O'REILLY®

A Practical Guide to Cloud Migration

Essays on Organizational Transformation from Google Practitioners

Kieran Broadfoot

REPORT

A Practical Guide to Cloud Migration

Essays on Organizational Transformation from Google Practitioners

Kieran Broadfoot

Beijing · Boston · Farnham · Sebastopol · Tokyo O'REILLY®

A Practical Guide to Cloud Migration
by Kieran Broadfoot

Printed in the United States of America.

Published by O’Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O’Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: John Devins
Development Editor: Virginia Wilson
Production Editor: Caitlin Ghegan
Copyeditor: Sharon Wilkey

Proofreader: Shannon Turlington
Interior Designer: David Futato
Cover Designer: Susan Thompson
Illustrator: Kate Dullea

February 2021: First Edition

Revision History for the First Edition
2021-02-28: First Release

The views expressed in this work are those of the authors, and do not represent the publisher’s views. While the publisher and the authors have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the authors disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

This work is part of a collaboration between O’Reilly and Google. See our statement of editorial independence.

978-1-492-09517-0

[LSI]

Table of Contents

Foreword

Urs Hölzle,
SVP Technical Infrastructure

Cloud computing has long powered Google, bringing the scale of our systems first to internal users and now to developers everywhere. This transformative history dates back to the very beginning (I joined Google in 1999). It was a real challenge from the start—one that could not be solved in a traditional way—so we took some risks and invented things along the way. Without change, and risk, we wouldn't have been able to create an infrastructure that now powers Search, Maps, YouTube, and Google Cloud.

In the early days of Google Cloud, we focused on lift-and-shift of our customers' technology stacks. Most envisioned this as the most practical approach, involving minimal time commitment, code refactoring, and business disruption. Many years and many migrations later, we recognize that this vision of minimal work is rarely realized. Change is an inevitable part of large-scale digital transformation, and no single path works for everyone.

But there is one element we know to be true for every company. *The story of digital transformation is a human one—one that involves as much cultural transformation as technological transformation.* It is this realization that has exposed deeper factors behind successful transformation and brought about the creation of this collection of essays.

Most cloud journeys aren't simple ones accomplished overnight. They require teams dedicated to long-term change and a willingness to learn from failures along the way. Fortunately, many organizations already possess the qualities needed to see through their own transformation. Aspiring teams—engineers, software developers, and business leaders alike—uphold a common goal to help solve real problems and drive the business forward. By embracing this mindset, your organization can persevere through significant digital change and emerge in a better place than where it started.

Throughout this report, we reflect upon the observations of enterprises that have already made the jump and provide specific opportunities to help accelerate your own journey by learning from their experiences. Not all experiences we present here will work for you, but we hope that many of the essays in this report will help you optimize your own transformation journey.

No matter where you are at, it's important to stay the course and commit to a future of growth and differentiation. Large-scale transformation represents a significant undertaking for any organization, but it's one that has been proven to offer substantial rewards. With the right framing and culture, as well as an understanding of the key technical and cultural opportunities, I'm confident that this kind of change is within your organization's reach, too.

Introduction

Kieran Broadfoot, SRE Director

> *"It is not the strongest of the species that survives, nor the most intelligent, but the one most responsive to change."*
>
> —Charles Darwin

It is highly likely you've been regaled with the benefits of "digital transformation." You've read with wonder how companies have fundamentally changed how they work to unlock new business opportunities. It all sounds so perfect. Therefore, you'll be pleased to hear that the material contained herein is not intended to preach that same idealistic message. That would be extremely dull. Rather, we presume you've reached the point of decisive action and need practical advice on how to move forward into the cloud.

Why We Wrote This Report

To aid you on this journey, we want to share our own experiences and failures. We do this because, ultimately, we recognize that you have an incredibly daunting opportunity ahead of you.

But why listen to Google? You've likely concluded, correctly, that we were born in the cloud and have never experienced the same challenges as you. Therefore, how could we possibly understand your unique situation? Well, the good news is that many Googlers have sat in the same seat as you or have supported those in a similar situation. We've experienced firsthand the task of transforming an organization without any practical advice for what this might entail. In other words, we've made the mistakes already. While we cannot promise that this material won't avoid further mistakes, we feel strongly we can help you find a way forward that works for your company, in your market, at this very point in time.

What You Will Learn

If you learn only one lesson from the content of this report, it should be this: culture comes first—always, and every time. No amount of wonderful technology will make your cloud journey successful, although it certainly helps! Rather, a concerted effort to understand who you are, why you do things the way you do, and how you can change is a necessary foundation for broader change.

To that end, we've introduced a consistent format for our essays. Each one signposts *who* in the organization should be accountable, *why* the essay drives meaningful change, *how* to practically deliver the change, and the *pitfalls* you should avoid. We've attempted to present the essays in a linear fashion so that each one builds on the last, but you should feel free to jump in, and around, as necessary. Only you will know your current state and where best to invest your time and energy.

The content is not intended to be a fully comprehensive reference for the journey you are embarking on; it is, however, intended to be consumable by anyone in your organization. This reflects our earlier primary observation—cloud transformation is a cultural transformation that requires buy-in from every person in your organization.

Good luck, and know that we're rooting for you. You've got this.

Who Should Read This Report

Large-scale cultural transformation cannot be achieved solely through the strength of character of any one individual, however much you may hope that to be so. In fact, our overriding message in this report is that achieving meaningful, long-term change requires a sense of community and shared values. To underline this message, each of our essays is targeted at one or more of the following groups, as specified in the "Who" section of each essay:

C-Suite Executives (CxOs)
: Those in executive sponsorship roles tasked with enabling a long-term cultural transformation within their organization.

Technical Leads

Individuals with the responsibility to define the technical strategy of the organization, which enables product/program leads to deliver strategic change.

Enabling Team Leads

Anyone responsible for helping teams overcome obstacles to delivering strategic change within the organization (for example, Product, Program, or Design teams).

Engineers (software, SRE, security, DevOps, et al.)

Anyone tasked with practical delivery of technology as an enabler for business transformation.

Business Functions (finance, legal, compliance, human resources, vendor management, continuity, audit, et al.)

Anyone in a position of influence who holds accountability to ensure the business operates safely and within the bounds of the law.

Managers

Anyone in a position of influence with responsibility for embedding new cultural values.

As we, the authors, come from a broad range of roles and backgrounds across Google, we think that the strategies and ideas presented here will benefit almost every reader: if you work in a company at any stage of the transformation to cloud, we invite you to reflect on these practices and lessons learned.

Acknowledgments

The editor and authors would like to recognize the work of numerous individuals who have supported the production of this report.

We are particularly thankful to Urs Hölzle for his ongoing support and opening remarks, to Salim Virji for his inspiration and leadership during a difficult year, and to Jessie Yang for her thoughtful copyediting assistance. We would also like to thank our external reviewers, Jonathan Johnson and Dieter Matzion.

The following Googlers were kind enough to provide significant time, energy, and expertise:

- John Abel
- Haylee Conradi
- Mike Dahlin
- Brian Dorsey
- Sandra Friesen
- Nick Godfrey
- Nathen Harvey
- Emily Kang
- Matt Levine
- Daniel Pettibone
- Ayelet Sachto
- Laura de Vesine
- Kevin Winters

CHAPTER 1

Managing a Successful Transformation

Andrew Gold, Strategic Cloud Engineer

> *Q: How do you eat an elephant?*
> *A: One bite at a time.*

Who

- CxOs
- Technical Leads
- Managers
- Enabling Team Leads
- Engineers

Why

While one can argue that the ultimate goal of digital transformation is to change a "normal" company into a technology company, it is important to recognize that transformation takes place in gradual phases, in accordance with operating pressures and business needs. The immediate goal of transformation should be to produce a happier, more efficient, more effective, and more profitable company. By any reasonable standard, the process should make an organization better. A successful transformation should be good for everyone in the organization, from the CEO on down. Admittedly, a successful cloud migration is unlikely to make the janitor's task of cleaning office cafeterias any easier, but it is safe to say that nothing good happens to employees of unsuccessful companies.

Let's consider some thoughtful guidance for the reflective organization: achieving success is not a given, but there are very real rewards for those willing to earnestly strive toward the goal. The leadership team will be able to demonstrate greater efficiencies and lower costs, thus displaying its competence to shareholders. The members of the technical team have a rare opportunity to refresh their skills and make themselves more marketable. The sales teams will benefit from more competitive offerings, leading to greater market share and the corresponding rewards. Even service personnel, like the janitor mentioned above, will enjoy greater job security and a more positive working environment, as successful organizations offer more opportunities and better benefits to employees at all levels.

In a business technical context, a successful digital transformation and cloud migration should lead to lower fixed costs, a more flexible allocation of resources, faster development, worldwide reach, a safer change control process, and faster time-to-market with objectively better products. What's not to like?

Before an organization begins a migration to cloud, it must have a workable end-to-end strategy for large-scale transformation. For a strategy to be workable, it must be simple, direct, and effective. If the basic strategy cannot be communicated to a roomful of stakeholders in under 10 minutes, something is badly wrong. Sharing the basic strategy with some of the nontechnical teams can also be helpful, as cloud services can have a significant impact on administrative teams such as human relations, vendor management, auditing, and security, to name just a few.

It's a mistake to underestimate the difficulties involved in the successful execution of a digital transformation. While such a transformation is well worth doing, and perhaps even necessary for the survival of the organization, the effort behind it will be very strenuous, and success will demand everything the organization is able to give. Before embarking on a journey of this magnitude, you must be certain that the organizational will to achieve these goals is present.

Beyond will, an organization must understand its business and technical drivers and then focus on excelling at the tasks that further those drivers. Technology can be a fashion industry, with trends moving in and out of vogue, methodologies rising and falling, and the leading edge of today becoming the technical debt of tomorrow. Therefore, it is essential that an organization hold firmly to its core values and competencies and not be distracted by the most current technology fads.

For example, retail organizations must offer desirable products to the public at a competitive price, while still retaining acceptable profit margins. A realistic digital transformation strategy focuses on the evolution of these drivers and effectively communicates the necessity and importance of these changes to everyone on the project. For a rapid transformation to occur, an organization must either want or need to change, and the impetus must be very powerful or else cultural inertia will halt or severely degrade any progress.

Transformation is not an all-or-nothing process, and there is no requirement that an entire organization must migrate to cloud in order to enjoy significant benefits. While a clear and direct plan is recommended, the first step could be as simple as moving one application into the cloud, performing a postmortem, and then making a more comprehensive plan based on the lessons learned. It may very well be the case that select services can be moved without undue disruption or excessive effort. By limiting the scope of proposed change to well-understood domains, individual thought leaders can have a visible impact on modernizing their working environment and show a technical path forward to leadership. Three or four individuals taking the initiative to demonstrate the benefits of technical modernization can change the direction of an entire engineering organization. If an organization has significant doubts about the viability of a cloud migration, these single-service migrations can be effective and useful first steps toward a more comprehensive effort.

It's important to be realistic in the initial planning stages. A very high percentage of major IT initiatives either fail outright or are rescoped downward so that some form of "victory" can be declared.[1] It is a rare cloud migration project indeed in which the principals are not under extreme pressure by the final release phases. Wishful thinking and undue optimism have no place in developing a migration strategy, as major structural transformations are very demanding and require the utmost exertion from the leadership and

[1] See " Pitfalls " on page 11 for more details.

implementation teams. This is true even for the most competent of organizations, let alone for normal ones. For an organization to transition from an existing state to a more desired state, it must understand where it is, where it wants to be, and the processes that convert from one to the other. This understanding should form the basis of an organization's transformation strategy.

The most common migration failure mode is trying to do too much too soon on a tight schedule. Little is understood about the new platform, so the organization experiences failure and then gives up. If it had just started small with strategic, low-hanging fruit, the organization most likely would have learned from modest early victories and then moved on with its transformation. One specific technique in this regard is to begin the transformation with only a single early adopter or first mover. Learn from the experience, and only then begin broader planning.

How

Let's look at the specific teams involved in managing a successful transformation, as well as at considerations for how to approach the transformation.

The Teams

By looking at the teams frequently involved in this transformation, we can get a sense of the organizational landscape. This will also give us insight into each team's motivation, allowing us in turn to fit them into our cloud migration strategy.

The executive team

If the executive team doesn't align on the need to transform, there's usually little point in proceeding, as internal strife will ultimately lead to paralysis, slipped schedules, and increases in cost. If the executive team lacks internal alignment, it won't be able to successfully mediate the inevitable conflicts among the implementation teams. While there is always room in a serious technical endeavor for differing points of view, everyone involved must honestly embrace the ultimate goals without reservation, even if opinions differ as to the most appropriate means of achieving those goals. Alignment, in this sense, does not mean an uncritical or unthinking acceptance of assertions from senior leadership; rather, it represents an honest intellectual comprehension of a thoughtful and methodical policy. This process is not without effort, as it is very important that objections and counterarguments be given due consideration and honestly rebutted. Once leadership has achieved a directional alignment, the next step is to thoroughly and convincingly communicate the new direction to the entire organization; a solid understanding of the strategic hows and whys will help the implementation teams remain closely aligned with the vision of the leadership team.

Finally, cultural transformation and digital upskilling begin with the executive leadership team itself. Middle and junior leadership will emulate the actions of the executive team, and the implementation teams will emulate their immediate superiors, so the executive team should do what it does best: lead by example. If the executive team can transform itself, the battle to transform the remainder of the organization is nearly won.

The platform consulting team

While doing so is entirely optional, many organizations find it helpful to rely on the technical expertise of the platform vendor and their partner consulting services.

