Operations Maturity Model: Current Incident Management and Disaster Recovery

	Realistic Recovery Time	12+ hours	1-12 hours	1-60 minutes	6 seconds (BGP convergence) - 10 minutes
	Serious Incident Threshold	**"Something is wrong!"**	**"Something is wrong with the database, storage, and network!"**	**"Something went wrong with your deployment!"**	**"The core infrastructure fabric is down!"**
Overall	"Which teams have regular on-call responsibilities?"	NOC and direct Ops staff, some additional management	NOC and direct Ops staff, small number of teams outside of Ops	24/7/365 for most teams with production systems; integration with corporate teams (HR/Internal Security), security incident management	24/7/365 Company Wide, Integrated, All Hazard
Overall	"What is expected of someone on-call?"	NOC staff "expected to respond immediately"; anyone else best effort (with post-facto punishment for slow response)	Formalized primary on call within specific teams	Common standards and expectations for all teams for primary, secondary… roles	Common standards and expectations for all teams for primary, secondary… roles. Self-managed
Overall	"How are people notified and engaged on an incident?"	Dedicated staff monitoring for alerts and handling first response; email, phone, best effort attempts to engage outside of NOC	NOC and semistandard notification tools tied to trouble ticketing system	Consistent, standard; "Please page database-core-primary, network-primary, search-primary, corp-it-primary, … and have them join conference bridge #11323"	Increasingly automated and integrated into regular monitoring tools, so that smaller and smaller groups are required to resolve complex issues.
Incident	Use of Common Terminology	Inconsistent even among teams	Some team/tribal knowledge and standards, not well mapped outside team	Common, consistent, well-known, easily understood	Part of company on-boarding process & regular refreshers
Incident	Establishment of Incident and Command	Often unclear who is in charge; many people may pile on or take disconnected action	Dedicated "person in charge" to take control; may be long delays while that person shows up	Clearly established at beginning of any incident, regardless of title/role; first person to respond is IC until command is transferred	Cultural impacts… "If nobody is in charge… you are in charge" begins to spread.

Incident	Unity of Command	Ineffective – reporting roles not clear; management may arrive and issue conflicting directives without assuming command	Ineffective; reporting roles not clear, management may arrive and issue conflicting directives without assuming command	Everyone involved reports to a single supervisor and receives directives from that supervisor alone.	Reduced chaos & increased effectiveness makes cowboy behavior and outside management intrusion increasingly intolerable.
Incident	Transfer of Command	Rare informal handoff when it occurs	Semiformal handoff as required; processes may be documented but not rigorously followed	When command is passed from one person to another, it is done following a process which includes a briefing and communication of current status.	More experienced leaders mentor junior people managing large incidents rather than transfer command.
Incident	Roles and Positions	Tied to specific people with specific jobs	Tied to specific people with specific jobs; semiformal roles coupled with tribal knowledge	Well defined, in many cases predefined with available training and checklists	Well defined, including advanced roles such as PIO, liaisons, specialized logistics and operational roles.
Incident	Communication of Objectives and Current Status	Limited to tactical understanding of people working on a problem; irregular status updates; frequent interruptions	Some regular communication of current objectives and progress	Everyone involved knows what the current status and objectives are for the incident, along with their individual responsibilities.	Tooling emerges to provide better automated reporting and status capabilities.
Incident	Organizational Structure	Expanding constantly as incident duration and visibility goes on; once engaged, responders staying till incident is close	Incident expands constantly, but some people are released once it is clear they are no longer required.	Expanded and contracted to evolve with size and complexity of the incident until it is eventually concluded	Average incident size becomes smaller over time as people and tooling get better for engaging right people quickly and releasing them when no longer required.
Incident	Planning and Documentation	Very limited; “too busy to write it down”	Tactical documentation, basic logging of some systems	Current status, issues, objectives, progress, and resources involved with an incident are tracked and frequently communicated; next actions are planned and communicated as part of a regular cycle	Pre-plans emerge for more complex incidents. Postmortem review processes identify meta problems/themes to be resolved improved over time.

