CLOUDSOFT

RESILIENCE THROUGH APPLICATION MANAGEMENT

KNOW WHERE YOU STAND TODAY FOR A MORE RESILIENT TOMORROW

ALEX HENEVELD (CTO) &
ALASDAIR HODGE
(PRINCIPAL ENGINEER)

JANUARY 2021

Alex Heneveld

Co-Founder & CTO

Alex brings 20 years experience designing software solutions in the enterprise, start-up, and academic sectors. Alex was with Enigmatec Corporation where he led the development of what is now the Monterey® Middleware Platform™. He founded PocketWatch Systems, commercialising results from his doctoral research. Alex holds a PhD (Informatics) and an MSc (Cognitive Science) from the University of Edinburgh and an AB (Mathematics) from Princeton University.

Alasdair Hodge

Principal Engineer

Alasdair is a solutions architect with 25 years' experience. An authority in cloud, software applications and automation across all major cloud platforms, he has been engaged in the design and optimisation of cloud services in banking and finance and other service-based sectors such as telecoms, electronic design and supply-chain automation for over 12 years.

Introduction

In today's environment, companies are facing unprecedented challenges, with two vital but often contradictory pressures:

> Customer-facing applications, internal applications and all IT systems need to run well, reliably and at scale

> Application development needs to be agile to respond quickly to changes in the business priorities

Cloudsoft are a leading high-end resilience software and consultancy company who work with companies of all sizes, from financial services to telcos and aerospace, to resolve the tension between reliability and agility.

In this eBook, we share some of our main learnings about the processes and tools that have the biggest impact on maximizing resilience and agility while keeping a good balance. Chief among these is application management, a solution that is often overlooked but which we have found to be one of the most powerful, able to give governance and visibility up and down the stack, and the area which we've specialized in for over ten years.

Introduction: Why is Resilience so Hard?

Everyone in the technology industry knows how important it is to make systems resilient: if people in a company can't use their systems, they aren't productive; and if customers can't access them, or if the systems are slow, then the customers can't make purchases, revenue goes down, and the company's reputation takes a long-term hit. In regulated industries, there can be additional swingeing penalties imposed by regulators, and, increasingly often, systems are interdependent and one minor outage or slowdown somewhere obscure can have a severe impact on a mission-critical system downstream.

Given how vital resilience is, why do so many applications go down so often?

Let us start by challenging the question. Most systems are pretty reliable. But "salience bias" — the squeaky wheel effect — means the failures get undue attention, in users' mind and on social media. Even a short period of downtime for a bank or credit card company gets headlines, but no one notices if they work for ten years straight. Keeping downtime below 5 minutes requires a lot of nines:

This is compounded by complex interplay between systems, especially with the rise of microservices and container architectures, where suddenly if 100 systems need to function to deliver a result, in order to get five nines reliability on the result, you need a mean (geometric mean) of seven nines on each of the microservices:

(0.9999999)^100

=

0.9999000000494

The solution in most cases is to architect differently: design so that systems can tolerate failure of individual components or degrade gracefully. This poses a completely different challenge:

To design and assess the reliability of a system, it is necessary to understand not just what it depends on, but how it depends on upstream systems and how it copes and recovers in the event of various failure conditions.

The complexity becomes such that it's no longer sufficient to have "the smartest person in the room": knowledge is needed up and down the technology stack, requiring lots of smart people to be talking and listening to each other. Subject matter experts are needed in networking, compute, storage; mid-tier, data systems; security policies and compliance requirements; and increasingly, to have knowledge of new platforms such as Kubernetes and clouds.

Expertise in resilience itself also becomes much more important. Many common good practices can help with design, and there are established concepts that help in evaluating resilience options.

Plan for failure

One of the singular best pieces of advice for resilience is to design defensively, anticipating failures and develop the code so that it manages as best it can in the presence of failures (rather than make things worse when things go bad!)

Recovery point objective (RPO)

Engineers enjoy solving problems, and ensuring no data loss even in the most extreme situations is an enticing challenge for some. However it is important to consider whether it is necessary in each case; for an e-commerce site, for instance, losing 5 minutes of orders per year is likely to be easily solved by manual processes (when the customer complains). It might be possible to get the data from other sources (e.g. credit card charges and reviewing confirmation emails), it's usually possible to mitigate the problem (e.g. email users who were active prior to the outage to apologize that orders may have been lost), and often much cheaper to pay out (e.g. give them the order for free, rather than run super-high availability)

"CAP" and "PAC/ELC"

In an ideal world, all data read/writes would be consistent (C) across clients and always available (A) to them, even tolerating partitions (P) if networks are down.