In broad strokes, the role of the vendor professional services team is to provide expert guidance in cloud migration and digital transformation. There is a difference here between the sales pitch, which is intended to persuade, and the operational execution that MUST succeed. Beyond the usual sales boilerplate about "trusted advisors" and "partners in success," the primary role of the platform team is to adequately prepare an organization for the upcoming effort, guide the organization through the usual series of obstacles and impediments, and challenge the organization to execute the steps necessary for success. In more concrete terms, the consulting team will delineate the project infrastructure and communicate the specific process improvements necessary for change to the organization's leadership team. The consulting team will also develop a comprehensive program to implement those improvements and do everything within its means to make sure the digital transformation is successful.

Partner consulting teams are typically used to augment the core platform team, which is usually quite small. The vendor partner consulting teams serve two primary functions. The first is helping to train and guide the organization's implementation teams through the initial steps of platform onboarding, and the second is helping to create the initial platform landing zone. In the first phase, consulting teams will perform most of the hands-on "keyboard" work necessary to jump-start the migration process, but they ultimately will transition to more of an advisory and educational role as the implementation teams gain experience and effectiveness.

The client leadership team has every right to expect excellence in both the vendor professional services team and any partner teams brought onto the project. Any concerns or doubts should be raised immediately with the professional services leadership, as it is their responsibility to ensure operational excellence from the entire consulting team. There is no reason to accept second-class professional services talent from any of the major platform vendors.

It is an anti-pattern for the partner teams to act as the primary implementation teams beyond the first stages of the migration process, except for very specific situations in which exceptional domain knowledge is necessary. Why an anti-pattern? Simply put, undue reliance on the partner teams prevents the in-house teams from gaining the knowledge and platform expertise necessary to take ownership of the project as a whole. The danger is that swift progress will be made by the partner teams while they are present, but paralysis and confusion will occur when they leave, because the in-house teams failed to truly take ownership of key project elements.

The internal implementation teams

The implementation teams must learn as much as possible as quickly as possible. This is both easy to say and hard to do, but it is also the simple truth. The process can be accelerated by having individual teams focus on the specific skills relevant to their normal operations, but a certain body of general knowledge is also necessary if the teams are ever to achieve genuine comfort in the new environment. Structured training has an important role, as do the various online self-study tools. It is essential that the teams embrace the new environment with an earnest desire to acquire new knowledge, as there are no substitutes for enthusiasm and motivation.[2]

[2] See Chapter 2 for more on this topic.

There is a significant learning curve here, and the cloud ecosystems can sometimes behave very differently than the data center environments, so having appropriate expectations is important. The best practices of virtual networking, firewalls, DNS forwarding, and DNS peering, as specific examples, differ profoundly from those in a physical data center, so those new to the platform must educate themselves accordingly.

Cloud platforms are essentially delivered as giant boxes of virtual LEGOs, and, as noted on the outside of a LEGOs box, "some assembly is required." Conceptually, the platforms can each be thought of as a set of interconnected software toolkits of varying convenience, complexity, and maturity. This is of necessity, as each organization has its own business and technical priorities, and no other approach would support the level of customization required by large enterprises. Flexibility and simplicity are opposing requirements. The platforms do their best to provide both, but as stated in the fine print of a car ad, "your mileage may vary."

As the teams evolve and become familiar with the new platform and its toolset, they should increase the scope of their responsibilities, first by maintaining the work implemented by the partner teams and then by actively superseding them in new development. Mistakes will inevitably be made, particularly when the work of multiple teams is combined or integrated, but these mistakes should be regarded as learning opportunities and treated accordingly. In the end, the accelerated learning process and the associated technical growth are outright wins for most organizations, and eventually these process improvements will lead to more rapid progress later in the migration. Few things are more satisfying than watching a team that stumbled hesitantly through the configuration of basic cloud services early in the migration gain mastery of the new platform and take control of its own technical destiny.

The individual

In this essay, I focus on driving change in large organizations, but not every organization is large, and it is important to highlight the role of individual technology champions. There is no inherent reason why a cloud migration needs to be guided by external consulting teams, and one can make a very good argument for a gradual migration led by internal teams, the footprint of which expands as the implementation team gains experience and knowledge. This bottom-up approach can also be safer when there is no immediate driving need for change, since the more gradual timeline lowers risk and removes the temptation to choose expediency over principle.

All teams are composed of individuals, and it is sometimes forgotten that team motivation and team leadership evolve from individual motivation and leadership. Every migration effort, large and small, begins as an idea in someone's head. Empowering individuals really means the active encouragement of excellence. We must be willing to accept ideas that are not our own and to look at problems in a new light when presented with new facts, even when those facts are presented by persons of comparatively low status within the organization. Good ideas can and do come from anywhere if our minds are open and we take the time to listen carefully.

The Methods

So far, I've looked at how specific teams are involved in managing a successful transformation, but leadership also plays a key role. Let's examine leadership, course correction, and time management, which form part of the strategy for working with the teams.

Leadership

A true digital transformation begins with a cultural transformation. As an organization's culture changes, the internal processes and underlying technology will change accordingly since technology is ultimately the external manifestation of an established internal culture. Of course, changes in the internal processes of an organization can also have a profound effect on its culture, so it is important that process and culture reflect and reinforce each other.

An aligned and focused leadership team with strong conflict resolution skills can overcome any number of obstacles. When the executive team is able to communicate its vision with clarity and vigor to midlevel management and the implementation teams, almost anything is possible. Add in a touch of empathy, a healthy dose of patience, and a genuine desire to responsibly decentralize authority, and all of the leadership ingredients are in place for a successful cultural transformation.

Monitoring Progress and Course Correction

At any time, a team member should be able to look at the project status and know whether progress is good, bad, or indifferent. If the current situation is unclear, stop and perform some form of gap analysis. When things go poorly, a blame-free retrospective that focuses on process failures rather than individuals will go a long way toward correcting most problems. There is no shame in failure, as everyone makes mistakes, and there is no such thing as perfect knowledge—but repeated failure for the same underlying reason is to be avoided. For reflective individuals, the goal is always to improve the development and migration process, not to painfully repeat avoidable errors.

In the real world, there are rewards for success and penalties for failure. In sports, one can always look at the scoreboard for objective verification of the current status, and that analogy can be very useful in this context. It is essential that any sustained effort of significance has clear, unambiguous indicators of success and failure.

To summarize: demand operational clarity, have meaningful milestones, aggressively track progress, embrace the process of blamelessly correcting failures, and make sure that everyone knows the current score.

Learning, Training, and Continuing Education

In practice, the most important group to receive enablement is the executive team itself, as improvements in its knowledge serve to empower the entire organization. Most learning plans overlook this idea, but it is worthy of some reflection.

Significant organizational changes require a great deal of upskilling and training. For training and enablement to be successful, there must be both an active learning plan and executive sponsorship. The active learning plan is used to address any concerns raised by a skills gap analysis, and the executive sponsorship is used to give authority and energy to the plan itself, as a plan without sponsorship is unlikely to produce much in the way of results. More specifically, the internal teams will be subject to many demands throughout the migration process, as they must keep the existing infrastructure and applications running while simultaneously attempting to both learn and implement on the new platform. Without executive support, learning will occur only in the margins, and infrequently at that. This reduction in training velocity is very likely to impede implementation velocity later in the project, so deferred technical enablement carries a significant cost.

Having a clear training strategy that focuses on self-service will simplify and accelerate the learning process. Different team members have differing levels of motivation and will learn at different rates, so having several options available can be helpful. Training environments such as Qwiklabs can be very useful for honing specific skill sets. A best practice is to make available a sandbox environment of some type in which team members can experiment with new technologies, and without fear of impacting others. This can serve several purposes. The planning and creation of a sandbox environment is an exceedingly valuable learning experience for the infrastructure team, and it can also be very valuable for the application development teams, who can then create experimental applications on the new platform.

Another useful practice is to make use of structured online courses such as those available from Coursera, and yet another can be to have internal "brown bag" lunch sessions over pizza or sandwiches. A sample lunch topic might be the best way to right size virtual machines, or how to tune data queries to avoid excessive costs. Those who actively embrace the new technologies should be empowered to access more advanced training as they become ready to consume it, as this will frequently inspire emulation by the less self-directed. Since learning is a continuous process, ongoing training should be a permanent feature of any successful digital transformation.

Time Management

Aggressively track progress on a weekly or even a daily basis. A sense of urgency is absolutely necessary for large organizations to make visible progress, but realistic timelines require thoughtful insight and a deep understanding of the real requirements of production-ready services. This deep understanding is usually absent at the beginning of a digital transformation, when so few of the structural processes on the new platform are completely internalized. It is inevitable on a project of this scope and nature that the implementation challenges will be much better understood at the end of the project than at the beginning. This is the primary reason why engaging a platform consulting team is helpful, as one of its main jobs is to guide its clients around the more obvious pitfalls during the initial project phases.

Without digging into the details of the various project management methodologies, it can be safely generalized that the greatest challenge to planning and time management throughout an organizational transformation is the very long list of unknowns. Simply put, the leadership and project management teams do not yet know what they do not know. This suggests that any plan with rigid deliverables and a rigid timeline is unlikely to be realistic or successful.[3]

General Dwight D. Eisenhower, a man well known for his planning ability, once observed that "plans are useless, but planning is indispensable." This quote captures the idea that events can render the concrete details of plans obsolete yet also maintains the utility of the planning process and the important information it provides. Any adopted methodology must handle change gracefully, as there will be significant changes throughout the project—very likely many of them. Plan well and thoroughly, but be prepared to adapt your plans as events unfold.

Pitfalls

In this section, we will look at some of the anti-patterns that commonly arise as part of a large organizational transformation, such as migrating to the cloud. The names might not be immediately familiar, although the symptoms may well be.

The Blame Game

When a team member is publicly humiliated for making a mistake, a cascade of negative events takes place, including the following:

- The team member becomes resentful of the public correction. Usually, they try to hide their resentment, but whether or not others perceive it, it still exists.
- The team is incentivized to hide mistakes rather than surfacing them immediately. The mindset becomes, "I won't let this happen to me."

[3] See Chapter 5 for more details on this important topic.

- The team is incentivized to shift responsibility elsewhere. The mindset becomes, “It wasn’t my fault.”
- The team becomes defensive and focuses on not being blamed rather than on achieving excellence. The mindset becomes, “I’ll never try to do more than I’m sure I can complete.”

All of these negative behaviors tend to slow team velocity and reduce progress. One mistake that is specific to Agile is blaming teams or individuals for incomplete sprint stories. Blame in this context might be something as simple as mild public ridicule, but consider the long-term consequences. In effect, the team has been incentivized to plan very conservatively, and to commit only to work it is absolutely certain it can complete within the current sprint. As teams typically relax, if only mentally, once the committed sprint work is done, they have essentially been taught never to exceed expectations. Remember, the actual effect of a custom or practice may be very different from its stated intent. In these cases, it can be helpful to focus on the actual results in order to see which techniques are most effective.

How should these types of problems be solved? All change begins with introspection, so a thorough and honest review of actual team processes and procedures is a good place to start. Many teams “understand” the concept of a blame-free retrospective but still place blame in practice. In the end, the team management techniques that are actually implemented, not the ones that are merely understood, are the ones that have an effect, so bring in external help and support if necessary. Consider creating pilot programs to test new techniques and surface the most helpful ones, as this can be a low-risk way to experiment with process improvements.

Unrealistic Project Tracking and Time Management

The leadership team must have a clear but flexible vision of the project end state, have coherent priorities, make decisions quickly, and communicate effectively. This sounds very simple, but of course the real world does everything possible to impede, confuse, and obscure. Concrete and measurable goals, thoughtful milestones, and a sane timeline, combined with a sound conflict resolution strategy, will enable the leadership team to communicate its vision effectively and win the confidence of the implementation teams. Intelligent risk-taking proportional to the opportunity should be accepted and rewarded. One specific goal should be to carefully track the rate at which the in-house teams take ownership of key project elements, as this will indirectly track the process of training and upskilling. The leadership team should be aggressively proactive without being intrusive and must be ready to quickly resolve the inevitable conflicts and roadblocks.

Since all projects have limited time and resources, thoughtful milestones are necessary to demonstrate progress and to promote a sense of urgency. It is a truism that a job will expand to consume the time allotted to it, and timeboxing can be a very valuable technique in this context, but it must be noted that highly structured plans composed of fixed deliverables and fixed dates are problematic because of the numerous uncertainties. Team energy, morale, and cohesion can be destroyed by the pursuit of goals commonly believed to be irrational or unobtainable. Dates that may have appeared plausible at the beginning of a project will frequently be perceived as unduly optimistic later in the process.

Letting the External Teams Solve the Hard Problems

A comprehensive, in-depth knowledge of platform architecture and individual service implementations is essential for the in-house implementation teams; without it, they will ultimately fail to live up to their potential. This concern is not theoretical and should be taken very seriously by the leadership team.

For example, in a recent project, the offshore partner team was genuinely talented and exceptionally efficient. The client would ask for a feature, and the offshore team would have it complete within a few days, or within a week at most. The client became entirely too comfortable with this situation, which continued for about six months. When the consulting contract reached completion, it was a very sobering experience for the client technical team to realize that its internal staff had very little idea how to operate, maintain, and extend its shiny new system. Ultimately, the client was forced to sign another statement of work with the partner, the intent of which was largely to help the internal team learn to manage its new platform.

Use the external consulting teams to enable the in-house teams, not to do work for them.

We’re Successful, So Why Change Anything?

