SAMPLE

Building Tomorrow's DevOps Engineers

The Recovery's Keeper in the Machine Era

Aiman JoeHanz

Free sample: Introduction & Chapter 1

putrabooks.github.io

Buy the full book:

Carousell carousell.sg/p/building-tomorrow-s-devops-engineers-ebook-pdf-1471385169

Ko-fi

Payhip

Contents

Introduction: The Way Back

Chapter 1 — The Recovery

Chapter 2 — The Territories

Chapter 3 — The Way In

Chapter 4 — The Working Day

Chapter 5 — The Recovery

Chapter 6 — The Pager

Chapter 7 — The Conversation

Chapter 8 — The Instruments

Chapter 9 — The Year Whole

Chapter 10 — The Long Path

Introduction: The Way Back

Every system that changes will, at some point, be in a state nobody intended. The question that separates a mature operation from an improvised one is not whether that happens. It is how long it takes to get out, and whether anybody knows how.

This book calls the thing worth keeping the recovery: the demonstrated ability to return a system to a known good state, quickly, under pressure, without the person who built it. It is not the same as reliability, although the two are confused constantly. Reliability is a property of the system in normal operation. Recovery is a property of the organisation when normal operation has stopped, and it is measured in minutes and in confidence rather than in uptime percentages.

The confusion matters because the two are bought with different things. Reliability is bought with design, redundancy and capacity. Recovery is bought with short ways back, small blast radii, honest detection, and people who have practised. An organisation that invests only in the first will describe itself as highly available right up until the moment it needs the second, and the moment is always sudden.

This book argues for three commitments.

Make one path in, and a short way back. Any change that can reach production by five routes has five sets of permissions, five sets of audit gaps and five chances that the rollback procedure will not match what was deployed. The discipline is a single path, used by everybody including the people who could bypass it, with a return journey that is as well built as the outbound one. Speed follows from this rather than competing with it, because a path everybody trusts is a path nobody needs to negotiate.

Assume the change will break something, and design for the break to be small. Nobody can predict which change will cause an incident, and everybody can constrain how much one change can damage. Small releases, staged rollouts, feature flags, isolated failure domains. The purpose is not caution for its own sake. It is to make the recovery cheap, because recovering from a small break is a routine operation and recovering from a large one is a crisis with a project attached.

Measure how fast you come back, not how rarely you fall. An organisation that tracks the absence of incidents learns nothing, because incidents are rare and the signal is in the tail. An organisation that tracks time to detect, time to

mitigate and time to resolve has a number it can improve every month, and the improvement compounds quietly in a way that absence never does.

The chapters ahead set out the settings this work happens in, the roads people travel into it, an uneventful day and an uneventful year, the craft of recovery itself, the particular wear of being paged, the conversations the role cannot dodge, the instruments it rests on, and the long haul. They are written for an engineer in the third or sixth year who can operate a system and has begun to suspect that the interesting part is not the deployment but what happens afterwards.

It is. The deployment is an event. The recovery is the capability.

Chapter 1 — The Recovery

What Recovery Is

Recovery is the capacity to return a system to a known good state, and it has four components that are usually discussed separately and should not be.

The first is a state to return to. Something is deployed and its configuration is recorded, so that "the previous version" is a specific artefact rather than a memory. An organisation that cannot name the exact state it was in an hour ago does not have a recovery capability; it has a hope.

The second is a way back. A procedure that has been performed before, by somebody who is not the author, in conditions where the system is degraded and the people are frightened. Rollback written during an incident is a plan being invented under the worst possible conditions.

The third is knowing that you need it. Detection, which is the least glamorous part and the most frequently under-built. A failure discovered by a customer is a failure discovered late, and the delay is charged to the recovery even though it belongs to the monitoring.

The fourth is knowing when you have arrived. A definition of healthy that can be checked, rather than a general sense that things look better. Without it, teams oscillate: roll forward, roll back, roll forward, each step taken on impression and each step adding confusion to a system that is already confused.

Two Folklores

Two stories about this work circulate, and both produce organisations that are confident and unsafe.

The gatekeeper holds that safety comes from control. Changes are reviewed by a committee, releases are permitted in windows, and the approval step is treated as the safety mechanism. In practice the approval is a signature by somebody with less information than the person who wrote the change, and its main effect is to make changes infrequent and therefore large. Large changes are the ones that produce long recoveries, so the folklore produces the outcome it exists to prevent.