The "CAP Theorem" states the mathematical truth that is obvious with a bit of thought: you can have at most two. Research in this area is very important to resilience design; for example the PAC/ELC extension asks "in the event of a network partition (P), do we maintain availability (A) but allow inconsistency (different clients will see different data), or do we maintain consistency (C) but make the data store unavailable to clients on the wrong side of the partition", and "else (E) in normal times, when the network is healthy, do we optimize for latency (speed - L) but allow brief inconsistency (so-called "eventual consistency"), or for consistency (C) across all reads and writes at a cost of speed. It's a complicated topic, but a fundamental consideration that often has a major impact on resilience, and requires an understanding of both user and business requirements.

More green and less red

Uptime — the number of nines — is improved not just by preventing failures (longer "green" periods), but also by optimizing the recovery time when there are problems. Thus investment in shortening downtime periods ("reducing the red") pays big dividends. A general ability to perform rapid recovery ("repave") for applications is a useful tool in the inevitable event of low-probability and unknown outages.

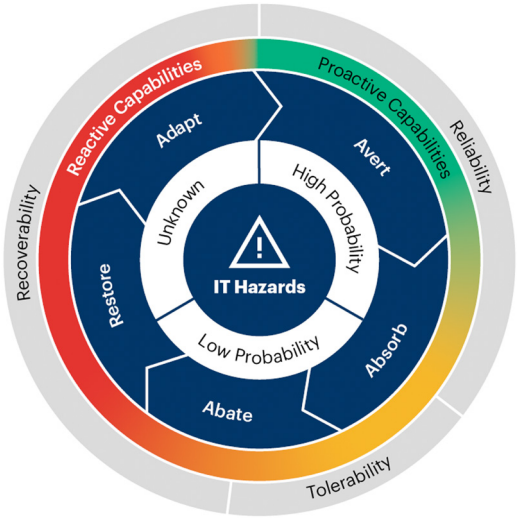
Gartner's IT Resilience Framework

Figure 2: IT Resilience - Reliable + Tolerant + Recoverable

Before we go any further, it is important to define what resilience means. Gartner's definition[1] (Predicts 2021: Value Streams Will Define the Future of DevOps) is a good one:

IT resilience is the organization's ability to anticipate, detect, assimilate and adapt to IT-related hazards — such as application defects, performance thresholds, security vulnerabilities, single points of failure and service provider outages — and to continuously improve capabilities to avert, absorb, abate and recover.

This expresses it in quite concrete terms and emphasizes the importance of continuous improvement. However in practice there are tremendous trade-offs between averting and adapting, and resilience is so hard because so many different factors come in to play. Business impact analysis (BIA) requires subjective value judgments and strategic thinking considered together with a deep understanding of technical costs of possible solutions, encompassing all aspects of the technical design. In addition, people's time is a scarce resource, and resilience, much like insurance, is something often viewed as a dull, tick-box exercise. The only way to achieve resilience is to engage all stakeholders, prioritize it, foster good communication, and put in place processes that facilitate shared understanding.



Source: Gartner
733708_C

[1] Gartner, Predicts 2021: Value Streams Will Define the Future of DevOps, Daniel Betts, Chris Saunderson, Ron Blair, Manjunath Bhat, Jim Scheibmeir, Hassan Ennaciri, October 5 2020.

Application Management: an Innovative (but Obvious) Approach to Resilience

One of the most powerful trends in resilience is shifting the focus from infrastructure to applications: instead of looking closely at the technology, hardware, and platforms, resilience should start with the logical application components from end-user delivery through to the back-end data plane, the responsibilities of these logical groupings à la microservices, the theoretical failure modes and mitigation for each, and their actual performance.

This makes the topic comprehensible to all the stakeholders, naturally directing conversations to questions from different functions:

Development:
How should the application be made resilient?

Architecture:
Which resilience patterns apply to this application?

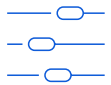
Infrastructure:
Will the application cope with a failure at this device?

Security:
Where is the data stored, is it encrypted and safe from hackers?

Operations:
Is any part of the application running sub-optimally?

Finance:
How much more expensive is a lower RPO?

Business:
What is the effect on users if the application is unavailable?



Process:
Can we demonstrate resilience to the regulator?