Every large organization has elements that are strongly resistant to change, and many times the most successful elements are the most resistant. Their thinking is very straightforward: if things are working now and have worked well for the past 10 or 20 years, why make changes? Certainly change for its own sake is counterproductive, but “change” in the current context refers to embracing a more advanced paradigm that bestows measurable, demonstrable advantages. It is imperative that the leadership team communicates the significance of these advantages so the midlevel and implementation teams understand the purpose, goals, and benefits of the new processes. If the midtier teams are doubtful of the organizational commitment, their uncertainty will lead to inertia, indecision, and the indifferent execution of higher-level policies. The leadership team must actively convince the implementation teams of the clarity of its goals and its commitment to a successful transformation. A clearly defined strategy composed of coherent individual elements describing concrete, measurable results is invaluable here.

The Bike Shed Effect

It is easy to become distracted during a migration effort by the myriad of surrounding details. For an organization to effect real change, it must provide useful products and services sooner rather than later. More prescriptively, it means the initial migration effort should restrict scope to a very lean minimum viable product (MVP) and release that MVP, whether internally or externally, as quickly as possible.

Parkinson’s law of triviality argues that people within an organization commonly give disproportionate weight to incidental issues because they understand them more easily than the much more complex challenges of their primary tasks. It is a mistake to dilute the limited available resources through the pursuit of the nonessential or trivial. Decide what is important to the migration effort and do nothing else.

CHAPTER 2

Celebrating (and Tweaking) Your Culture

Adrienne Walcer, Technical Program Manager, Google SRE

Who

- CxOs
- Technical Leads
- Managers
- Enabling Team Leads
- Engineers
- Business Functions

Why

It's possible to achieve a cloud transformation while keeping your momentum and keeping your team on board. To do so, you need to pay careful attention to your company culture.

Let me tell you about Company A. Company A was a start-up that used a proprietary application to generate reports for a niche market based on public datasets. The company had begun as a relatively small outfit, serving in-region partners and growing year by year, but eventually it landed a pretty sizable contract. The ongoing maintenance required a shift away from the infrastructure on which the company had previously been reliant. Oh yes, it was moving into the cloud.

Company A's leadership contracted with an IaaS/PaaS cloud provider and set an aggressive internal mandate: within six months, they'd conclude their migration. The company's application architecture was not optimized to be cloud-deployed, and the development team pushed back. The team was already utterly busy even without making major changes, and it needed time to learn the infrastructure it was shifting to and appropriately redesign the heart of the company's operation. But leadership was firm—the big contract they'd landed had received major attention, and they were anticipating a few more large RFPs headed their way soon. The dev team members had to self-organize, and they managed to do it—taking distributed computing courses in their free time and sprint-designing their way through weekends.

At the end of those six months, more than 75% of the application development team realized that they now had valuable new skills, and they didn't feel like Company A provided a supportive environment. They took their newfound skills and found profitable work elsewhere. Company A had transformed its technological stack without transforming its company culture, and the employees that did the work and made the change weren't given a voice. But it didn't have to happen like that. What if Company A had proactively maintained a positive and supportive culture while undergoing its cloud transformation? With planning, resources, and empathy, it is possible to achieve a cloud transformation while maintaining both your team and your company's momentum. An empowered workplace culture that values psychological safety can supercharge any organization.

How

Let's dive into the basics of cultural dynamics in the workplace. First, we'll build the perspective you need for a positive organizational transformation. Then we'll talk about the preparatory analysis you'll want to do before getting started, both in understanding your employees' perspectives and in understanding how these changes might reverberate through your tech stack. Finally, we'll talk about psychological safety and its role in preventing change-related attrition.

The V-model of a standard systems development life cycle (shown in Figure 2-1) offers a basic picture of how to effectively approach organizational change.

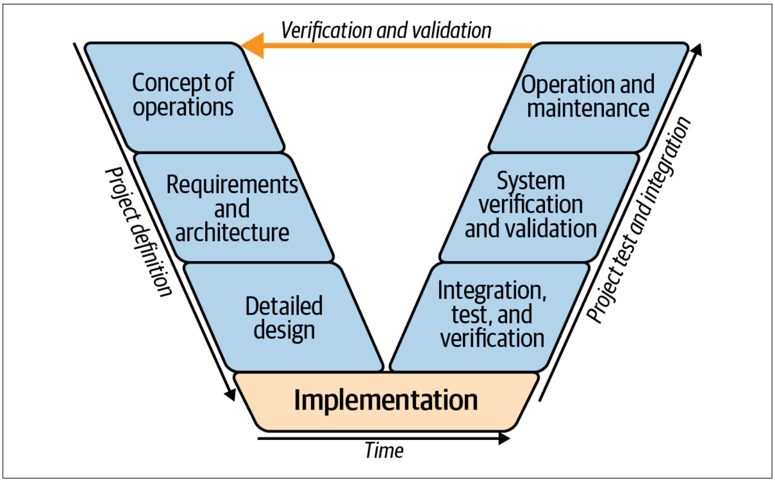
Verification and validation
Concept of operations
Requirements and architecture
Detailed design
Implementation
Integration, test, and verification
System verification and validation
Operation and maintenance
Project definition
Project test and integration
Time

Figure 2-1. The V-model showing a standard software-development life cycle (source: https://oreil.ly/VModel)

Note that there's a lot of investment placed in the planning, requirements gathering, and design phase of this life cycle. You'll need to carefully consider and incorporate the needs of your employees in all phases of the life cycle if you want your company culture to remain positive and collaborative throughout this cloud transformation.

Organizational habits are really hard to break. Like...really hard to break. Within your enterprise, each team has developed patterns, ways of working, and unspoken truces with other teams. These habits developed for a reason—they alleviated some form of stress, protected team members against some kind of burden, or provided some benefit to the team. And breaking these habits is like trying to end a coffee addiction—abrupt withdrawal is difficult. (In Charles Duhigg's *The Power of Habit* [Random House], the author suggests that it takes more than three weeks to break a habit.) A cloud transformation demands that your teams find new ways of working and potentially requires breaking habits that may have been deeply ingrained in your teams for a long time. Asking them to do this cold turkey won't work out well, and your company will feel the strain much more than if you simply took the time to pay attention to your employees' needs.

A part of understanding company culture is understanding your employees' needs—how they're incentivized to behave and the performance criteria they're being asked to uphold. Review how your employees are compensated—how they are rewarded just for maintaining the status quo—and mark that down as your baseline. If you ask your employees to learn new things and to make quality, sweeping changes, then they need to be appropriately incentivized to do so. If you can, find ways to help them buy into your success. A meaningful carrot leading toward the outcome you want (something that more deeply respects your employees' needs than a pizza party) can signal to teams that this is a change that can be accomplished together.

There are two key techniques for driving respectful organizational change:

- You'll want to put together some centralized resources to support your employees through this change. Don't leave this step up to individual teams unless you're comfortable with an end result that has inconsistent implementation and inconsistent documentation. If cloud transformation is important to your organization, then managing and supporting that transformation should be a formal part of one or more persons' duties. (More on this later.)
- The subject matter experts who have built and continue to maintain your stack, services, or products should be fully recognized for the value they generate and treated accordingly. Instead of imposing deadlines on them, utilize a bottom-up approach to estimating the amount of time it will take to make relevant transitions. Encourage component leads to set deadlines that accommodate the needs of that system. You can enforce the deadlines that they established themselves, but the enforcement of deadlines that were set centrally without considering individual component needs won't land as well and might result in disgruntled employees or subpar work products.

In transitioning your business to cloud, you're asking your employees to learn new technologies and apply them. For some, this might mean learning a new tool or new workflow. For others, it means learning a new programming language, or learning new infrastructure. In asking your employees to learn new things and grow, you're placing them in a potentially vulnerable position. Some might doubt their ability to adapt to new things, and some might feel "left behind" in an environment that was previously comfortable.

Consider the five stages of grief that employees might experience during this type of organizational change:

Denial
: "I'm going to ignore those weird 'cloud' people. It's business as usual for me."

Anger
: "How dare you risk our business with this change!"

Bargaining
: "I see what you're doing—I'll shift this piece as long as things mostly stay the same."

Depression
: "This sucks. I don't feel like I'm good at my role anymore."

Acceptance
: "Woah—I get it now! This is pretty cool!"

To create a psychologically safe workplace, be supportive of these changing needs and offer appropriate resources for folks who are up to the challenge of growing with you.

Here is where your company's relationship with empathy becomes important. A diverse and inclusive workforce has a wide variety of needs, from accommodating different learning styles or abilities to accommodating different levels of abledness. One-size-fits-all solutions actually tend to fit...no one. In asking employees to go along with this transformation, you need to be prepared to support them for the duration of the ride. In Chapter 1, we explored the importance of training and continuing education. When considering cultural effects, it's time to ask questions like: are the educational and training materials supporting this transition accessible to all? Are you inadvertently asking employees to invest nonwork hours toward your mission? Are expectations clearly laid out and easy to navigate? Failing to provide appropriately for a diverse (and neurodiverse) employee population is a powerful tool to drive attrition—and while moving to cloud adds value, you don't want to compromise your company's position on the labor market to get there.

What should those resources be? Providing clear expectations and consistent, centralized guidance is key. To do this, we recommend:

- An HR-stamped policy that accommodates learning time during work hours.
- A centralized hub that points to the available documentation discussing approaches, policies, processes, procedures, and design concepts (essentially, a business architecture that's vetted by an accessibility specialist).
- Contact information for the central person or team who is project managing your company's transition, with senior and credible sponsors. It's really helpful for this team/person to offer drop-in "office hours."
- A clear listing of which teams are driving transitions for which components, and their determined transition schedules (system ownership detailing).
- Some joint working groups so that similar systems can share expertise or offer collaborative support for the employees actually doing the work, and to test for end-solution integration. Having a psychologically safe place for discussion without fear of retribution drives collaboration and gives folks a feeling of social acceptance during a potentially stressful time.
- Processes for decision making and escalation resolution.
- Documenting everyone's processes and outcomes, thus empowering your teams to learn from one another.

If you work employee support into every stage of your cloud transition, it's possible to maintain the open information exchange and motivated proactivity that not only sustains a great transformation but also might have overarching positive effects for your company culture as a whole.

Pitfalls

Here we will look at some of the friction you might encounter while driving the cultural aspect of organizational change. By preparing to identify and overcome these pitfalls, you can more smoothly move your organization's culture in the desired direction.

Top-Down Direction, Bottom-Up Crickets

Setting timelines for your team without listening to the team or consulting it for feedback does three things:

1. It guarantees you won't meet your deadlines.
2. Your team becomes sad that it can't meet your deadlines.
3. You've told your team members that their expertise or opinions don't matter that much.

Demonstrate compassionate leadership. While it's compelling to set central deadlines and dictate the pace of change, recognize that this type of top-down leadership fails to acknowledge the people who are accountable for your systems and services. Valuing your employees is central to creating a positive workplace culture. Respect the expertise of your team members and lean on them to provide you with timelines when change can happen. Or if you set a deadline, keep a line of communication open so that your team can keep you in the loop with challenges, objections, or feedback. Give everyone a voice, and lead by example through listening, as Nickolas Means explains in this talk at the Lead Developer conference. You'll be more likely to hit your deadlines, and the members of your team will view their successes at meeting those deadlines as their career wins.

Teams Can Support Themselves

A "desert island" economy is bad for office morale. Teams break off into separate clans, warring for that last bunch of bananas or bit of resource capacity...it gets ugly. You need unity. To create a feeling of camaraderie among your employees, you need to ensure that everyone is hearing the same messaging and getting the same levels of central support. Policies need to be empathetic to employee needs but also must be the same for everyone. If different teams are each creating their own documentation, standards, and training material, it means that the end work product might not be consistent, getting there might involve in-fighting (from inconsistent sponsorship), and different teams might not feel equally valued by the organization. Organize things centrally, and make navigating this change easy for your employees.

Feelings? Meh.

Employees are here to work, and feelings don't matter. If that sounds old school, that's because it is. Modern businesses manage too much information for one person to go it alone. We collaborate. And for collaboration to be productive, all parties need a bit of safety and space to express their preferences, opinions, and needs in order to keep showing up for that working relationship. Change threatens the status quo of these relationships, which can be a stressful experience. If you prioritize employees' psychological safety and well-being during periods of change, it's far more likely they'll still want to be your employees post-change.

CHAPTER 3

Defining "Good" for Your Organization

Andrew Milo, Delivery Executive Director

Who

- CxOs
- Managers

Why

Why define "good" for your organization?

Change is hard. In fact, it's been said that no one likes change but everyone loves progress. For many people in a large organization, a new company initiative may seem as common as their weekly coupon circular, and viewed cynically, such initiatives often result in the same level of personal and organizational improvement. Many of these ideas are great—but ideas alone are not enough to create impact.

So where do things typically go wrong? Often it starts right at the beginning. Dr. Laurence J. Peter, an educator well known for formulating the Peter Principle of management, observed that "if you don't know where you are going, you will probably wind up somewhere else." As a leader, you are looking to create a better overall outcome for your business. The problem is, you can't do it alone, and others are the ones who actually need to do most of the changing.

However, expecting others to change simply so *you* can get a better outcome usually doesn't have the necessary sticking power.

Therefore, the key to success is to ensure that the change you're planning is truly built on bringing progress not only to the business but also to the *people* who are needed to make the change really work.

Said another way, if you engineer a holistically better situation, you can rise above the norm and make this change "stick" with the organization and actually prompt real, meaningful progress toward a better business outcome for everyone. This is a fundamentally better approach than simply using the hammer of authority and policing compliance.

This brings us back to defining "good" for your business. Defining the aforementioned "holistically better situation" is the crux of the matter. This goes beyond vision—defining ”good" gives you a *practical description of all the aspects of your business* that are needed to make that vision a reality.