The hero holds that value comes from intervention. The engineer who finds the problem at three in the morning, follows the symptom through six systems and restores service by hand is admired, promoted and remembered. The admiration is genuine and deserved, and the system that requires it is a

system whose recovery depends on a particular person being awake. Heroism is what happens when recoverability was never built, and celebrating it is how an organisation tells itself the problem is a virtue.

The version that survives both is dull. Changes are small, the path is one path, the way back is rehearsed, detection is honest, and the ordinary week contains no stories. An organisation with no incident stories is not lucky. It has built the thing that makes stories unnecessary, and it will describe itself in terms that sound like an absence of ambition.

Three Commitments

One path in, a short way back. The single path is not bureaucracy; it is the precondition for knowing what is running. Five routes to production means five rollback procedures of which three are untested, and an incident in which nobody can say which one applies. The discipline is to make the one path fast enough that bypassing it is not worth the effort, and then to treat bypassing it as an incident in its own right.

Small breaks by design. Blast radius is a design decision made long before any incident. A release that touches one service can be reverted in minutes; one that touches twelve cannot. Staged rollout, feature flags, isolated failure domains and reversible data changes are all ways of buying a cheap recovery in advance, and the cost is entirely in engineering attention at the design stage.

Measure the return, not the fall. Time to detect, time to mitigate, time to resolve. Those three numbers are improvable every month and they describe the capability precisely. Incident counts are a measure of how much the organisation changes, which is not a thing to be minimised.

Where the Difficulty Sits

None of the commitments is technically advanced, and the work is hard for reasons that are organisational.

The first is that recovery is invisible when it works. A rollback that took ninety seconds and was never mentioned by anybody is the trade's characteristic success, and it leaves no artefact, no ticket and no slide. An organisation therefore funds what it can see: new features, new platforms, new dashboards, each of which is demonstrable.

The second is that the investment is anticipatory. Nobody can prove that the

rollback would have taken four hours without the rehearsal, in the same way nobody can prove a prevented error. The argument must be made on the basis of the incidents that have happened, which means the case is strongest immediately after a bad one and forgotten within a quarter.

The third is that the incentive runs the other way for individuals. An engineer who spends a month making the deployment path reversible produces no visible feature and is reviewed against people who shipped things. The engineer who saves the system at three in the morning is thanked in public. Both signals point away from the work that prevents the second situation.

The fourth is that the skills are distributed awkwardly. The people who understand the system deeply are the people who built it, and they are the least available during an incident and the most likely to leave. Recovery must be built for the person who did not build it, which is a design constraint that feels like an insult to the person who did.

The Thing That Is Actually Kept

What this trade keeps is not uptime. Uptime is a property of a system and it is bought with money and architecture.

What is kept is the organisation's ability to be wrong safely: to make a change, discover it was wrong, and be back where it started before anybody outside the team noticed. That ability is what allows an organisation to change quickly, and it is the only honest answer to the question of how a team can move fast without taking unacceptable risk. Speed and safety are not in tension when the way back is short; they are in tension when it is not.

The capability lives in the path, the rehearsals, the runbooks, the detection, and the people who have done it before. No vendor sells it ready-made, and a reorganisation kills it unless somebody committed it to writing. The chapters that follow are about building it, defending it, and handing it on.

The Inventory Nobody Has

Ask an engineering organisation which of its services can be rebuilt from scratch tonight and the room goes quiet. Ask it to name, service by service, who could restore each one at three in the morning without help, and the silence lasts longer. Neither question appears on any dashboard, and both determine what happens on the worst night of the year.

An inventory of recoverability is not a list of services. It is a list of services with

the recovery route written beside each one, the time that route is expected to take, the person who has actually walked it, and the date they walked it last. Four fields, none of them difficult to produce, and almost no organisation holds all four for anything but its newest systems.

The reason is structural rather than careless. An inventory is created once and decays immediately: every deploy can invalidate the route, every departure can remove the walker, every dependency upgrade can change the time. A document that decays in a week looks worthless, so it is not maintained, so nobody trusts it, so it is not consulted during an incident, which confirms that it was worthless.

There is a version of the artefact that survives. It is not written by hand but generated from the system's own instruments, so it cannot drift without the drift becoming visible. Where that is impractical, the trade falls back on the rehearsal: a recovery is walked on a schedule by somebody who did not build the service, and the outcome is recorded. The rehearsal is more expensive than the document and the only version that stays true.