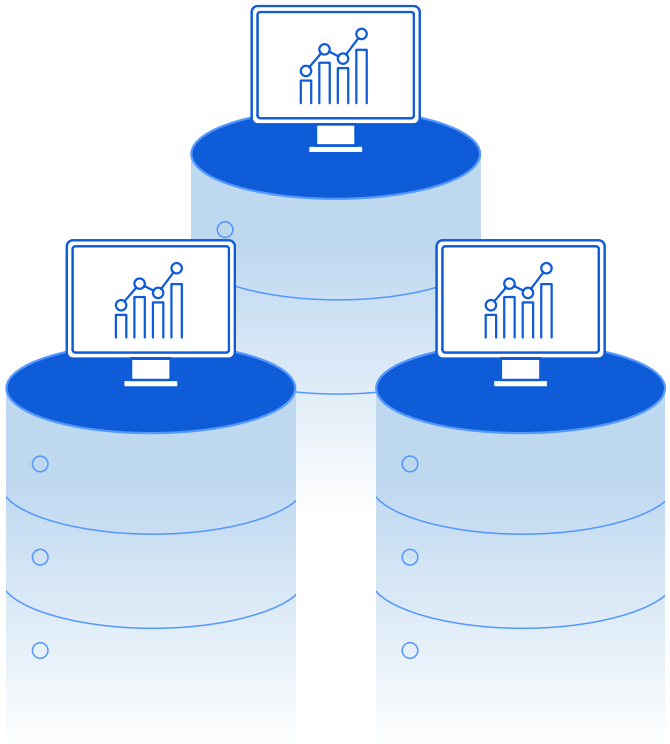
It will never be the case that everyone will understand everything, but with a top-down application-first focus, the discussion is more accessible and the topics more important. The developer who doesn't know the concept of an RPO is a bigger problem for a resilience strategy than the CFO who doesn't understand virtual networking.

An application-led approach is not to say infrastructure isn't relevant. It absolutely is, as problems with compute, storage, networking won't go away, and it is essential to be able to see easily how logical application components map to them. However infrastructure is only one of many supporting components, and the common focus on them for resilience can distract from more important issues and exclude important voices from the conversation. These should be considered alongside other supporting components, including software technologies, platforms, and services, and viewed from an overarching perspective of delivering the business requirements. By framing the problem in terms of applications, it becomes much easier to share the knowledge brought by each different subject matter expert.

Another major benefit of this shift is that common solution patterns for resilience become apparent: even if the database systems are different, or the underlying platforms are different, many of the same high-level architectural designs for resilience can apply. Again, the shift isn't intended to prevent resilience from being considered lower down the stack, e.g. running multiple dedicated network links for high availability, but for these options to be considered in light of the overall application requirements: if an application can run fine with mere "eventual consistency", there is no need for a super-high-availability network, and the super-high cost should be avoided.

By teasing out these aspects of application architecture and requirements and making them explicit, and by finding commonalities across systems, it becomes simpler for all the stakeholders to have the understanding they need. This is important up front, when collective knowledge and evolved best-practice patterns can lead to a better resilience solution out of the gate; however the application-focus and emphasis on common patterns continues to pay dividends throughout an application's lifecycle. Where runbooks are similar, operations become more familiar and thus reliable, and recovery time goes down. And if, for instance, an operations team can see that eventual consistency is allowed for some applications, then in the event of a network slowdown, they can prioritize those applications which need full consistency. When an infrastructure upgrade is planned, or a migration being considered, the affected applications and the risk to them can easily be seen. When the regulators come knocking, the evidence of resilience is easier to come by.

And with commonalities comes the ability to evolve faster: any improvement, in any aspect of resilience design, can be more easily shared. Investment in automation, in fire-drills, in new technology, these can be done not just to one subsystem of one application, but to a pattern which can be rolled out to every application that uses that subsystem. With agility so important to companies, investment in testable, systematic, widely-understood, application-led resilience is vital: the best processes for resilience make it easier for the impact and risk of new technologies to be assessed, and allow that assessment to be done once for a wide range of applications. This investment also allows new technologies and ideas to be applied to resilience itself, causing it to get stronger over time rather than rot as is often the case with disparate, static runbooks.