A final point before getting to *how* all this can be accomplished: there's the basic question of "Why is this change important?" People may not care about your definition of "good"—they may question why defining "good" for your business is even necessary. The answer isn't nuanced at all and is the foundation of what "good" really looks like. We view digital transformation as a "cannot fail" initiative, largely because every organization has a choice: disrupt itself or be disrupted by the market. This sentiment was best articulated by the CEO of a large multinational corporation who said, "I want to grab the hand of disruption so that it doesn't grab me by the throat later." He's right to view it this way. For better or worse, this situation will continue to intensify over the next 5 to 10 years. There is a very clear arms race going on across every industry and market—at its center is the practical application of advanced analytics and machine learning in automating core business capabilities. The firms that augment their humans by democratizing access to these new superpowers fundamentally outperform the organizations that don't. The resulting and ongoing level of market disruption has the potential to significantly change the face of most industries.

You have a choice: will you choose change, or will you allow change to be imposed upon you by the market, by the hand of a more agile competitor?

Commitment from the C-Suite is a must. Ideally, there is a CEO mandate with committed dates; transformation typically won't happen otherwise. Transformation at scale is a multiyear journey that requires consistent grit and organizational willingness to keep the process front and center over an extended period of time. Many transformations fail because of the shortsightedness of leadership and the loss of resolve midway through the process. Often this is a result of too much delayed value that is front-ended by a lot of painful change. The CEO needs to work with the board to find a way of sustaining a multiyear transformation strategy that is sustainable through the natural cycle of changing personnel. Broad buy-in from many C-level executives assists in making sure that things don't go backward several steps when inevitable changes occur. Also, company culture largely reflects the attitude of those at the top—if leadership won't make the change and back it for the long haul, their organization takes note, and inertia wins by default.

Direct reports of the C-Suite (or those in a similar role) are the team that makes or breaks a transformation initiative, since the biggest enemy of such initiatives is organizational inertia. These important people are the ones who steer everything toward *progress* and make sure that the program isn't simply inflicting *change* on the organization. The team is the connective tissue between the strategy defined by the C-Suite and the execution of that strategy by the rest of the organization. They typically define "good" because they have these two things:

- the business context to understand the desired outcome
- the technical context to understand the way things are today

Ideally, they also understand why they are the way that they are.

How

The more people that benefit from this transformational change, the more likely it will succeed. Thus, your ability to define "good" for your organization in a way that makes it easy to understand, measure, and communicate impacts your success dramatically.

Fortunately, there is a well-worn pattern for doing all of those things, though sometimes an organization may lack the structural execution and discipline to take them seriously and do them well.

This is where the C-Suite's commitment and the resolve of their direct reports become critical.

The pattern is fairly straightforward:

1. Define the end goal based on business value. This becomes your North Star vision against which all other decisions are made. Yes, it's that foundational. This is a 5-to-10-year vision of what type of business your organization wants to become—what it *needs* to become—in order to stay on top/compete/survive.[1]
2. Each aspect of the North Star then gets an action plan devised of sprints, which outlines where things are today and the required objectives and resulting milestones needed along the way to get to the desired end state. Even if you don't yet know *how* you'll accomplish each milestone, you have to lay out the road map. At a minimum, milestones often cover people, process, and technology for completeness. Sometimes a fourth aspect of data is added because of the criticality of this capability going forward and the relative immaturity of many organizations to utilize it in an effective manner. Tracking it separately accelerates improvement through increased focus.
3. Each milestone is then defined by a set of Objectives and Key Results (OKRs) that are assigned to the responsible group or groups. Depending on the complexity, key results are often best expressed as a key performance indicator (KPI), which allows you to measure progress toward the North Star vision. In many cases, if it isn't a number, you can't measure it and you won't know if you're making progress.

Simply stated, a value-based vision translates to a state of your organization that is needed to realize the vision. We then build a road map composed of incremental milestones, with each milestone defined by OKRs to make it concrete and real.

[1] This North Star vision is very different from the "good" we are looking to define. The North Star vision is the business benefit you are striving for—our task is to define what that actually means in terms of what you need to do to realize these benefits.

Defining the End Goal

There are many versions of this intent, which you may call strategic imperatives, the North Star, or even business differentiators. While the names differ, the intention is the same: an outline of what needs to be true for your business to achieve a stretch-goal market position. Hopefully, you already have your end goal identified, but if not, ask yourself the following questions:

- What goal would you set if you knew you couldn't fail?
- To flip it around and look at things from the other side: what's holding you back the most in your business? What goal would you set if this thing that is holding you back was simply gone? What do you want to accomplish?
- What differentiates you from your competitors? Will that hold true in the future, or does it need to change? Where do your competitors have the upper hand, and why?

These are just a few of the ways to begin thinking about a North Star vision. While these are likely specific to your business, some common inclusions touch on:

- Business agility
- User value
- Product velocity
- Data activation, both in the enterprise and commercially
- Service availability/uptime/continuity for service providers
- Innovation/market-defining impact
- Employee well-being and development

Each of these items should be directly tied to business value for the organization. For example, business agility is desired not simply for the sake of agility but rather for the ability to quickly take advantage of new opportunities or handle new and unforeseen threats. Product velocity isn't sought for speed alone but rather provides a shortened feedback loop of market success and a faster return on investment for those initiatives that bear fruit.

Ultimately, defining the North Star is all about outlining what your organization needs to be in order to achieve your business goals.

Creating an Action Plan

When the C-Suite has articulated its North Star items, and there is agreement on them, it is the responsibility of the direct reports to build a supporting action plan. This involves a candid analysis of where things stand today on the one hand and a well-fleshed-out description of the state of the North Star on the other. The state of the North Star, or *North Star state*, is the business state that is required to deliver on the North Star vision. For example, if the North Star calls for increased product velocity, the North Star state will likely cover, among other things, the product development process (often a migration to Agile), as well as the operations process (ideally a migration to SRE or some other DevOps discipline).[2]

Taking these two aspects as an example, each of these components requires a series of sprints in and of itself. How do you guide an organization that uses mostly Waterfall methodology to Agile? How do you break down walls between the product, engineering, change management, and operations teams to create a single, unified, product-driven delivery chain that is focused on user value and positive business outcomes? This doesn't just happen on its own—it requires specific, intentional planning.

Start by marking guideposts along the way from where you are (your existing state) to where you want to be (the North Star state). You can't mark guideposts without the two states fully defined, and you can't get to the end state without finding points of value along the way. The journey from one state to the other is now the focus and crux of our definition of "good."

To properly outline this journey, ideally you drive things from both sides of the spectrum. Where *can* you go next from where you are? What do you have in the line of sight for improvement? Additionally, what is *one step removed* from your North Star state? What would things look like when you are *almost* there? Keep doing this iteratively for each new step, one forward and one back, and eventually you work on meeting in the middle. Outlining these intermediate but distinct states helps make things increasingly clear. Taking the dual-vectored approach does two things—it keeps things real by

[2] If you need inspiration in this specific area, be sure to look at DORA Research's research program, which outlines numerous capabilities for consideration.

driving forward from where you are, but it also keeps your eye on the prize and allows you to mentally "live" in each new stage as you flesh out its various components. This approach isn't required, but it does help orient things along the way, versus only working forward or only working backward.

A caveat: don't be afraid of how many steps you identify to begin with—get it all out and work to simplify after the fact. If you are far from your desired state, there may be several milestones along the way. That's OK. At least you know that now and can prepare those around you for a slightly extended journey. Don't fall into the trap of not wanting to communicate how far apart things are—as long as you make sure that each step brings progress, people will continue to push forward because each new milestone is worth it.

Making Progress

The key to making progress is twofold:

1. Each milestone should provide value for both the business and the people who make the business run.
2. Measure progress with tangible results articulated in numeric KPIs so that people focus on the outcome instead of the activity.

Defining Value

Defining value for the business is largely handled by your North Star vision. Ideally, each milestone is getting the business incrementally closer to that final North Star state across all business attributes. In our prior example of product velocity, one of the tracked points of value might be in the area of product rework efficiency. By driving the percentage of rework development down, the business reduces the cost of shipped software (or of go-live for an internal project). This is a valuable goal indeed.

Now, you may wonder: does this goal also have value for the development team itself? Yes, absolutely, *if* you look beyond the cost. Few developers enjoy revisiting code for the purpose of fixing errors over and over again. Most find it tedious and professionally stifling and would categorize this under *toil*: work that lacks enduring value and is reactive rather than active. As described in the "Eliminating Toil" chapter of *The Site Reliability Workbook* (O'Reilly), edited by Betsy Beyer et al., reducing this toil is a worthwhile goal for software engineers. In fact, the DORA research in this area shows that several improvements in your development pipelines can reduce pain and boredom in many areas. This ultimately means less burnout for your people. Completely eliminating toil may not itself be a practical target, but reducing it wherever possible is a realistic strategy. This definitely means progress for the people who need to enact the change and for the business. They both have skin in the game, as well as something to gain with a well-played hand.

Now you see that reducing rework and tracking the associated metrics (more on that later) is a goal that serves the business as well as the teams that need to change in order to deliver the better outcome. Similar goals are likely found in all areas of your North Star vision. If you are struggling with finding tangible benefit for your people, talk directly with the teams that most need the change. Find out what is blocking *their* improvement and what is causing *them* the most pain—you might be surprised by just how well your goals link with theirs. Often the lack of performance by a specific team is tied to things that are its biggest sources of friction. Remove the friction and the value gets flowing again.

If, after a thorough investigation, you still can't find common ground, then for the longevity of your transformation, adopt a "last resort" balanced plan of half the actions for the business and half the actions for the people involved. The business still benefits in many ways—for one thing, your people won't fight the change you are trying to create (remember, a transformation initiative's biggest enemy is organizational inertia)—and the business indirectly benefits from less employee churn, higher satisfaction, and increased likelihood of innovative approaches discovered by happier, more fulfilled employees.

Initially, you may think that you can't afford this "last resort" type of approach, that budgets are too lean and that business needs are too great. In reality, it is these dire situations that require the biggest support from the surrounding teams, and keeping them engaged, motivated, and actively working with your transformation plan is of utmost importance. If you don't make progress for them as well, you won't have the velocity to keep things moving forward, and each step will be harder than the last.

Measuring Progress

Now that you have a road map that outlines the milestones to get you from where you are (your existing state) to where you want to be (the North Star state), it's time to outline the execution of that plan—that is, how you get from one milestone state to the next.

Using OKRs is a very effective mechanism for guiding your team from one milestone to the next. They are particularly effective for several reasons, but here we are keenly interested in the fact that they strongly tie your business objectives to the work people do every single day.

Up until now, I've largely been talking about developing the objectives portion of an OKR. Developing appropriate *key results* (KRs) is equally important. Depending on the complexity, a key result is often best expressed through the use of a KPI. A simple KR might be "Publish CSAT by Dec. 2nd" or "Ship Project X by Y Date." These are binary outcomes that don't require a KPI. Most other things benefit from being given specific numbers that are easily understood, well-defined, and meaningful. Effective key results express measurable outcomes that, if achieved, directly advance the objective.

This is an area in which many organizations struggle. Some suffer from the lack of discipline to track key results, while others suffer from tracking things at the wrong level. Still others suffer from tracking too *many* things and therefore diluting the clarity of purpose for both themselves and the teams responsible.

To avoid this, determine around three key results per objective and make sure that they are SMART goals—specific, measurable, achievable, relevant, and time-bound. Key results describe outcomes, not activities, so words like *consult*, *help*, *analyze*, and *participate* are red flags.

In our prior example of wanting to improve product rework efficiency, you might set the following KRs:

- Reduce the percentage of total time spent on bug fixes from 20% to 10% by EOY.
- Reduce the percentage of time spent refactoring code during each sprint from 10% to 5%.

- Reduce the number of bugs found in acceptance testing by 40% for each delivery by the end of Q2.

To make these metrics real, the team has to have a clear understanding of where to find them and clarity on how they are measured. Additionally, measurable key results should include evidence of completion, and that evidence should be available, credible, and easily discoverable for everyone. Here are some important points to remember:

Tracking metrics that indicate improvement for the people involved is equally as important as tracking the metrics for the business.

In the reducing product rework example, many organizations would simply track the highest-level metric—the overall reduction in cost—because that is the *business* goal. The problem is that it keeps the people focused on something that doesn't provide *them* with intrinsic value. Doing both is a winning combination. Tracking the metrics that intrinsically matter to members of the team and giving them the flexibility to work and organize as they see fit to positively influence these metrics gives your transformation the staying power it needs.

Both the broader business and individual team managers should strongly guard against weaponizing these transformation metrics.

The metrics need to be an honest and accurate accounting of the way things really are, yet they won't always tell an objective and complete story—especially *across teams*. Saying that one team is performing better than the other because one team's metrics are better than the other's is a big, juicy temptation that seems to make sense on the surface—particularly to higher-level managers who are primarily concerned with the business outcome! The fatal flaw of this temptation is it presupposes that the "best" outcome possible is consistent across teams and across projects. *This is fundamentally untrue.* Each project has varying degrees of friction due to different levels of complexity, existing technical debt, audacity of the attempt, and aggressiveness of the goals. Said another way, the rework metrics of a team managing a mature in-market product look drastically different than the rework metrics of a team working on a nascent 10x Moonshot. This is obvious. However, the is true for all projects to a varying degree. *If you weaponize your transformation metrics,*

> *they will be gamed, and you nerf your best tool for ensuring holistic progress.*

To summarize, progress is measured by keeping track of appropriate key results, which provide value to both the business and the individuals responsible for delivering the desired outcome. These key results are often best expressed and tracked as specific metrics that are based on SMART goals and deliver transparent outcomes that are well understood by all stakeholders.