Incident	Span of Control	Organization is flat under single manager and with many responders trying to work with limited management; frequent contention/confusion	Some ability to split into smaller teams, typically around specific task groups (database, network, etc.)	As incident expands, teams are split to keep under manageable span of control (3-7 people reporting to them at any given time)	As incident expands, teams are split to keep under manageable span of control (3-7 people reporting to them at any given time), leaders with special training step in to assist or take over.
Incident	Integrated Communications	De facto communication channels available/used by some groups; not shared; some people working separately from incident, individually or in groups	Defined communication channels available/used by some groups; not widely shared; may be closed or unreliable	Standardized, well known, reliable, frequently used, well understood, easily interoperable, universally accessible. (Often recorded)	Standardized, well known, reliable, frequently used, well understood, easily interoperable, universally accessible (often recorded)
Incident	Information and Intelligence Management	Responders too busy to provide regular updates	Frequent interruptions/ disruptions to get status updates, make sure “right level of urgency,”provide outside information, etc.	Dedicated incident communication roles, including PIO (internal facing), liaison roles, deputies assigned to get /track specific information	Senior Management gravitate toward PIO and liaison functions, allowing them to work across org boundaries in incidents.
Incident	Accountability	Informal, no core ability/ retirement to track who is engaged, who has been requested, what people are doing	Limited with dedicated person to track who has been requested; who is engaged currently	Every person involved in an incident performs to common standards, including how they respond when dispatched and how they check in and receive assignments	Tooling emerges to better track the tactical status and long term operational performance of individual responders, teams, and the management chains they report through.
Post-Incident	Postmortem Focus	Ad hoc – focused on human-error blame finding and punishment	Formal, focused on blame finding/avoidance & punishment	Formal, officially blame-free	“5 whys”, including business and cultural issues.
Post-Incident	Root Cause Orientation	Focus on triggering event typically blaming specific operator error or specific hardware failures	Focus on triggering event or human error”; blaming process and or infrastructure	Primary focus on on underlying technical root causes, systemic fixes	Primary focus on unique insights and opportunities from lessons learned, application to other areas within infrastructure

Post-Incident	Root Cause Mitigation/ Resolution Approach	Cycle between protecting heroes and then firing them – "three strikes" rules, etc.	"Let's implement more process and overhead," "let's add more and more layers of review," and "lets spend more money to prevent this problem"	Improved tooling programatic checks, operator tools for special cases. Some focus on building Resiliency, partial	Increase resiliency, increase appropriate operator tools, focus on self-healing fixes.
Post-Incident	Root Cause Elimination Rate	<10% – mostly break fix detection	10% within 3 months – mostly simple fixes; tracking but little progress against goals vs. other priorities; frequent recurrence	20% – of easily fixable issues eliminated within 3 months, programs to eliminate larger issues over time	80%+ resolved within 3 months, remaining have programs to fix architectural problems. Recurrence is infrequent and a very big deal.

	Realistic Recovery Time	**1-5+ days**	**24 hours**	**1-12 hours**	**0-1 hours**
	Disaster Threshold	**"We lost the SAN!"**	**"We lost both of the SANs!"**	**"We lost a facility in Virginia"**	**"We lost Virginia!"**
Disaster Recovery	Overall Posture	"We made a plan for that a long time ago."	Some critical systems backed up and recovery process understood by key practitioners. Other systems unlikely to recover easily.	Failure happens. Some critical systems resilient to single facility failure, most critical systems can be failed to alternate facility within 4 hours.	Failure happens. Multi-site resiliency built into every level of stack and exercised as regular part of doing business (often part of regular deployment cycle) - Exceptions regarded as threat to the business.
Disaster Recovery	Overall Scope	Primarily database/ storage tier backups	Storage + limited amount of cold standby	Warm standby w/ increasing amounts of active/active with facility isolation and fast-failover.	Comprehensive: all teams, systems, and functions
Disaster Recovery	Disaster Threshold	Disruption/destruction of single database or SAN	Disruption/Destruction of multiple databases instances or SANs	Destruction/Disruption of a single facility	Simultaneous Destruction/disruption of multiple facilities. Software failures resulting in data loss or fabric failure.