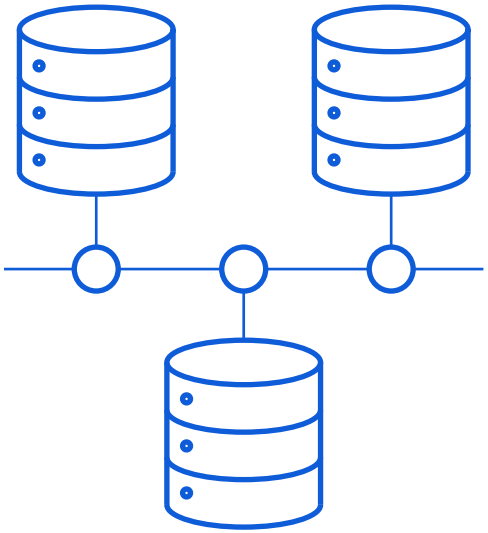
The Cloudsoft Application Resilience Maturity Model

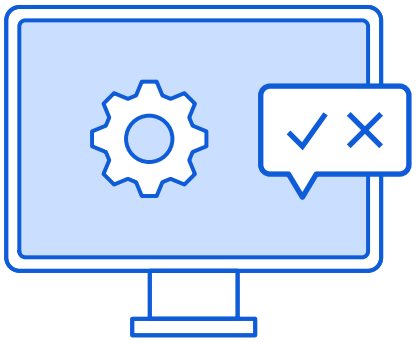
The intelligence and data gained during our time as consultants working in the field and developing application availability software has led to the foundation of our Application Resilience Maturity Model. The model assists organizations by helping them to identify where they are on their resilience journey and highlights the next critical steps required to improve operations. When organizations understand the stage they are in, they can fully appreciate where they sit in comparison to competitor brands and the risk facing their business.

Stage 0 Silos

Each application or line of business determines their own technology stack and operational processes including resilience.

Over time, some applications have become very reliable, but many are not. There are no shared learnings or processes, the technology for applications and resilience is extremely heterogenous and of variable quality, resulting in frequent fire-fighting activities to ensure all critical applications are available.



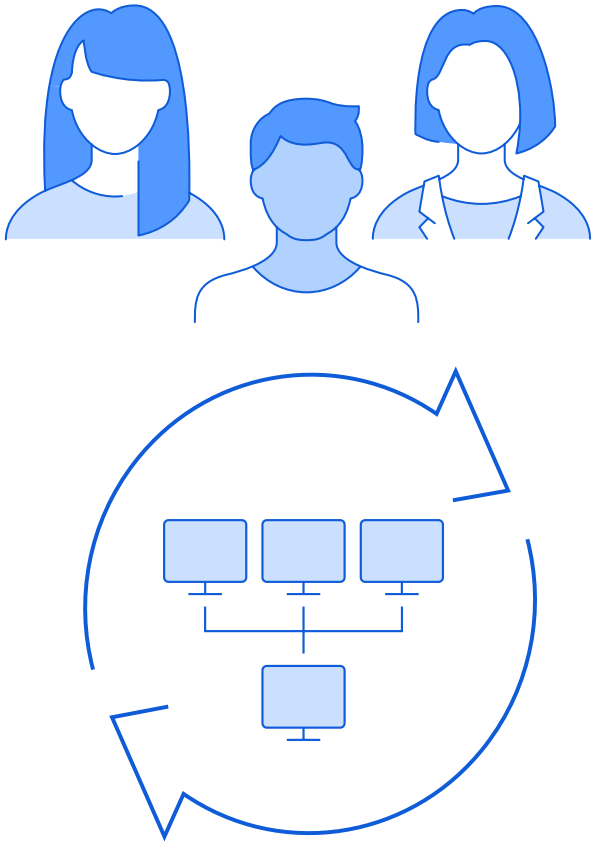


Stage 1 Architecture Review Board (ARB)

Much of the environment remains the same a stage 1 but with the addition of a further process, ensuring every application that moves to production must be approved and periodically re-approved by the ARB. This starts to encourage setting standards for all applications to achieve, especially if the requirements for approval are clear.

Typically, critical requirements for applications will include alerts on failures, documented runbooks and scheduled manual fire-drills. A pragmatic architect or engineer will examine approved applications when building new ones, reusing existing and approved architectures and runbooks, although many will not - believing it would be easier to write a new document than to refactor an existing one. This flaw in the process harbors the continuation of heterogeneity.