In complex situations, you may find that nesting OKRs at each organizational level tremendously improves your ability to make progress on multiple fronts. Used this way, OKRs can assist in actually delivering on time a traditional KPI pyramid.[3]

Pitfalls

There are many pitfalls in the journey to your North Star state—fortunately, most are easily avoidable for the organization that is committed for the long term and is disciplined in creating holistic progress with each and every milestone. We've already outlined some relevant pitfalls in line with each of the preceding topics, but there are a few meta patterns that should be called out and guarded against.

Failure to Recognize the Existential Threat

This results in a failure to prioritize the transformation or a lack of intestinal fortitude to see a multiyear program through. Done well, the transformation should be tightly bound to strategic imperatives that drive the company over the next few years, giving it (a) value and (b) longevity. CIO longevity (or lack thereof!) can sometimes be a challenge, so making sure that support for the initiative is broad-based, shared by the lines of business and mandated by a CEO who understands its importance, is the ideal situation. For most companies, going through a digital transformation at some point is a hard requirement to stay in business. It is nearly impossible to compete with a market entrant that is faster to market, more cost effective, and better aligned with the changing needs of the user. An

[3] Search "kpi pyramid," or look at an example for SaaS products at *https://www.productplan.com/saas-product-metrics-pyramid*.

organization that doesn't have the consistent willpower to disrupt itself is living on borrowed time.

I Have 99 Problems, and They Are All Squeaky

Many organizations fail because they get distracted and focus too much on specific tactical pain points, aka the squeaky wheel syndrome. This will often occur because of the natural tendency to over-index on the "heroes of the past" versus aligning toward the future. These people know where all the skeletons are and can make challenging areas of change seem nearly impossible. In addition, they usually have enormous organizational influence, and many people will look to them to see how they react to the transformation initiative. To avoid this pitfall, make sure that these "heroes" have a clear role to play in the new vision, one that excites them and makes them some of your strongest supporters instead of being some of your largest roadblocks. Sometimes this involves inviting them to take either an active or a mentorship role in transformation activities. At other times it is more about empowering them to "hold down the fort" as the rest of the organization explores the best options to move forward. This allows you to "desqueak" the wheel without stopping the train of transformation.

Crisis-of-the-Day Thinking

Many organizations fall prey to an unhealthy focus on short-term savings for budgetary reasons, with the result that your transformation program is hijacked by the Crisis of the Day. Holistic transformation isn't free, nor is it inexpensive, and it is easy to put aside what's important for what seems to be most urgent. Oftentimes an organization will shortchange its future because of the very real constraints of the present. CFO-led organizations are particularly prone to this pitfall. Making an ally of the CFO and ensuring that they 100% understand and agree with both the long-term and the intermediate value of the transformation will help to guard against this situation. Sales-led organizations can also fall prey to this pitfall, as this year's "Big Renewal" may sabotage planned change and underscore old patterns that a few big customers still prefer due to their own lack of movement forward. Guard against this by building a joint vision with your largest, most influential customers. Don't just inflict change on your customers either—show *them* the progress as well, and be the ones that can help them achieve it.

Even with all of this important alignment done properly, transformational leadership needs to be flexible and realize that these constraints will absolutely have an impact on the velocity of progress. Don't self-aggrandize the initiative (your market position may give you more time to change than another firm), but as important, don't shortchange it (no one is immune to the need to evolve). Plan it out mindfully, and make sure that competitive threats are properly understood and weighted appropriately by all stakeholders—the CEO, the CFO, and the CRO, most critically.

What a Cute Little Project

This comes down to a failure to scale well-intentioned projects—small initiatives show promise but have limited value or no path to scale with broader teams. Many organizations have been working on transforming themselves for some time but have seen frustratingly limited success with holistic change. This is typically due to one of several reasons:

- Innovation is occurring in tangential or less important areas of the business, without any natural path to the broader organization. Transformation initiatives should be undertaken in areas that are a strategic imperative for the organization—this will give them the resources, focus, and appropriate ROI needed for scaled change. Each initiative should be aligned with a specific milestone on the road map to the North Star state.
- The initiative didn't balance progress for the business with progress for the underlying teams. In such cases, the initiative has a tendency to simply fizzle out because of the lack of intrinsic value to the participants. No organization can properly police every aspect of the change needed for holistic transformation, nor would it be effective if it could. People have to *want* to change because they see the inherent value for themselves, and then everything else in the organization works to support them in that goal. When people see their personal progress, they will persist through the challenges and deliver for themselves as much as for the business.
- There is a lack of clarity across the team as to how to achieve the desired outcome and what is required to deliver the requested change. This situation can be avoided by using the OKR method to clearly outline specific objectives per milestone and

materially keep track of the progress through the use of measurable key results. This keeps people focused on the things that deliver the desired outcome instead of their being distracted by the various activities that may be employed in the process.

Don't reflexively question your organization's ability to change—first question the methods used to accomplish the change. Following the recipes outlined in this report and adjusting as necessary, without violating the core principles, should give you enough progress in strategic areas to use as an anchor point for scaled transformation.

Fiscal Flexibility at the Cost of Efficacy

Another pitfall of traditionally CFO-led organizations is a blind focus on a multi-cloud strategy. On the surface, multi-cloud can make a lot of sense. First and foremost, it doesn't lock you into a single provider. Enterprises have dealt with technical lock-in for decades, and while technical shackles were a hallmark of the last 40 years, the cloud provides an opportunity for flexibility to be the hallmark of the next 40. Second, the desire to consistently get the best price by playing one cloud provider off another will often lead people to build an arbitrage market for cloud services within their own organization. The thinking here is that by enabling their workloads to work on any cloud, they can seamlessly shift spend to the cloud that provides the lowest cost at any point.

We wholeheartedly agree with the first point, and most Google Cloud Products embrace open protocols and standards that provide a strong foundation, where flexibility is an equal citizen with capability. We also agree that many organizations, particularly ones that are heavily regulated, may benefit from a multi-cloud approach in both the short and the long term.

Where things can begin to go wrong is when organizations start to cater to the lowest common denominator available across multiple clouds in search of the absolute lowest cost per cycle in all areas. We recommend a balanced approach in which the cost of switching is compared to both the TCO and ROI of workloads that embrace differentiated cloud capabilities. Many organizations have spent significant portions of their transformation dollars unnecessarily building themselves a cloud-agnostic platform. Doing so is not their core competence, nor does it give them a rational ROI when all costs are considered. This is classically being penny wise and pound foolish.

For example, attempting to create one enterprise platform that seamlessly obfuscates the differences between Redshift, Snowflake, and BigQuery is a tall and expensive order and would negatively impact what can actually be accomplished with the final product. Said another way, the intersection of the capabilities of these products is much smaller than the superset of their overall capabilities. Limiting yourself to only the subset in an effort to get the best price per project will cut you off from many transformational capabilities that you'll end up competing with in the market but that won't be available to your organization.

To guard against this, fairly calculate the total cost of ownership and accurately reflect the return on investment rather than just the base cost. While there are situations that legitimately benefit from a multi-cloud approach, most often the facts show that (1) the cost of building and maintaining a cloud-agnostic platform, (2) the loss of functionality due to providing only the lowest common denominator, and (3) the additional hard cost of scaled data egress back and forth across multiple clouds will together far outweigh the cost savings obtained by running on the lowest bidder du jour.

In situations where multi-cloud is beneficial, we recommend focusing on open source protocols and open source technologies and building the proper layers of abstraction needed to swap out underlying technologies as needed without significant application disruption. In fact, we've developed Anthos as a platform to help people move in this direction if their business needs this core flexibility.

CHAPTER 4

Framing Your Transformation with Clearly Articulated Policies

Lydia Thomas, Solutions Consultant,
and James Brookbank, Cloud Solutions Architect

Who

- Business Functions
- Team Managers
- Technical Leads

Why

Cloud transformation is often treated as a traditional project for enterprises, something that the technology team undertakes in isolation and that broadly follows the same textbook approach for every organization. However, real-world enterprises are a unique mix of ingredients, including policies, regulations, people, and culture baked in over dozens or even hundreds of years of history. Organizations are significantly more likely to reap tangible benefits from cloud computing when they directly link cloud transformation principles to their business objectives.

For many enterprises, cloud isn't the first technology transformation they've undergone—frequently these transformations mean labeling the existing technology estate as "legacy" and buying or building new software or hardware. Many of these technology transformations have been expensive failures at best; however, successful transformations have occurred at those organizations that treat their deep understanding of regulatory and business environments as an incredibly valuable asset. For this reason, you need to use this exact same approach when making the journey to the cloud.

How

Start by relating your core business objectives to transformation principles and policies. Principles are the fundamental truths that form the foundation of your transformation and help guide decision making. There are often multiple ways to achieve business objectives, so encouraging people to live and breathe a core principle is better than setting exhaustive rules that can be followed in letter but not in spirit (Google's principles are an example of the former). Your focus should be on enabling people to demonstrate leadership at every level rather than being bound by a series of directives that disenfranchises the individual.[1] In particular, business functions and managers need to be persuaded by the transformation narrative and must be willing to amend the detailed guidance within the context of their specialist areas. These influencers are your greatest assets once convinced, and your biggest hurdle if not.

Similar to principles, good policies focus on outcomes and not tasks; however, they are more prescriptive guidance. They are the vehicle to harness the bureaucracy of your business instead of fighting against it. Policies and policy frameworks should empower people to operate safely within well-understood guardrails. They should also contain sensible defaults to nudge behavior in the right direction.

When developing a system with machine learning (ML), you must take care to establish strong principles from which the ML model can act. As you entrust important tasks and processes to the ML in this system, bear in mind this basic truth: the original labeling of data we use to train a model is carefully selected to ensure that the model behaves in the way we expect it to, while performing its tasks to reach some target. We think the same can be said for organizations—setting targets in an organization without guiding principles for how those targets should be achieved is akin to

[1] See Chapter 3 for more detail on this important subject.

deploying a machine learning model with no curation or training toward a common outcome.

What, then, are some examples of key principles that empower a leader to be “present in principle”? This is a leader who sets out and reinforces strong principles within their organization, and those principles are represented at every meeting, big or small, even when the leader is not actually present. Strong principles are what bring you closest to meaningful delegation, and they guide the decisions being made in every facet of your organization without micromanaging. During Project Aristotle, in which Google went about testing what truly made the most effective team, the top two attributes were psychological safety and dependability. Creating these pillars within your organization may seem implausible at first, but reinforcing these principles by continuously tying them to decisions and role-model behaviors helps challenge poor behaviors and ideas without altering your organization’s common goal.

How, though, do you decide what principles are already incumbent in your organization, be that by design or not? Also, how do you know which principles, above and beyond what’s been shown by studies, are important for your organization or industry? Assessing the characteristics of an organization accurately without a method is untenable—what you observe in a leadership position is highly likely to have information bias due to your presence. In contrast, sampling theory goes a long way toward getting you, the leader, a representative view of the characteristics and principles of your organization. Small, representative groups with carefully designed samples scale accurately to tell you what existing principles you’re dealing with, and whether to foster them or counter them.

It’s critical to remember that policies aren’t “set and forget” activities. Only by frequent and visible reinforcement of behaviors can they be consistently adopted. Despite the temptation to view cloud transformation as solely being technology driven, among all principles and policy areas, cultural transformation is actually the most valuable component (and unfortunately the hardest), as Chapter 2 describes.

Pitfalls

There are a lot of potential challenges to a successful cloud transformation. What follows are some real-world examples of anti-patterns that we’ve seen organizations encounter when setting policies. This isn't designed to be an exhaustive list and doesn't necessarily mean you'll experience any (much less all) of these scenarios.

Forgetting that the Journey Is the Goal

One of the more common scenarios is to make the transformation an end in itself, with policy statements such as "we need to transform in order to modernize applications." It's true that applications frequently do need to be modernized in order to solve a business problem—for example, time-to-market for new features. However, the time-to-market is the real principle that deals with increasing business agility, and modernizing apps is just one way of achieving this.

The Half-Hearted Transformation

Transformations are often attempted piecemeal for good reason, especially in large organizations where changes can't all be made at the same time. However, when changing a policy, it's common to leave current financial or performance incentives in place, hoping that individuals or departments follow the new policy, even if it's to their own detriment. We've seen extremes of this where teams were basically told that their jobs will disappear once applications have moved to cloud. Unsurprisingly, the migration progress was extremely slow! If you set a policy, make sure your teams are incentivized to follow it.

Building Without a Strong Business Foundation

In a similar way, policies are often changed without the basic foundational elements needed to support them. If your current business objective is cutting costs, then it seems counterintuitive to spend money on seemingly unrelated activities. A common example is a policy to shift left when approaching security. This is a highly recommended approach that can result in more secure applications delivered both faster and more cheaply. However, a lofty principle such as "security is everyone's responsibility" needs to be accompanied by a rock-solid foundational investment in tools and training.

A Transformation in Letter, Not in Spirit

Policy changes can often occur as a result of external influences such as regulators. These can often be phrased in terms of activities instead of outcomes. At one organization, the statement was made that "the regulator needs this particular software package installed for security." This is the same anti-pattern as with business objectives—regulators do not want tick-box governance, where an organization follows the rules in letter but not in spirit. In this scenario, the regulator absolutely wanted the outcome to be an increase in security, but the software decision was actually made by the organization. The policy should have focused on the security outcome and not on the software used to achieve it.

CHAPTER 5

Building Leadership Through Decider Groups

Joseph Bironas, Solutions Architect

Who

- Technical Leads
- Engineers
- Enabling Team Leads
- CxOs
- Managers

Why

Real agility lies in the ability of organizations to decide and react quickly. Often our natural organizational communication channels are structured in ways that inhibit the people closest to the problem to act in ways that are nimble and aligned with the business. Centralized communication patterns create organizational bottlenecks that stifle progress and innovation. Top-down methods of decision making push authority to people who frequently have less direct knowledge or context for making well-informed decisions.