Disaster Recovery	Exercise Frequency	Never	Rare	Quarterly, increasing in scope complexity & difficulty.	Constant at small scale, regular large-scale exercises with ever increasing complexity & challenge
Disaster Recovery	Exercise Model	Conceptual/tabletop with partial execution of some recovery components	Full execution of planned recovery activities against smaller DR site/fleet. Not brought into production.	GameDay: Planned failure injection against critical systems escalating to full data center level failure.	GameDay: Full scale live fire exercise with no notice during high production load - no big deal.
Disaster Recovery	Recovery Confidence	Practitioners: “I don’t know but it would be pretty bad”; Management: “We’d figure it out.”	It’s hard & won’t go perfectly, but we can probably figure it out”.	“It will work, it did last few times we did it”.	“Of course, we test failure modes as part of our deployment cycle.”
Disaster Recovery	Exercise Success against RTO	Unknown – never attempted	Limited: Most data recovered w/ some survivable losses, app rebuilt but not able to put into production.	Partial: All data recovered, app put into production load but at reduced capacity and/or with some diminished functionality.	Approaching 100%
Disaster Recovery	Cost, Impact, Danger to Attempting Recovery	Unknown/likely very high – substantial downtime, data loss, system inconsistencies, insufficient capacity, limited knowledge	High - Down time, data loss, capacity shortages, inconsistencies	Medium - Some down time & small amounts of data loss. Manageable capacity coupled with easy repurposing. Possible degraded mode operation.	Negligible. In some cases positive. Able to take entire facilities offline for rolling upgrades, repurposing for capacity management
Disaster Recovery	Resistance to Initiating Recovery	Extremely high – not even clear when it should be implemented	High - Much time/effort will be expended trying to scope/fix the problem rather than spin up DR effort.	Low - Clear standards when required and comfort with exercising plan.	Irrelevant for many systems - Operating systems + app stack are reinstalled at reboot for most systems.
Disaster Recovery	Practitioner Posture	Fearful, denial, defensive	Cautious / Concerned	Accepting	Indoctrinated/Ready
Disaster Recovery	Executive Posture	“Priority for next year”	“Top priority right now, even though it is expensive to make happen.”	“Every team knows this is critical, baked in to SDLC and planning cycles. We take this seriously”	Source of pride, confidence, marketable competitive advantage

Operations Maturity Model: Current Infrastructure and Operations

	Mean time to Production	4-12 weeks	1-4 weeks	1-5 days	0-5 minutes
	Challenge	**Automate common tasks: Scripts, OS Compliance, Updates, Patches**	**"Configuration Management"**	**"Application Management"**	**"Continuous Delivery"**
	SysAdmin to Server Ratio	**1 SysAdmin to 25-250 systems**	**1 Systems Engineer to 250-500 Systems**	**1 Systems Engineer to 500-1000 Systems**	**1 Site Reliability Engineer to 1000+ Systems**
	Constraint	**Human ability to perform complex repetitive tasks**	**SE time to automate vs. "just getting it done"**	**SE time to support Applications & Dev teams**	**Data center hardware manufacture, delivery and installation and/or IaaS provider capacity**
Hardware	Reliability	"Every server sacred" - High availability support expectations across entire stack, support contracts, dependence on vendor or onsite SE for onsite replacement and maintained.	Limited HW support on some systems with fleet redundancy. High availability support expectations across most of stack, support contracts, dependence on vendor or onsite SE for onsite replacement and maintenance.	Limited HW support on all systems, onsite sparing strategy w/ quick response from low-skill onsite staff to fix normal failures.	Fully commoditized, no HW support, regular (automated) replacement of equipment after failure detection by low-skill staff.
Hardware	Requisition Process	Request Form + Architecture Review Process For Everything	Request Form & w/ Arch Review Process for new services only. Regular scaling cycles.	Exceptions managed with Arch Review. Automated request process + Emerging Data center API for everything else.	Data center API (Infrastructure as a Service) + cost accounting for almost everything. Special cases tend to be CorpIT / Windows services and/or non-standard hardware (FPGA / GPU environments in finance)
Hardware	Storage Model	"This is the best SAN money can buy", total dependence on vendor and support contracts	"The best cluster of SANs money can buy"	Software partitioning and replication strategy to span multiple SANs	Built in Software - w/ Storage as a Service.
Hardware	Standardization	Every project special snowflake	Limited SKU standardization - Difficult to repurpose equipment	Extensive SKU standardization - Hardware is easily/frequently repurposed.	Complete SKU standardization, special cases regarded as risk to business.

