break Things on Purpose

Find failures before they find you.

Domain 12: Disaster Recovery

RPO/RTO, Backup Testing, Failover Procedures

Ops Bot | Operations | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	No DR plan; backups untested; single region
2	Basic backups; DR plan exists but untested
3	RPO/RTO defined; backups tested; failover documented
4	Regular DR drills; automated failover; multi-region
5	Active-active; automated recovery; continuous DR testing

ANTI-PATTERNS (RED FLAGS)

- → Untested backups
- → Unknown RPO/RTO
- → Single point of failure
- → DR plan never tested
- → Manual recovery procedures

EVIDENCE CHECKLIST

- → RPO/RTO documented per service
- → Backup restoration tested quarterly
- → Failover runbooks exist
- → DR drills conducted annually
- → Multi-region deployment (if applicable)

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	Are RPO/RTO defined and met?	6
2	How often do you test backups?	6
3	How do you test failover?	6
4	Do you have multi-region capability?	6
5	How automated is recovery?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Chaos	Chaos tests DR capabilities
Security	Backup encryption, access
Incidents	DR invoked during major incidents

FOCUS AREAS

- → **RPO:** Recovery Point Objective (data loss)
- → **RTO:** Recovery Time Objective (downtime)
- → **Backups:** Regular testing, not just creation
- → **Failover:** Tested, documented procedures

Plan for Failure

Test your backups, test your failover.

Domain 13: Security Reliability

Secrets, Certificates, Vulnerability Scanning

Security Bot | Governance | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Secrets in code; manual cert management; no scanning
2	Basic secrets vault; some cert automation; ad-hoc scans
3	Secrets rotated; cert auto-renewal; regular scanning
4	Zero-trust principles; scanning in CI; short-lived creds
5	Dynamic secrets; continuous compliance; automated remediation

ANTI-PATTERNS (RED FLAGS)

- → Secrets in source control
- → Long-lived credentials
- → Manual certificate renewals
- → No vulnerability scanning
- → Security as afterthought

EVIDENCE CHECKLIST

- → Secrets vault in use (HashiCorp, AWS SM)
- → Certificates auto-renew (cert-manager)
- → Vulnerability scanning in CI/CD
- → Credential rotation policy documented
- → Security incident runbook exists

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How do you manage secrets?	6
2	How are certificates managed?	6
3	How do you scan for vulnerabilities?	6
4	How often do you rotate credentials?	6
5	How do you handle security incidents?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Release Eng	Security gates in CI/CD
DR	Secure backup storage
Documentation	Security runbooks needed

FOCUS AREAS

- → **Secrets:** Vault, rotation, no hardcoding
- → **Certs:** Auto-renewal, short expiry
- → **Scanning:** SAST, DAST, dependency scanning
- → **Zero Trust:** Verify explicitly, least privilege

Security as Reliability

Secure systems are reliable systems.

Domain 14: Documentation

Architecture Diagrams, Runbooks, ADRs

All Teams | Governance | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Tribal knowledge; outdated docs; no runbooks
2	Some docs exist; quality varies; runbooks partial
3	Architecture documented; runbooks for critical paths
4	Docs as code; ADRs tracked; runbooks tested
5	Docs auto-generated; executable runbooks; always current

ANTI-PATTERNS (RED FLAGS)

- → Knowledge only in people's heads
- → Docs abandoned after creation
- → Runbooks that don't work
- → No architecture diagrams
- → Decisions not recorded

EVIDENCE CHECKLIST

- → Architecture diagrams exist and are current
- → Runbooks linked from alert definitions
- → ADR repository maintained
- → Documentation review process exists
- → Onboarding docs enable self-service

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	How current is your architecture documentation?	6
2	Do runbooks exist for all alerts?	6
3	How do you track architecture decisions?	6
4	How do you keep docs up-to-date?	6
5	Can new team members onboard via docs?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Alerting	Alerts link to runbooks
Incidents	Runbooks aid response
Dependencies	Service maps document deps

FOCUS AREAS

- → **Architecture:** C4 diagrams, service maps
- → **Runbooks:** Linked from alerts, tested
- → **ADRs:** Decision records with context
- → **Freshness:** Regular review cadence

Docs as Code

If it's not documented, it doesn't exist.

Domain 15: Dependency Management

Service Maps, Vendor SLAs, Dependency Health

SRE Bot | Release | Max 30 Points

0-6	7-12	13-18	19-24	25-30
AD-HOC	FOUNDATIONAL	STANDARDIZED	ADVANCED	OPTIMIZED

SCORING CRITERIA BY LEVEL

LEVEL	CRITERIA
1	Unknown dependencies; surprise failures from vendors
2	Partial dependency list; some vendor tracking
3	Service maps exist; vendor SLAs tracked; alerts on deps
4	Dependency health dashboard; degradation strategies
5	Auto-discovery; vendor SLA enforcement; antifragile design

ANTI-PATTERNS (RED FLAGS)

- → Unknown critical dependencies
- → No vendor status monitoring
- → Single vendor for critical path
- → Outdated library dependencies
- → No fallback for vendor outage

EVIDENCE CHECKLIST

- → Service dependency map exists
- → Vendor SLAs documented and monitored
- → Dependency health dashboard available
- → Degradation strategies for critical deps
- → Library dependency scanning automated

ASSESSMENT QUESTIONS

#	QUESTION	MAX
1	Do you have a complete service map?	6
2	How do you track vendor SLAs?	6
3	How do you monitor dependency health?	6
4	What's your strategy for vendor outages?	6
5	How do you manage library dependencies?	6

RELATED DOMAINS

DOMAIN	RELATIONSHIP
Reliability	Circuit breakers for deps
Observability	Track dependency metrics
Security	Library vulnerability scanning

FOCUS AREAS

- → **Service Maps:** Visual dependency graphs
- → **Vendor SLAs:** Tracked, compared to internal SLOs
- → **Health:** Dependency health as metric
- → **Degradation:** Graceful handling of dep failures

Know Your Dependencies

Your SLO is bounded by theirs.