In the year described in this book, the inventory is the artefact that starts everything. It is January, the practitioner has just taken over the reliability function for a set of services she did not build, and the exercise takes three weeks that nobody wanted to fund. What it produces is not a document but a list of fears, ranked by evidence.

The Three Commitments

The definition offered in this book has three commitments, and each of them is a position that somebody reasonable can dispute.

The first is that there should be one path into production rather than several. The dispute is legitimate: multiple paths let a team choose one suited to its risk, and a single path imposes a standard on cases it fits badly. The answer is that every additional path must itself be recoverable, documented, and rehearsed, and the cost of that is rarely counted. Organisations with four deployment routes have four recovery procedures, three of which are untested.

The second is that every change must be reversible, and where it cannot be, the irreversibility must be named in advance. The dispute is that reversibility is expensive, and the expense is real. The answer is that the expense is incurred once and the absence is paid on every incident, and the two are

never compared because they appear in different budgets at different times.

The third is that the measure of the trade's work is the time to return rather than the rarity of failure. The dispute is that rarity is what customers experience, and a system that fails twice for ten minutes is better than one that fails once for two hours. The answer is that rarity is a description of the past and return time is a capability, and the only one of the two that transfers to the failure nobody predicted.

The three commitments hold together because they are all versions of the same idea. A system should be changeable without being fragile, and the test of that property is what happens when the change goes wrong. Everything in the following chapters is an elaboration of that single position, applied to different parts of the work.

Where the Difficulty Actually Sits

The difficulty in this territory is not the technology, and practitioners who believe it is will spend a decade improving the wrong thing.

The technology is genuinely complicated and it is also, for the most part, well documented, learnable, and forgiving. A misconfigured system throws an error. A wrong command produces a message. The learning loop is fast and the feedback is honest, which is why the technical part of this trade is acquirable by anybody with the patience for it.

The difficulty sits in three places that are not technical. The first is judgement under uncertainty: deciding what to do when the information is insufficient and the clock is running. That capacity is built from experience and it cannot be shortcut, though it can be accelerated by working near somebody who has it.

The second is persuasion. Every improvement in this territory costs somebody else something, and the practitioner's job is to make the trade legible to people whose incentives differ. This is a communication skill and it is the one most often described as politics by people who have not acquired it.

The third is stamina. The work is preventive, its successes are invisible, and it is conducted in an environment that rewards the visible. Practitioners who cannot sustain effort without recognition do not last, and the trade loses a great deal of talent to that specific attrition.

The consequence of misreading where the difficulty sits is a familiar career shape: rapid technical growth, then a plateau at the point where the remaining

problems are not technical. The plateau is often misdiagnosed as a lack of seniority, and the remedy sought is a title rather than a skill.

The Word That Means Four Things

The vocabulary of this trade is unusually bad, and the badness is a practical obstacle rather than an aesthetic complaint.

The central word means at least four things depending on who is speaking. It means the tools, to a vendor. It means the team, to a recruiter. It means a philosophy of collaboration, to a conference talk. It means the practice of operating systems in production, to most practitioners. Arguments that appear to be about substance are frequently arguments between two people using the same word for different objects, and they can run for years without resolution.

The second bad word is culture, which is used to describe everything from a set of shared habits to a management style to an excuse for not changing a process. When a practitioner says an organisation's culture will not permit a change, they usually mean something specific and testable: that a particular person will refuse, or that a particular incentive rewards the old behaviour. Naming the specific thing is the difference between a diagnosis and a complaint.

The third is ownership, which in this trade is a real technical concept with an imprecise everyday use. Ownership means a named person who is accountable for a service's behaviour, who can authorise changes to it, and who carries the consequence. Organisations often use the word to mean that a team's name appears in a repository, which is a materially weaker claim.

The remedy is not to fix the vocabulary, which is beyond any practitioner's reach. It is to be precise in the sentences that matter. A practitioner who says "this service has no named owner and no tested recovery route" has made a statement that can be acted on, and it took four more words than the version that would have been understood as an opinion.

The Machine in the Recovery

Automation has changed this trade more in the recovery than in the delivery, and the change is not the one usually described.