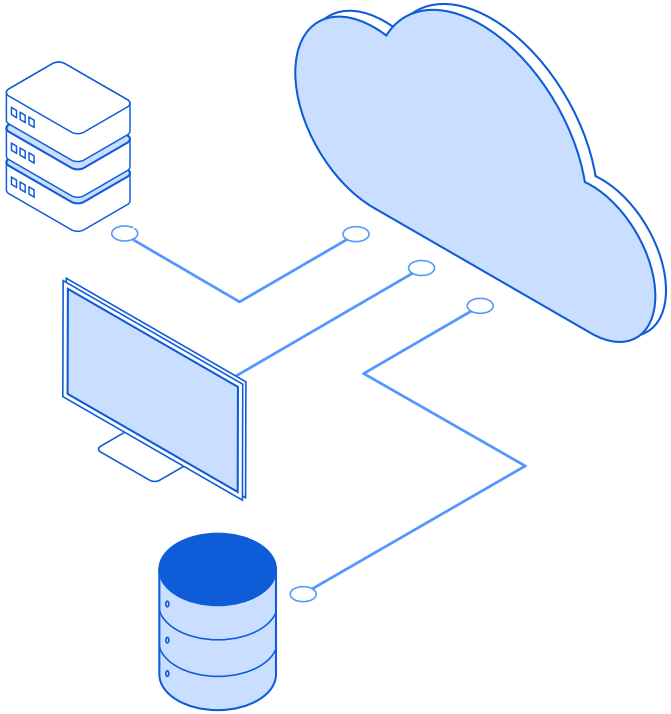
The problem may be reduced by fiats which mandate specific technologies, but it requires proactive review and the introduction of processes to prevent such fiats from impeding innovation. Improving resilience plans is time-consuming, difficult to share upstream and there is little incentive to invest in continual improvement so everything tends to be “just good enough”, operating with infrequent panics that require resolution by application experts.



Stage 2 Shared Site Reliability Engineering (SRE) function

Resilience maturity starts when there is a team dedicated to delivering it across all applications. This function may act as the ARB (2) or have members on the ARB along with other stakeholders. Most of their time however is spent working with the app teams during development and after release, to apply good and standard practices where appropriate.

This function is often able to start to develop tools and paper-patterns that app teams can use, and they are able to feed back to application teams when improvements have been identified. This function may perform fire-drills more often, following runbooks without involving the application teams so availability is improved and recovery is much more reliable.

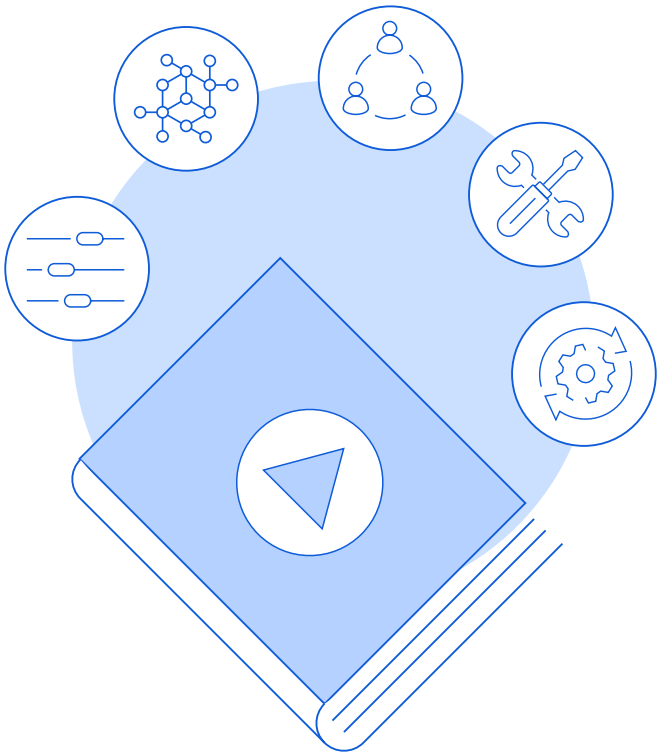


Stage 3 Modern platforms and architectures

Standardized approaches to using containers, Kubernetes, PaaS platforms, cloud environments, serverless and micro-services are advised by the shared SRE function (3). Applications are written or modernized to move into these platforms, which provide increased and standardized resilience.

As this space is fairly new, there are many anti-patterns - expecting any one platform to be a silver bullet or to automatically be resilient. Unfortunately, although they make applications resilient if configured correctly, there's a lot of work involved in running these platforms in a resilient way. The use of any given platform or architecture has to be a part of a modern resilience strategy and organizations must accept they won't work for every application. If the SRE function (set up in stage 3) isn't involved in their curation, the organization slides back to the starting point of the Application Resilience Maturity Model.

The SRE function is not replaced by modern platforms and architectures, especially for applications which straddle platforms, but in many instances, the role of the SRE function is much simpler than at stage 3. This means availability and recovery both improve, although it can still be a lot of work to develop applications and resilience plans. It can also be difficult to correlate failures up and down the stack or identify the appropriate runbook in production.