Hardware	Data center Facilities	“Tier 1”, Expensive, Inefficient, Power or Space Constrained & Poorly utilized, Difficult to repurpose. Lots of surplus redundancy to maintain HA.	(Mostly the same) “Tier 1”, Expensive, Inefficient, Power becomes constraint due to HW density. Difficult to repurpose. Lots of surplus redundancy to maintain HA.	Smaller more efficient data centers, power & bandwidth constrained. Limited redundancy w/ Hot / Hot / Hot (N+1/N) Availability strategy,	“Redundant Array of Inexpensive Data centers”
Operating System	Management	Many flavors, many versions with unclear support, upgrades take years if ever.	Limited flavors, some standardization, many versions supported, upgrades driven on quarterly/yearly cycle. Largely a manual effort with some patch-management.	Standardization, few supported versions built around a Just Enough Operating System (JEOS) concept, upgrades happen on regular cycle including regular reboots	Internally maintained, released, managed. Upgrades constant as infrastructure is rebuilt frequently.
Operating System	Patch Management	Manual, difficult, dangerous, irregular	Manual, difficult, less-dangerous, semi-regular	Automated, programmatic	Irrelevant for many systems - Operating systems + app stack are reinstalled at reboot for most systems.
Infrastructure	Configuration	Manual - basic installation and ad-hoc scripting.	Partially scripted installs of golden images. Post-install scripts and lots of manual cleanup.	Automated installs based around JEOS. Very high success rates for deployment.	Full stack automation, deeply integrated with deployment and other privisioning systems.
Infrastructure	Source of Truth	Tribal Knowledge + Out of Date wiki	Tribal knowledge + Wiki (Manually updated… occasionally), ad-hoc tools, basic hardware CMDB	Proliferation of disconnected domain/team/org specific “Sources of Truth” - Data center, Hardware, OS, Security, Patch, Image, Software, App, etc. + Tribal Knowledge + Wiki + Ad-hoc tools	Central Source of Truth as hub with APIs connecting other systems. Full stack automation from application to bare metal. Ad hoc tools using API. Self documenting… Infrastructure as Code.
Infrastructure	Services (DNS, CDN, Auth, PKI, etc.)	Ad hoc, manually configured, poor understanding of how they work, limited operational knowledge, no service owners	Core services have service owners. Limited internal automation.	All services have service owners. Partial automation and some APIs for configuration.	Fully automated - Integrated into App framework, managed programmatically.