The usual description is that automation makes recovery faster. That is true and it is the least interesting part. The more significant change is that

automation makes recovery repeatable, and repeatability is what converts a capability held by a person into a capability held by an organisation. A recovery script that runs identically at noon and at four in the morning is not merely convenient; it is the mechanism by which the third person problem is solved.

The second change is that automation moves the moment of judgement. Where a recovery is scripted, the judgement happens when the script is written, by somebody with time and a full picture. Where it is improvised, the judgement happens at four in the morning by somebody with neither. Moving the decision to the calmer moment is a large improvement and it is invisible in the outcome, because both paths produce a restored system on the good days.

The third change is a new failure mode that the trade has been slow to internalise. An automated recovery is a program, and programs have bugs, and the bugs are exercised precisely when the system is already unwell. A rollback script that has never run against a real failure is a hypothesis about the failure, and it is the kind of hypothesis that fails in the direction of making things worse. The discipline that addresses this is the same as the discipline that addresses any code: it is tested, and its tests are run against deliberate failures rather than against healthy systems.

The fourth change is subtle and worth stating. Automation makes the recovery cheaper, and a cheaper recovery changes the calculus of risk. Organisations that can return quickly tolerate more change, which is the benefit, and they also tolerate more careless change, which is the cost. The trade's answer is not to slow the change down but to keep the recovery honest, which means rehearsing it and measuring it rather than assuming that because it exists it works.

What a Customer Buys

A customer does not buy reliability, and understanding what they do buy changes how the work is prioritised.

What they buy is a promise about an outcome they care about, delivered with enough consistency to plan around. The consistency is the part the trade supplies, and its value depends entirely on what the customer was told. A promise that is modest and kept is worth more than a promise that is ambitious and mostly kept, and the difference is not arithmetic: a customer who plans around a modest promise and is never surprised will stay, and a

customer who plans around an ambitious promise and is surprised once will reconsider.

The second thing they buy is predictability of the failure. A system that fails in a known way, with a known recovery, at a known cost, can be planned around. A system that fails in novel ways cannot, even if it fails less often. This is why organisations that publish their incident history and their recovery times sometimes retain customers better than organisations with a better record and no communication.

The third is the absence of surprise, which is distinct from the absence of failure. A change that alters behaviour without notice is experienced as a fault even when it is an improvement. The trade underrates this continuously, because the practitioner sees the change as a fix and the customer sees a difference they did not ask for.

The fourth is the truth on the day. During an incident, the customer is buying information: what is happening, what it means for them, and when it will change. A practitioner who has built the machinery to answer those three questions has built something that is worth more than a marginal improvement in the failure rate, and it is usually cheaper.

The Two Folklores

Two stories about this work circulate widely, and each is a partial truth that becomes false when it is used as a complete account.

The first is the story of the gatekeeper. In this account the system is safe because a person stands at the boundary and refuses what should not pass. The gate is the control, the refusal is the value, and reliability is produced by preventing bad changes from reaching production. The story tempts because it reads clearly: the gate has a name, the refusals have records, and the protection is visible to whoever asks.

The story fails because a gate produces a queue, and a queue produces pressure, and pressure produces a large release that passes because it is too big to refuse. The gatekeeper who has blocked eleven small changes will approve the twelfth because the twelfth is now the only way to make progress, and the twelfth carries all the risk that the eleven did not. The control has produced the concentration it was meant to prevent.

The second is the story of the hero. In this account the system is safe because somebody is capable of restoring it under any conditions. The skill is the value,

the recovery is the achievement, and reliability is produced by having the right person available. This story is also legible and it is more flattering, which is why it survives even when it is expensive.

It fails because it makes recoverability personal. A system restored by a person is a system that cannot be restored by anybody else, and the organisation's capability is one resignation away from being a liability. The hero is also, in the aggregate, the reason the underlying faults are never fixed: the faults are being absorbed, competently, every night.

The account that survives is duller than either. A system is recoverable when the path is one, the way back is short and rehearsed, and the break is small enough to be found. None of that requires a gate or a hero. All of it requires the willingness to spend money on outcomes that will not be visible, which is the actual difficulty and the reason both folklores persist.

ABOUT THE AUTHOR

The DevOps volume of the Building Tomorrow's People series belongs to the keeper of recovery, the person whose work is judged entirely by what happens after something breaks. Aiman JoeHanz follows the deploy, the rollback, the alert that pages the wrong team, the postmortem written to be survivable, the error budget that settles an argument between two departments, and the automation that removed a human check and called it progress.