Stage 4 Application resilience modelling and automation

When teams reach the final stage of the Application Resilience Maturity Model, they are formalizing their architecture and runbooks, applying tools that allow architectural patterns and recovery patterns to be re-used. This also includes the automation and testing of applications, with full awareness and accessibility by all stakeholders.

The pinnacle of application resilience involves tools that transform the runbook from a Word document to a model in source control, extending techniques such as infrastructure-as-code. The best performing tools seek to model logical application components so that recovery processes are easily visualized, while integrating the breadth of technologies introduced in stage 4 to provide consistency. This means availability and recovery are optimized and strategies are not only reused, they are improved, automatically tested and rolled out. As a resulting effect, all stakeholders have a clear view of what is running and where, not only in peacetime but also in the (much-less-likely) event of an incident.

Wide Area failover needs to be coordinated across three areas, a blueprint helps to undertake this in a more principled way, recording each step. Once blueprinted you can test the resilience of these patterns.

Resilience Maturity Model

It helps to understand where we are so we can make improvements. To help we have outlined this Resilience Maturity Model:

	0	1	2	3	4
	Silos	**Architecture Review Board (ARB)**	**Site Reliability Engineering (SRE)**	**Modern architectures services and platforms**	**Application resilience tooling for modelling and automation**
Summary	Each application team determines their own technology stack and operations plan including resilience with minimal oversight.	As (0), but every application that moves to or is in production must be reviewed periodically by the ARB.	A dedicated team assesses resilience across the organization, working with app teams on an ongoing basis to design and improve solutions.	Clouds, Kubernetes, PaaS platforms, and new architectures like micro-services and serverless can simplify and improve many aspects of resilience.	A standardized service gives a uniform high-level approach to resilience across the organization
Problems	Frequent outages and inefficient recovery for many apps Very wide variability in resilience technology and approaches Frequent reliance on app experts for recovery Very limited reuse and shared evolution of resilience solutions	Inefficient recovery for many apps Wide variability in resilience technology Reliance on app experts for recovery Very limited reuse and shared evolution of resilience solutions	Reuse & evolution of resilience solutions varies by app team but is manual & time-consuming SRE teams may stifle innovation & adoption of new tech, especially for complex apps Limited automation of recovery except for very common apps (as automation is difficult from first principles)	Not all applications are suitable for these standardized platforms Running these platforms can be difficult Specialized expertise is required for failures (although they are less frequent) Applications that have external dependencies need to cross platforms/clouds/services require special attention	Difficult to design well — so consider using Cloudsoft AMP Can take time to gain traction across an organization (start small) Necessary to integrate with existing systems
Next steps	Establish a review board (level 1), SRE team (level 2), and/or common resilience tooling (level 4).	Establish a permanent function to work with app and op teams to assess and assist with resilience (level 2), encourage platforms which provide greater resilience (level 3), and/or invest in common resilience tooling (level 4).	The SRE team should be pro-active in promoting platforms which provide greater resilience (3) and established resilience tooling (4).	Look to combine established resilience tooling to complement the limitations of these approaches and develop consistent resilience across these approaches and outlier apps (level 4).	Encourage cross-team collaboration so application developers recognize resilience as an exciting engineering problem, not a documentation chore or meeting discussion, and so they build more business-oriented metrics into blueprints and move beyond automated resilience to automated optimization based on business priorities

Experience at each Level

Do you recognise your own organisational experience?

Experience

0

With each team having responsibility for resilience, some apps have over time become very reliable, but many are not, depending on the skills and bandwidth of the team. Technology for running applications and recovering applications is extremely heterogeneous, making it difficult for operations teams to be experts in the often very different recovery processes. This can have a partial upside in forcing some DevOps by engaging the development team in operational problems, but failures can be frequent, recovery often slow, and a reliance on implicit knowledge forms which breaks down as the teams change. Operational plans are usually written down, but with wide variance in approach and style and quality, contributing to inefficient recovery. There is not shared learning or a process for resilience reuse. Things are flaky and fire-fighting is normal on all but a few core apps (if you're lucky!).