To have agile projects and development breakthroughs, it's necessary to allow for easy structure and restructure of communication channels and information exchange patterns to push decision-making activities to people who are best informed. This runs contrary to common hierarchical habits of pushing information up to those with the authority to make decisions.

DevOps Research and Assessment (DORA) regularly analyzes many companies to determine what practices lead to companies' success. The 2019 Accelerate State of DevOps Report established four key findings related to organizational decision making. Two state that agility and communication patterns are critical elements to leading technological transformation, and two state that increased productivity and change management processes are factors in improving work-life balance and reducing burnout:

- Delivering software quickly, reliably, and safely is at the heart of technology transformation and organizational performance.
- The best strategies for scaling DevOps in organizations focus on structural solutions that build community.
- Productivity can drive improvements in work-life balance and reductions in burnout, and organizations can make smart investments to support it.
- There's a right way to handle the change approval process, and it leads to improvements in speed and stability and reductions in burnout.

In practical terms, we often see developers complain about a lack of autonomy or authority to make or execute on decisions. Frequently, this is coupled with complaints about the quality of upper-level decision making. Those same developers commonly escalate information to managers or executives for key technical decisions.

These behaviors indicate a number of symptoms—specifically:

- Lack of authority to make decisions
- Lack of oversight or review of important technical decisions
- Dissatisfaction and/or a sense of inability to change communication patterns
- Lack of clear organizational roles defining decision scopes and authorities
- Lack of clearly defined guardrails in which decisions can be delegated

High-functioning organizations recognize the steering committee approach as an anti-pattern.[1] They have leaders at many levels, not just at the top. High-functioning leaders know the scope of their decision-making abilities and lean on experts to make decisions for which they are suited. Cross-functional teams form or reform frequently to bring in stakeholders or new experts to surface relevant data and define new actions in almost real time. More frequently, teams are using non-meeting communication channels, such as group chat, to trade information and make decisions, rather than pulling people into expensive meetings.

The 2019 Accelerate State of DevOps Report shows that the most effective ways to scale practices are proof of concept as a template and seed, grassroots efforts, and communities of practice. Building communities and moving decisions closer to the problem reduces the chance of errors and increases the likelihood of success while improving happiness, productivity, and autonomy overall.

As you'll see, changes to how decisions are made can drive opportunities for people to lead (i.e., shape projects and strategies) and build communities of interest[2] around a number of problems, which leads to better outcomes—both in terms of the quality of technical decision making and for the teams and individuals as well.

How

To prove this highly cross-functional model of leadership development and communications organization, let's look at a realistic albeit contrived example. A company would like to switch from a homegrown CRM solution to a newly purchased SaaS solution. How does a company most efficiently and accurately make this transition? This case describes the organizational models, decision-making structures, and review processes that must be in place to drive major change in the most effective way possible.

Our fictitious company has two tools supporting decision making: RACI and RAPID.

[1] For more on organizational decision making, see Paul Rogers and Marcia W. Blenko, "Who Has the D? How Clear Decision Roles Enhance Organizational Performance", *Harvard Business Review*, January 2006.

[2] See Chapter 6 for more details.

RACI is an acronym that describes roles in a responsibility-assignment matrix:

Responsible
: Those who do the work to drive the making of a decision.

Accountable
: Those who are ultimately answerable for the outcome of a decision.

Consulted
: Those who are sought out to provide input or guidance on decisions.

Informed
: Those who need to be kept up to date on the impact or overall outcome of decisions.

RAPID is an acronym that describes the phases of decision making:

Recommend
: Those responsible for proposing, gathering input, and winning buy-in

Agree
: Those required to buy in in order to execute on a decision; they may negotiate modifications with the recommender over concerns

Perform
: Those responsible for executing decisions promptly and effectively

Input
: Those responsible for providing key information to the recommender to determine feasibility and implications

Decide
: The single person responsible for being held accountable for the decision, committing the organization to implementation, and closing the decision process

The RACI framework is primarily used to determine who is required for decision making and their role in the process. A RAPID decision starts when one or more persons develop a proposal or recommendation on a key decision. Each of the phases in RAPID may have different RACI memberships.

Before attempting to make a decision, it's important to determine the degree of risk associated with the decision. Is it high risk or low risk? Is it easy or hard to undo a bad decision? In those cases where there is low risk and the outcome is easy to undo, it's safe to push the decision as low as possible. Doing so creates learning opportunities for deciders in a way that is safe for the company if the decision doesn't go as expected. When decisions are higher risk and harder to undo, more people are generally involved in the process, and the consequences of bad decision making are much more costly. Figure 5-1 presents a visual guide to making decisions from the perspectives of risk and reversibility, describing the organizational outcomes.

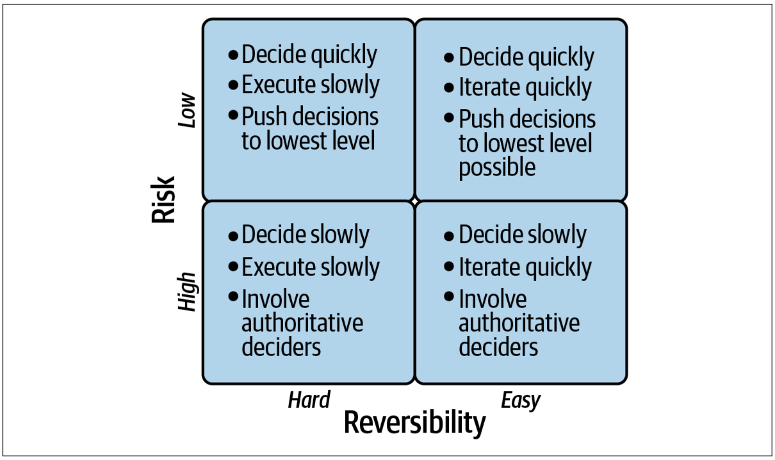

Figure 5-1. Risk versus reversibility for decision making (adapted from source by Joseph Bironas)

Commonly recurring problems are opportunities to build a "community of interest" (sometimes called "centers of excellence" or "guilds") among thought leaders in the space, to raise their visibility and ensure that expert voices are informing key decisions that touch specific sets of problems.

In higher-risk/higher-cost scenarios, it's possible to push decisions to the lowest level possible by leveraging well-informed members of communities of interest in various phases of the RAPID decision-making process. Decentralized and low-level decision makers do not connote “lone wolf” or “working in isolation.” Peer and executive reviews are important steps in high-quality decision making.

In a complex project such as the decision to move from one CRM to another, one of the biggest decisions is where to begin. What data and workflows do you migrate first? How should that decision be made? Arguably, the highest impact/lowest risk cohort of topics and users that exist today would be the ideal starting point, but who knows what that is? Is it a technical or business decision? Is it the VP executive sponsor? Is it the director of the Customer Experience team, whose interest is principally tied to the consumers of the service?

Overall execution depends on a large cross-functional team consisting of director and VP-level sponsors, line-level managers, product and program managers, software engineers, customer support specialists, and others. I’m using Objectives and Key Results (OKRs) for this example, but any well-established goal with a measurable outcome works. They are all ultimately responsible for the objective: “decrease the time spent delivering solutions to customers.” To simplify, let’s assume all executives reinforce a culture of pushing key decisions down. This obviously glosses over details of getting that level of buy-in, but the process remains roughly the same, regardless of the size and complexity of the decision.

One of the directors asks a program manager to be the recommender and proposes a short list of candidates to pilot the migration. Primarily, this decision rests on the axis of “risk of customer impact” versus “technical complexity of the move.” The program manager enlists the help of support specialists, who have direct knowledge of systemic, customer-facing risks, and they begin to inform plans for the initial migration cohort. The program manager also enlists the help of software developers interested in optimizing the CRM experience, who have visibility into the performance characteristics of the existing and new systems.

The program manager, in collaboration with these groups, provides them opportunity and leverages their experience to get agreement and buy-in directly from the teams that are responsible for performing work and maintaining the solution. This short-circuits the pitfalls of a “steering committee”–style practice in which information is relayed through faulty channels in order to make decisions and provide buy-in, which may be incomplete. Once the obvious nonchoices are removed due to a higher-than-acceptable risk or impact to customer experience, a shorter list of candidate targets remains.

This small cohort becomes a separate team during the time it takes to reach a decision. They are responsible for:

- Gathering facts, analyzing data, and sharing them with fellow teammates
- Understanding risks, recognizing decision points, and recommending solutions
- Seeking buy-in from relevant stakeholders who might be impacted by the decision
- Reducing or mitigating risk factors impeding the success of the project

This team, empowered[3] to build consensus among its members and make decisions, has everything needed to review the decision with executive sponsors and move forward.

The key elements of this solution are:

- A culture of pushing decisions to the most informed deciders and incentivizing the success of projects through problem domain expertise rather than level or role
- A sober interpretation of the risks, costs, and potential negative outcomes associated with the decision
- Clear roles and responsibilities for actors in the decision-making process
- A collaborative attitude and clear, transparent communication among decision team members
- The time, experience, and skill to bring relevant information and facts to the table
- Peer and business-level review with relevant stakeholders, including those who need to be consulted and informed on any decisions.

[3] For more on empowering employees, see David Marquet, "6 Myths About Empowering Employees", *Harvard Business Review*, May 27, 2015.

This example, though intentionally simplified, shows how complex problems are turned into decision points that small cross-functional teams close to the problem are capable of solving. This approach has many organizational and professional growth benefits for those involved. Businesses that distribute leadership responsibilities more equitably among their team members see greater project success and employee happiness. Leaders emerge throughout the organization based on their skill, knowledge, and ability to collaborate on solving hard problems, rather than on arbitrary measures not aligned with business outcomes. Escalations aren't necessarily part of the process and can be avoided to attain peak agility and performance.

Pitfalls

Decision making is fraught with risk, and you will encounter obstacles along the path to sustaining an organization with transparent processes and collaborative attitudes. Let's take a look at some of the issues you may come upon and why they present challenges.

A Few Loud Voices Dominate

Ensure peer review by preventing the "loudest voice" from dominating the conversation and blocking conflicting ideas or information from coming to the foreground. Pull out quieter voices to increase the pool of available information that leads to higher-quality decisions.

Toxic Mindsets Limit Individual Empowerment

Resignation and cynicism undermine information exchange. Beware of environments in which either attitude thrives. To build good leadership skills, it's necessary to focus on clear, collaborative communication and empowerment. Practices such as micromanagement, mocking, or retaliation can disincentivize knowledgeable decision makers from actually making decisions.

Lack of Investment in Training

When moving from a top-down compliance model of operation to a team-based decision-making model, there's a strong dependence on competence and clarity to make successful decisions. Spending time on clarity of purpose and technical training helps prepare emerging leaders.

Imbalanced or Imprecise Empowerment

It's necessary to be able to measure and affect empowerment. Be specific about the activities required, such as "generate options" or "come up with a plan." These set concrete terms on which emerging leaders can establish their footing.

Crisis management leverages structure, coordination, and communication, often under high stress and at a fast pace. Highly trained teams apply a different organizational structure in crisis situations. Leaders step into roles regardless of their day-to-day positions. Managers who can't find their role during a crisis can dramatically impede decision making and empowerment.

Lack of Leadership Visibility

When decisions are made and positive outcomes are gained, ensure the leaders of those decisions get the recognition and visibility they deserve. Celebrate the positive impact and recognize those involved in the decision within the business and even publicly.

CHAPTER 6

Developing Centers of Excellence

Kuntal Mitra, Staff Solutions Consultant

Who

- Technical Leads
- Engineers
- Enabling Team Leads
- CxOs
- Managers

Why

Building an internal team to drive cloud adoption success through common practices is a natural inclination. At Google, we've seen this team called many different things: Cloud Services, Center of Engineering, Innovation Council, Cloud Engineering, Community of Practice, and even Cloud Platform. We refer to a team so organized and directed as a Cloud Center of Excellence, or a CCoE. A well-appointed CCoE begins with a small team that understands the vendor's cloud adoption framework[1] and is able to use it as a guide for implementing cloud technology that is aligned with business goals and strategy. The Cloud Center of Excellence team then becomes the

[1] For information about the Google cloud adoption framework, see *https://oreil.ly/adopt-frame*.

enabler for transforming how other internal teams serve the business in their transition to cloud. However, a CCoE should not be considered a blocker for transformation or for migrations to occur.

The CCoE team accelerates cloud adoption by:

- Aligning cloud goals with the larger organizational and business strategy
- Providing leadership, best practices, research, support, and training for the move to cloud
- Advocating for cloud adoption within the organization and helping to ignite enthusiasm
- Serving as a catalyst for innovation
- Creating the infrastructure and framework that are necessary to become a cloud-first business

Furthermore, we've observed that successful CCoE teams are multidiscplinary. Members of the team reflect the diverse perspectives of the stakeholders in the organization. The initial CCoE team starts small with the following core roles as it builds its cloud strategy, road map, team, and governance:

Leadership:

Executive Sponsor

Provides leadership and direction on the cloud strategy and ensures the cloud team has the appropriate support, resources, and funding. Responsible for advocating and demonstrating behaviors and mindsets that encourage a cloud-first culture.

Program and Technical Cloud Team:

Cloud Lead

Oversees the CCoE and the overall cloud transformation efforts. Responsible for working with the executive team on cross-team collaboration and ensuring the cloud road map aligns with the wider business goals.

Cloud Architecture Lead

Defines the cloud design and network architecture vision. Works closely with cloud architects, network engineers, security, and cloud engineers to implement best practices that allow for scalability.