Provisioning	Physical Servers	Manual - Order, Receive, Unbox, Install Components, Rack, Stack, Cable, Configure BIOS/Storage	Manual - Order, Receive, Unbox, Rack, Stack, Cable, Configure BIOS/Storage (But components installed at Factory & no need to repurpose/burn in)	Semi-automated - Order, Receive, Unbox, Rack, Stack, Cable	Fully automated / Horizontally partitioned. Just add more servers and allocate them as storage.
Provisioning	Base Operating System	Manual - Many options must be selected at console and requires physical media	Manual w/ network mount - Possible to mount base OS install images as virtual disks. Standard OS install run via console.	PXE + Kickstart w/ limited access console	Unattended installation of Just Enough Operating System(JEOS) driven by CM w/ no console access for all machines.
Provisioning	Storage	Specialized, Infrequent, Difficult/Expensive - Largely managed by vendor to maintain HA support.	Partially standardized. Largely managed by vendor to maintain HA support.	Partial automation - Horizontally partitioned across specialized storage machines. Storage/Cluster management.	Fully Automated / Horizontally partitioned. Just add more servers and allocate them as storage.
Provisioning	Network Edge	No real of “Edge” - Manual, Complex, Non-standard configs, lots of L2 VLAN spanning.	Mostly manual. Some “Edge” standardization, efforts to limit L2 VLAN spanning.	Semi-automated. TFTP boot configs standard, rare L2 spanning.	Simple & Fully automated - Integrated into rack level provisioning model
Provisioning	Load Balancing	Manual, Complex, interdependent, fragile	Manual w/ some convention, complex but increasing isolation from other components, fragile.	Semi-automated. Managed in part programmatically with some applications making API calls to configure. Isolation. Less fragile.	Fully automated - Integrated into App framework. Robust.
Provisioning	Network Core	Manual, Complex, limited standardization or convention between neteng people/teams “Scary” and/or “Magic”. No ability to monitor/inspect. Tribal Knowledge only.	Manual, complex, Some standardization. Limited monitoring, heavy dependence on tribal knowledge and “Magicians”	Semi-automated. Deep instrumentation, monitoring, metrics. Common management tools. Programmatic management of ad-hoc changes.	Managed programmatically, fully automated. (Still “Scary” and/or “Magic”)
Network	Topography	Extremely complex VLAN spanning racks to core.	Flattening of VLANs minimizing complex inter-switch spanning.	Partial Rack level isolation, Flat VLAN, L3 everywhere possible, BGP for services w/ GSLB concepts standard	Complete Geo/Facility/Rack level isolation, Flat VLAN, L3 everywhere possible, BGP for services w/ GSLB concepts standard

Infrastructure	Security	Primarily focused on data center security, access control, and compliance requirements.	Data center security, access control, compliance + perimeter security and limited auditability. Some IDS.	Advanced/active IDS / Persistent Threats, DDOS response/mitigation.	Defense-in-depth.
Vender	Relationship	Strategic / Captive / Locked-In	Diminishing Dependence / Tactical	Empowered / Competitive	Open Market / Liberated
Common	Instrumentation & Metrics	Manual, inconsistent, isolated, domain specific. Used exclusively by Operations staff when available.	Basics standardized at hardware / infrastructure layer. Some common application metrics, largely created reactively by operations staff.	Common metrics across all services, proper instrumentation driven as a production requirement for all services	Metrics built into core platform, included by default in almost every case regardless of Dev, test, or prod.
Common	Deployment Cycle	Monolithic software stack built, tested, deployed on irregular cycle	Monolithic stack, built, tested, deployed on regular cycle	Stack split into services deployed on increasingly frequent cycle	Continuous Deployment - Versioned services / API endpoints
Common	“Who is accountable for app availability?”	The individual Operations staff member who deployed it	Operations	Operations + Service Owner	Service Owner
Common	“Who is accountable for app performance?”	Not considered unless causes problems at deployment	Managed by Operations, driven back to software team	Service Owner w/ Ops help	Service Owner
Common	“Who writes app software”	Devs	Devs (responsible for specific areas)	Dev (Service Owner)	Dev (Service Owner)
Common	“Who deploys app software?”	Operations	Operations w/ coordination with Dev teams	Service Owner w/ Ops help	Service Owner
Common	“Who monitors app software?”	If it is monitored, Operations monitors it	Operations	Service Owner w/ Ops help	Service Owner
Common	“Who feels pain when app breaks?”	Operations	Operations + Very small subset in Dev as escalation	Service Owner w/ Ops help	Service Owner
Common	“What happens when app breaks?”	Operations rolls it back	Operations staff rolls it back, escalates for fix within software team	Service Owner rolls it back w/ some Ops help, ultimately responsible for fix	Automatically rolled back.

Common	“Who fixes software when it breaks?”	Operations staff requests fix	Operations staff drives fix within Development org	Service Owner responsible for fix, with assistance from SRE	Service Owner
Common	“When is Dev done with app software?”	When the Dev check it in	When it has passed regression test	When it has been deployed	When they have deployed it