1

Having a broad review function starts to encourage some consistency, especially where the requirements for approval are clear. Typically these requirements will include alerts on failures, documented runbooks, and scheduled manual fire-drills. Ideally these are regularly refreshed and applications in production re-reviewed. When building new applications, architects and engineers will sometimes look at those previously approved, and reuse architectures and runbooks. However many will not, as it may be considered easier to write a new document than to refactor an existing, so there is still a lot of heterogeneity. Mandating specific technologies can reduce this "sprawl", but without good processes it can also stifle innovation and improvement. Improving resilience plans is still the responsibility of each app team, and is a time-consuming activity and difficult to share with other teams. There is not normally a culture or process of investing in continual improvement for resilience, so everything tends to be "just good enough" with not infrequent panics that require the app experts to be raised.

2

Resilience maturity starts when there is a dedicated function which amasses expertise in good resilience design and shares it with teams. Often this is called the "Site Reliability Engineering" team although its remit is often broader than "sites". This function may act as the ARB (1) or, usually better, have members on the ARB along with other stakeholders. However most of this group's time should be spent working with app teams, both during development and after release, to apply good and standard practices where appropriate. This function is often able to start to develop tools and paper-patterns that app teams can use, and they are able to feed back to app teams when improvements have been identified. Some applications do not fit common patterns and can remain outliers, but for all apps including these outliers an SRE function is able to perform fire-drills more often. This may be done manually, following runbooks, or with some custom-built automation, but crucially it is usually done without direct involvement of the app teams so the quality of the resilience plan is much greater. This leads to more efficient recovery and increased availability, and panics start to be less frequent.

3

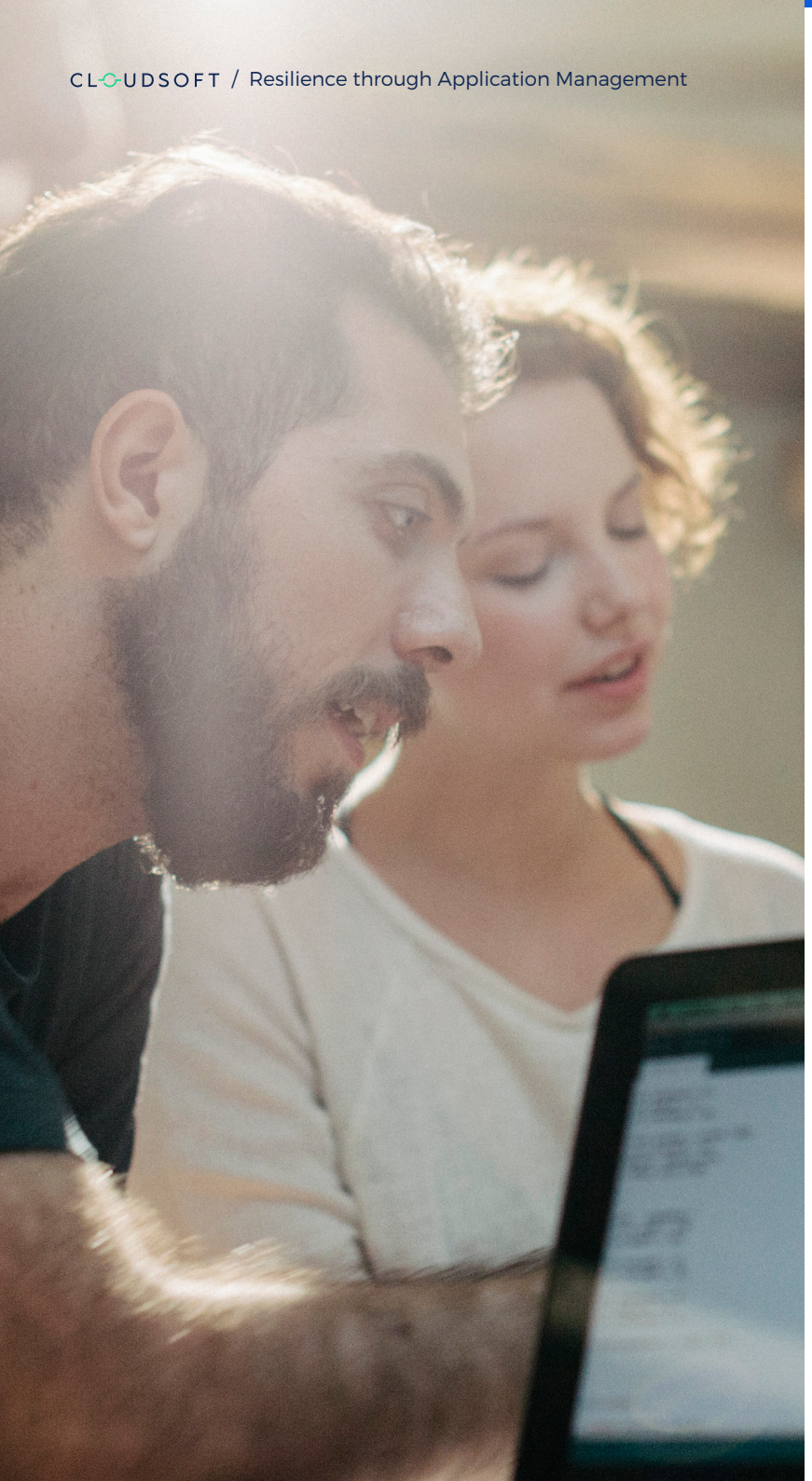
Standardized modern approaches — using containers, cloud, serverless, PaaS platforms, micro-services — are widely used, often promoted by the shared SRE function (2). Applications are being written or modernized to move into these platforms to provide increased and standardised resilience. As this space is fairly new, there are many anti-patterns — expecting any one to be a silver bullet (many will be needed, and they will keep evolving), expecting them to be resilient automatically (they make apps resilient, if done right, and that takes work, and there's a lot of work to run these platforms themselves in a resilient way). Nevertheless this has to be a part of a modern resilience strategy. Organizations should accept they won't work for everything, and if the SRE function (2) isn't involved in their curation you're back to step (0) or (1). For apps that fit with these approaches, availability and recovery are often much improved, although for complex apps it can still be a lot of work to develop applications and the resilience plans. Where there are problems however, it is often now harder in production to correlate failures up and down the now-more-complicated stack and to identify appropriate remediations, especially as organizations will over time need several.