Cloud Operations Lead

Executes management, maintenance, and support of scalable solutions. Responsible for the end-to-end life cycle of cloud applications and their ongoing support.

Cloud Security Lead

Responsible for setting the vision and standards for security standards and controls across cloud applications and infrastructure. Ensures software and services are designed and implemented to the highest security standards.

Business and Cloud Adoption Team:

People and Adoption Lead

Leads change enablement across the organization and on individual programs as the enterprise adopts cloud. Advocates for adoption and develops strategies to create a cloud-first culture.

Cloud Finance Lead

Enables cloud cost and showback capabilities. Develops standards and requirements to optimize costs and trace cloud costs to originating business units; drives change from CapEx to OpEx.

Teams are:

Empowered

Team members have decision-making authority without the need for higher-level sign-off.

Visionary

The team takes a multiproject viewpoint to understand repeatability and long-term benefits or goals for the organization.

Agile

The team understands how to deliver short-term wins such as short development cycles and an iterative approach to building products.

Technical

The CCoE should include experienced individuals with a history of architecting and building past solutions within the organization.

Integrated

Individual members come from existing areas of the business to allow for easy integration into existing teams and organizational constructs.

Hands-on

The group includes individuals who are able to do the hands-on work needed to build and test cloud solutions.

A CCoE is not a static entity. Rather, it continually evolves to keep pace with the innovation associated with adoption of the cloud. Executive leadership should design the CCoE as an adaptive structure that evolves as the needs of the organization do.

How

A CCoE provides guidance, facilitates adoption of leading practices, and promotes *proactive governance* to consistently *accelerate and derisk* cloud adoption. An effective Cloud Center of Excellence drives, supports, and accelerates cloud transformation by enabling the enterprise-wide adoption of *business-aligned* cloud strategies.

This CCoE needs solid and widespread advocacy from within the organization. Without such advocacy, all responsibility for generating enthusiasm and momentum for cloud adoption rests with the executive sponsors. This top-down approach not only can be slow to scale but also fails to embrace the inherent democratization of IT resources that cloud computing offers.

One way of generating such advocacy is to bring together a group of people from within the organization to provide leadership, best practices, research, support, and training for the move to cloud. With a team like this, it becomes easier to foster engagement and momentum organically and also to create the infrastructure and the frameworks necessary to become a cloud-first business. As the strongest advocate for cloud adoption within the organization, the team helps ignite enthusiasm. It can also serve as a catalyst for innovation. The team members, deeply attuned to the business strategy as they are, find new ways to achieve the desired goals. The initiatives and pilot projects that this group develops are more likely to evolve into endeavors that position the organization for long-term success.

The CCoE should be a cross-functional team of people with deep skills and expertise. It is important that the skills of the CCoE team members be robust enough to support their roles and activities. Organizations at different maturity levels and with different cloud goals and business objectives require different roles in their organizations.

The CCoE team should start with a few critical roles, covering skills across program management, cloud deployment, change management, and support, such as *Program Leads*, *Technical Leads* across Operations, Architect, and Security, *Product Leads*, *Managers*, and *Engineers*, with support from *CxOs* and other executive-level IT leaders. The initial team of cloud experts and advocates will be critical in setting up the foundational cloud strategy and operations in partnership with the business. You should plan to staff the CCoE with a technically capable and highly engaged team, composed of full-time leads with support from full-time and/or part-time engineers based on needs, as you build your cloud strategy, road map, team, and governance. Ideally, the team should be staffed internally from different business functions to bring the right skills, so that team members are aligned with the corporate culture and business strategy and come to the role with their corporate networks already in place. To buttress the team with additional skills not available internally, organizations may turn to external partners and/or consultants. This begins the progression toward their goals and objectives, along with the effort to skill people across functions in cloud knowledge. Less obvious is the need for stakeholders from various *business functions*. Business agility and time-to-market are one of the key motivations for the formation of a CCoE. As such, the key stakeholders should have a vested interest in these areas. Examples of business stakeholders include line-of-business leaders, finance executives, operations executives, and business product owners.

During the initial phases of designing and building a CCoE, strategy, socialization, and engagement are key activities. After a successful launch period, as application migration plans increase, additional key roles should be defined to add critical expertise based on cloud goals. Further defined roles should allow for deeper focus to enable additional migration agility and innovation. Finally, as the organization becomes increasingly mature in the cloud, each function of the CCoE grows into transformation squads, each one with specialized knowledge and full-time cloud-related responsibilities. As the CCoE evolves and continues to grow over time, some portions of the CCoE may be dissolved as they are adopted by different business units or may no longer be needed, as the organization will increasingly be populated with engineers who have experience and a deep understanding of how critical functions currently operate.

CCoE isn't a one-time implementation. The Cloud Center of Excellence continually evolves to keep pace with the rate of innovation associated with cloud adoption. Modern enterprises should be very purposeful with the organizational design principles of their cloud program structure. The leaders of cloud programs should observe the flow of their value within the organization and design an adaptive structure that evolves alongside the internal needs of the enterprise. To realize value from CCoE, organizations should take time and follow these steps:

Perform organizational analysis

Any cloud implementation strategy should be grounded in an organizational analysis. The analysis documents insights from user groups, change impact, organizational needs, potential watch points, key influencers, and use cases, which eventually helps the organization to establish its cloud vision, mission, strategy, objectives, and key performance indicators to measure the progress of CCoE. The cloud vision maps business objectives and needs to technology capabilities. It provides the sponsor with a key tool for selling the benefits of moving to the cloud to stakeholders and decision makers within the enterprise. The vision is critical to establishing a realistic adoption strategy. It provides the business context for making cloud computing transformational for your business.

Define strategy, vision, and mission

The crucial part of creating success is to define the strategy, vision, and mission of the CCoE. The CCoE should be deeply attuned to the business strategy, and the initiatives that this group develops are more likely to evolve into endeavors that position the organization for long-term success.

Secure strong executive sponsorship

A successful CCoE empowers transformation. For a CCoE to be successful, it must itself be empowered, endorsed, and supported by strong executive sponsors. Comprehensive buy-in should expand across the C-Suite and must reach across business functions, not just the IT function. Strong executive sponsors are actively involved in cloud deployments. They champion CCoE practices across the organization, and they support the CCoE through its evolution.

Build a cross-functional team

The foundation of a successful CCoE is its members. The level of expertise, vision, and willingness to build the necessary intellectual property for cloud delivery dictates the success of the Cloud Center of Excellence and the cloud implementation. Therefore, it is important to select CCoE members carefully. Plan to staff your CCoE with a technically capable and highly engaged team. Ideally, you'll be able to staff internally, so that team members are aligned with the corporate culture and business objectives.

Engage with the organization

CCoE members are the cloud champions within an organization. Their work should excite and inspire other employees about the benefits of moving to cloud. For better leverage, the CCoE should be placed high enough within the organizational structure that the team is able to create the momentum necessary for a successful cloud adoption journey.

Measure performance

Build OKRs to measure the performance of CCoEs over a period of time. The objective should be the mission statement of the goals, expressed concisely. The key results are measurable outcomes that, taken together, imply that the objective is reached if the key results are reached. Measuring the performance of CCoEs is critical to their evolution and to keeping pace with the innovation associated with the adoption of cloud

Optimize and scale

Constantly evolve your cloud operations to meet future cloud objectives. Using detailed business and cloud maturity assessments, determine the current state of CCoE capabilities and map those to incoming cloud initiatives to gain insight on areas of improvement for the CCoE, in order to optimize current state and drive greater value.

A well-designed and thoughtfully implemented CCoE not only helps to jump-start a large-scale, organization-wide transformation but also accelerates an organization's transition to the new operating model. To learn about building a Cloud Center of Excellence, download the whitepaper "Building a Cloud Center of Excellence" for practical guidance and strategies.

Pitfalls

Starting a CCoE is a significant undertaking, and there are many potential challenges to building a successful center of excellence. What follows are some real-life challenges that we have seen organizations encounter when trying to set up and operate a center of excellence. This isn't designed to be an exhaustive list and doesn't mean that you'll experience any (much less all) of these challenges.

Lack of Effective Communication

An effective center of excellence is built on strong communication. By communicating the CCoE's purpose, the executive team is able to build credibility, drive collaboration, and gain appropriate stakeholder buy-in and proactive engagement from employees across the business, since it is now able to better understand and correlate with the vision and benefits of the change.

Process and People

Oftentimes, the center of excellence becomes overwhelmed due to a lack of engagement, technical expertise, or self-service processes elsewhere in the organization. This can leave the CCoE ineffective at meeting growing demands. To help counter this, the leadership team should consider publishing strong guidelines for engaging the CCoE: processes for asking questions, intake forms for consulting and guidance, and utilization of self-service as much as possible.

Most of the skills or competencies that a CCoE is built for are niche or rare and hence are a major challenge. The skills of the CCoE team members must be robust enough to support their roles and activities. To provide the CCoE individuals with the knowledge, resources, and tools to succeed in their role, it's important to equip them with the right reskilling and/or upskilling resources. A needs analysis helps define the training plan, which captures information on which individuals receive training, the format and course those individuals receive, and the targeted time frame for training.

Misaligned Culture and Technology

A major part of successful CCoE adoption is cultural and organizational change. Therefore, the executive team must understand this and be able to effectively communicate this change throughout the organization. Both the executive sponsor and the CCoE itself must be held accountable for determining the goals and plan to bridge any organizational and cultural changes that may be encountered during the transformation process. It usually takes time to build a cloud-first culture while fostering innovation and growth as the top descriptors of the organization. To achieve this mindset, the CCoE should consider embracing a culture of cloud knowledge and thought leadership in the organization, socializing successes, hosting roadshows, and holding office hours or informal "brown bag" Q&A sessions to engage with stakeholders across the business and organization.

Bottlenecks in the CCoE

Occasionally the CCoE ends up being an exclusive group of "experts" in the organization, in contrast to an inclusive group of peers who continue to learn and grow together. This exclusivity fosters bad norms and behaviors that chip away at healthy organizational cultures. The experts are removed from doing the work. They are able to make recommendations or establish generic "best practices," but the path from generic learning to the implementation of real work is left up to the learners. For example, experts may build a workshop on how to containerize an application, but they rarely or never actually containerize applications themselves. This disconnect between theory and hands-on practice eventually threatens their expertise. In cases like these, the CCoE becomes a bottleneck for the relevant expertise for the organization, and this cannot scale as demand for expertise in the organization grows.

To help counter this, the executive team should consider hosting Cloud Immersion Workshops, forming tiger teams supporting pair programming to reskill the existing workforce and/or setting up "lightweight" CCoE hubs staffed with cloud champions, stewards, or custodians across the organization that promote leading cloud practices, innovation, and collaboration. It could also consider adopting communities of practice across the organization to enable Innovation at the Edges, which can lead to new ways of working—

operating in a transparent, collaborative, and autonomous environment. To continually drive the rate of innovation with cloud adoption, the CCoE should encourage cross-functional collaboration and look for ways to foster it within its teams and across the organization.

CHAPTER 7

Scaling Innovation

Awais Malik, Infrastructure Cloud Consultant

Who

- CxO
- Technical Leads
- Managers
- Engineers

Why

Brilliant innovators all throughout history have shown us the impact their creativity and inventions have had on us. Thomas Edison, Nikola Tesla, Benjamin Franklin, and many others did amazing things with *limited resources* and capabilities. They experimented and demonstrated their findings, in spite of their scant resources, and their inventions profoundly impacted our lives.

More recently, access to innovation platforms has become increasingly easier to gain. With the advent of cloud and SaaS services, building solutions has never been easier, and these have become the perfect catalyst to spark innovation.

It's apparent that no one can consider business innovation as just a nice-to-have anymore. As such, technology should be viewed not as a cost center but rather as a core capability and key business enabler. More importantly, enabling innovation at the micro level encourages teams and even individuals to rapidly experiment with creativity. This rapid experimentation is the key to fast failures, and many iterations of fast failures eventually lead to successes. Fortunately, the cloud enables this experimentation through its favorable procurement and cost model, which helps perform polynomial proofs of concept over time that are commercially feasible. With proper controls and well-defined performance measurement indicators, decentralization becomes the way to evolve your products and services over time while staying ahead of the competition.

How

To make this change a reality, establish a set of standards and processes, and identify the roles critical to helping organizations decentralize.

Standards and Processes

Establishing a set of standards and processes helps account for security and enables innovation from the bottom up.

Define a set of controls that allows you to properly manage and control the provisioning of resources across your cloud platform. Given the elastic nature of cloud, without proper definition and organization you run the risk of overprovisioning and compromising the security of your workloads and data. Lacking good, measurable controls, system engineers tend to overprovision, which leads to more attack surfaces that compromise security.

Project structure brings calm to chaos. The idea of running production workloads side-by-side with sandbox and "innovation" workloads sounds intimidating and even irresponsible. However, modern cloud platforms provide organizational capabilities with proper security in place for you to do exactly that, while ensuring the security of both data and workloads.

It is vital to define the right boundaries with respect to the type of workloads being deployed. These types should be categorized as production, stage, innovation, and so on, and provisioned in their folders' structures, which allow for consistent and relevant management policies to be applied.

Figure 7-1 is a sample illustration representing a secure yet flexible resource structure in Google Cloud Platform.