4

The SRE team (2) has put in place a service that gives real-time visibility of applications and can perform resilience automation itself as well delegate where appropriate to other platforms (eg (3) and report on their status. Application teams can access common architecture and resilience patterns understood by this service and can use them in dev/test. The architecture and runbooks discussed previously are formalized into these re-usable patterns, which are tested and evolved on an ongoing basis.

This is the pinnacle of DevOps and application resilience, where tools transform on-paper runbooks and design documents into executable models under source control, extending techniques such as infrastructure-as-code.

The best of these tools seek to model logical application components, so that recovery processes are easily visualized. It is also crucial that this tooling can be integrated with the breadth of technologies used in the organization, including (3), to leverage their strengths and provide consistency across all of them. Through this service, all stakeholders have a clear view of the information they need — from high level details of what is running, compliance, resilience plans, and issues, to low level details of where they are running and what they are doing — available in peacetime and in the (much-less-likely) event of an incident.



Conclusion

A successful resilience strategy can only be achieved by acknowledging that it is a difficult problem and understanding what makes it difficult. With this in place, organizations can appreciate what each of many different stakeholder groups need to bring to the solutions, and put in place the mechanisms that allow each of these groups to get what they need over time to sustain and evolve the solutions.

This eBook identifies four main ingredients, laid out as a maturity model for application resilience where companies ordinarily progress sequentially through the levels. However by being aware of the levels, and by being aware of the application-focussed approach and consistent application management tooling, companies can quickly and safely incorporate all the ingredients and move rapidly to level 4.

Cloudsoft are the company behind Cloudsoft AMP, the leading Application Management Platform, and the only software which gives both composable design-time models and consistent run-time models for any application in any environment. To learn more about how Cloudsoft AMP can turbo-charge your resilience strategy and get your teams to level 4 resilience, contact Cloudsoft today.

Speak to an expert or Learn more about AMP

Further reading

Discussion Paper: Building the UK financial sector's operational resilience

Bank of England DP01/18
Prudential Regulation Authority (PRA) DP01/18
Financial Conduct Authority (FCA) DP18/04

Operational resilience in financial services: Seizing business opportunities

KPMG, June 2019

Operational resilience in financial services: Time to Act

PwC, June 2019

HC 224: IT failures in the Financial Services Sector, Second Report of Session 2019–20

House of Commons Treasury Committee

Cyber and Technology Resilience: Themes from cross-sector survey 2017/2018

FCA

TSB Review: An Independent Review Following TSB's Migration onto a New IT Platform in April 2018

Slaughter and May, October 2019

Time to flourish: A practical guide to enhancing operational resilience in the UK financial services sector

Deloitte, 2019

Operational Resilience is Financial Resilience

Accenture, 2019

Final Report: EBA Guidelines on ICT and security risk management

EBA/GL/2019/04
European Banking Authority, 29 November 2019

© 2021 Cloudsoft Corporation Limited. All Rights Reserved. Registered Company No. SC349230.