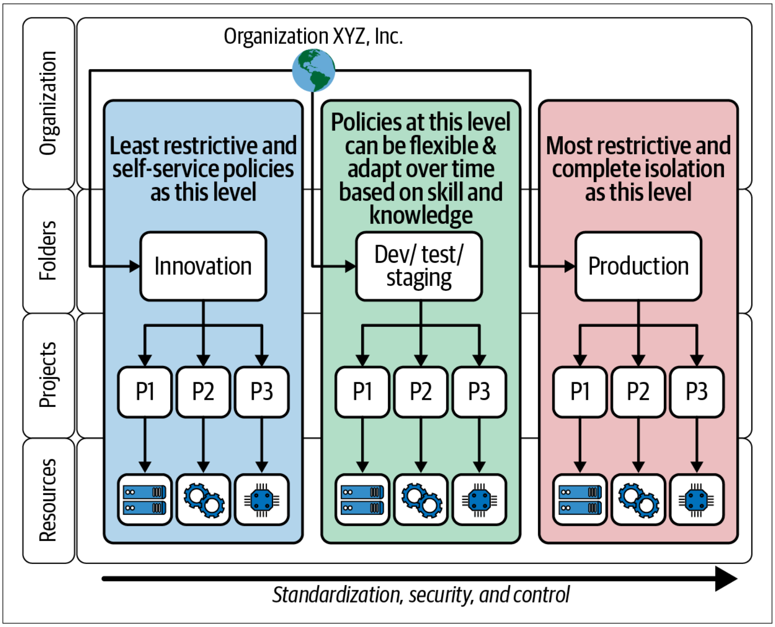
Organization
Folders
Projects
Resources
Organization XYZ, Inc.
Least restrictive and self-service policies as this level
Policies at this level can be flexible & adapt over time based on skill and knowledge
Most restrictive and complete isolation as this level
Innovation
Dev/ test/ staging
Production
P1
P2
P3
P1
P2
P3
P1
P2
P3
Standardization, security, and control

Figure 7-1. Google Cloud Platform resources (adapted from source by Google Cloud)

Services available on cloud platforms are increasing in number and continue to appeal to enterprise customers due to their limited management and easy setup. However, managed services tend to create heartburn for security and compliance, since most, if not all, management aspects are abstracted from them.

This is where a carefully structured project hierarchy can aid with enabling most or all services for innovation-type workloads, while disabling them for production workloads if they don't meet the compliance requirements or haven't yet been vetted carefully by the organization.

Cost control allows you to define and allocate budget for workloads, functions, and capabilities. Understandably, a significant portion of that is allocated for production workloads that help run the business. However, defining budgets using a benefits–cost ratio as an indicator and allocating them to various innovation teams can help measure the impact of their innovations with respect to spending. The insights produced from that are then used to adjust budget allocation and perhaps reward and further enable the teams that are doing well, while revisiting the objectives for the teams that aren't.

Secure your workloads by type, and define the right security policies for them. The most important aspects of security for cloud applications are identity management, access control, and service isolation. These policies should be conservative when it comes to production workloads; however, they can and should be relaxed for innovation workloads. This approach allows innovation teams to do more with the services available to them. Delegating access management also enables more effective and efficient collaboration.

Implementing this strategy may raise concerns with data management and the loss thereof when controls are loosened. Data management can easily be an essay on its own, but the most critical point to remember is to set a precedent to always use nonproduction or mock data for innovation workloads while carefully monitoring data egress and ingress. In addition, a stringent process should be defined for leveraging production data, if and only when absolutely necessary.

Monitor your workloads for activity and data. By definition, innovation workloads are supposed to have limited exposure. Metrics can and should be defined to monitor anything that goes against that assumption. Excessive data egress and spike in CPU and/or network activity are a few of the many metrics that can be monitored with alerts to ensure these workloads are not compromised by inside or outside threats.

We've discussed standards and processes that enable innovation, but as we see next, roles are important, too.

Critical Roles

The following roles are critical for helping organizations understand, embrace, and operate with decentralization.

CxOs and Managers are key to setting the tone for any change in an organization. Similarly, their voice is extremely vital when it comes to delivering a clear message. The message should emphasize the importance of creative thinking and embracing failures as learning opportunities. With that, they can be the driving force that shapes their organization's culture, defines success, and thrives on it in a decentralized environment.

Technical Leads play a vital role in defining the guardrails around the cloud platform as they relate to the foundational architecture principles and best practices. Scaling innovation means those guardrails must be readjusted in a way that gives innovators enough flexibility and access to innovate while keeping the most critical workloads for an organization running efficiently and securely.

Engineers, in our opinion, are where the rubber meets the road. With clear guidance from the leadership and proper provisioning controls in place, these contributors can start innovating. These individuals or teams should work with their immediate leadership to define goals that are measurable and in line with the overall objectives of the organization.

To conclude, it is absolutely vital to dedicate a task force that can help perpetuate innovation throughout the organization while creating a culture of decentralization. Chapter 6 details the mechanics of establishing and cultivating this practice at scale.

Pitfalls

To provide an environment for organizational and product growth, you'll want to ensure that teams work smoothly together with only the minimum necessary overhead. Striking this balance takes iteration and reflection, and you may encounter some of the following challenges.

Failure to Provide Autonomy

More often than not, organizations tend to create centers to monitor the state of innovation across various teams. Even though the intention behind these centers is good, they end up creating unnecessary hurdles for teams, which can slow the pace of innovation. The whole point of scaling innovation is to accelerate innovation and provide teams with the autonomy required to make decisions fast, experiment, fail, and eventually succeed over many iterations. Anything that slows that process should be reviewed and adjusted accordingly.

Failure to Provide Broad Access

Conservative access strategies are often applied with broad brushstrokes across all types of environments in an effort to enforce security. However, such strategies can be detrimental to innovation. As long as proper controls (mentioned earlier) are in place, innovation environments and workloads should be provisioned with significant access privileges for their respective teams and contributors.

CHAPTER 8

Higher-Order Architectures for Multi-Cloud

Poonam Lamba, Cloud Customer Engineer, and Patrick Sheehan, Enterprise Cloud Architect

Who

- CxOs
- Technical Leads

Why

Companies adopt hybrid cloud—with at least one private and one shared cloud instance—as well as multi-cloud positions for a multitude of reasons. For some, large capital investments that have yet to reach maturity (i.e., have incurred "technical debt") may dictate a measured, phased approach to cloud adoption. For others, favorable feature sets may compel them to deploy in a multitude of cloud platforms, allowing them to achieve "best in breed" goals. Regulatory and compliance requirements such as GDPR may limit the locations where information may be sourced and stored and hence could be a driver for multi-cloud as well. Personnel may also dictate their adoption strategies—for example, models where development teams have autonomy to choose their preferred frameworks and platforms. Regardless of their reasons, many companies today are approaching the cloud as a broad fabric of services from which they can pick and choose those that best suit their needs, much like a "services buffet."

In this essay, we discuss the primary considerations for hybrid and/or multi-cloud adoption. Going "all in" on public cloud—that is, "cloud first/native"—is a great choice for new or smaller/nimbler organizations, but most, if not all, medium-to-large enterprises are adopting hybrid or multi-cloud strategies.

Hybrid cloud strategies are a combination of on-premises and public cloud. A hybrid cloud approach is typically selected in cases where:

- Risk posture requires that some of your data and workloads stay on-prem
- Enterprises not born in the cloud need a first step to support on-premises systems for an often indeterminate period of time
- Core systems exist that simply cannot be moved to cloud—for example, certain mainframe workloads

Multi-cloud strategies involve multiple cloud providers. Organizations often implement multi-cloud strategies due to:

- "Best of breed" design principles
- Employee experience with varying expertise in one or more clouds
- Availability of a cloud in a specific geographic region
- High availability/disaster recovery: potentially improving reliability and availability of workloads by spanning multiple clouds
- Mandated risk or compliance requirements
- Perceived cost-arbitrage opportunities.[1]

This essay is geared toward Technical Leads within an organization, but executives must remain aware of the complex implications of their broader strategic decisions. The goal is to lay the foundation for hybrid and/or multi-cloud adoption from a technical perspective, outlining the options you must take into consideration before embarking on your cloud journey. Avoiding pitfalls and ensuring you enter your planning phase with "eyes wide open" ultimately leads to a greater chance of success in this venture.

[1] See the pitfall " Fiscal Flexibility at the Cost of Efficacy " on page 36 in Chapter 3.

How

Note that both hybrid and multi-cloud approaches are more complicated than a singular cloud adoption, requiring:

- A centralized identity infrastructure, often involving a third party; you will then need to consider integration options for the various cloud providers and current on-premises platforms
- Complex networking, including potential for dedicated lines of communication to various clouds, potential for Virtual Private Cloud (VPC) peering, routing and firewall rules, security perimeters, scanning, and active protective measures
- Trained staff to manage cost/resource and operate in differing environments

A successful cloud transformation includes meeting your organizational, compliance, and technical goals across the core domains of technology: applications, data, infrastructure, and security. In our experience, we've found that customers who have successfully implemented multi-cloud have done so by focusing on the foundational building blocks listed in Figure 8-1 for each cloud provider.

We discuss each one briefly here.

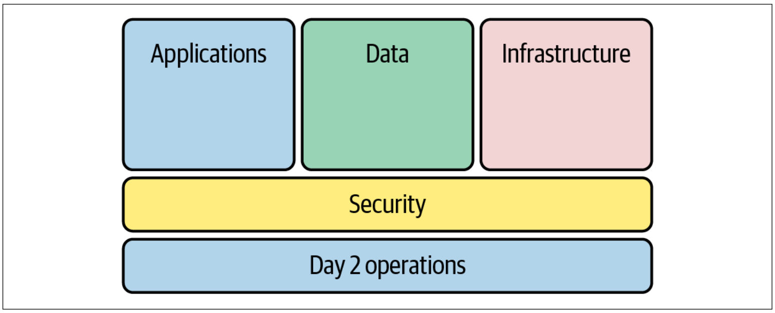

Figure 8-1. Pillars of transformation (adapted from source by Google)

Applications

> *"Not all applications are legacy. Applications become legacy apps when we stop caring for them."*

Initial discovery and rationalization are needed to understand and broadly categorize the application workloads to see which applications are close to cloud ready, cloud native, and legacy. This shapes your decisions for cloud adoption. This may also be a useful exercise to understand your application portfolio and technical debt and to flag duplicate workloads that might be consolidated.

Depending on the application, choose the right compute options for running application workloads—for example, for compute SaaS, PaaS, IaaS. You should prioritize migrating, optimizing, and then modernizing your applications for faster cloud adoption. For new applications, consider 12-factor principles and cloud nativity.[2]

Data

Data is at the heart of your business and cloud adoption; make sure you understand what your data contains, where your data currently resides, and from where it is sourced. Then consider where it will be post–cloud adoption. Do the sources change? If so, how? If not, why? How do you migrate your data to the new location(s)? What are the impacts on the volume and velocity of your data? It's smart to begin with data discovery and perhaps with rationalization as well. You may already have many systems that store the same data. Understand all the data governance needs of your organization. Establish solid data governance policies—access patterns, redaction and data loss prevention strategies, security models, retention and archival strategies, and auditing of access methods, for example. After undertaking this discovery and rationalization, you can move to next steps:

1. Decide which datasets to keep on-prem and which ones to move to the cloud. You can also map the datasets to a cloud provider based on the regulatory, governance, and technology needs.[3]
2. Define your wave1 datasets—low-hanging (low risk but with business value) datasets for cloud migration.

[2] Cloud-native applications are those designed and built specifically to be operated in a vendor cloud environment.

[3] Database options available in GCP are listed at *https://cloud.google.com/products/databases*.

3. Build a plan for longer-term migration. You can also work with your cloud provider to identify the best options available for your needs.
4. Decide which provider to use for long-term data storage from a technology, lock-in, and cost point of view. You may wish to explore Google BigQuery or BigQuery Omni.

Infrastructure

Design and build your infrastructure to minimize cross-cloud calls in order to curtail egress charges, network latency, and so on. Consider all connectivity options between multiple cloud providers that make sense for your organization. Regard network, security, and infrastructure as software.

1. Design and build DevSecOps practices from the start.[4]
2. Consider storage and replication strategies.
3. Adopt an API-first approach, and avoid vendor lock-in.

Security

As you adopt multiple clouds, consider the "four pillars" of information security: authentication, authorization, auditing, and encryption. You likely have multiple existing tools that may or may not be used once you move to cloud providers. Each vendor cloud platform has a suite of capabilities designed specifically for information security. There may be overlap and likely redundancy not only with each cloud provider's offerings but also with your existing on-premises platforms and frameworks. Interoperability must be considered; standards-based open systems allow for simpler and easier adoption compared to closed/proprietary systems.

You may consider some key guiding security principles for your transformation:

- Invest in security from the start; security should be considered for all key design decisions.
- All application interfaces should be designed and built with security first in mind.

[4] See Chapter 9 for more details.

- Data in transit and data at rest should be encrypted, and all access should be secure.
- All access should be audited in logs that can be easily retrieved and analyzed.
- Observe the principle of least privilege/authority for all users.
- Day 2 ops: focus on telemetry data and monitoring for infrastructure and applications.
- Design and test reliability, high availability, and disaster recovery.
- Consider all audit, compliance, and regulatory requirements.
- Infrastructure and applications should be built for resiliency and reliability.
- Design and build infrastructure and applications for observability and measure the performance.

Whether you adopt a single cloud or a hybrid/multi-cloud strategy, step one is understanding where you are today, which involves discovery and rationalization across applications, infrastructure, data, and security areas. Once you have completed initial discovery, carefully consider the cloud options for your organization with the factors that we've discussed in this essay. You can refer to the pitfalls discussed in the next section to understand the drawbacks of implementing multi-cloud. Once you have picked a cloud strategy, you need to plan and execute it. Solutions for moving to Google Cloud are listed at *https://cloud.google.com/solutions*.

Pitfalls

While multi-cloud gives you the flexibility to run your workloads with any provider, drive cost benefits, and provide high resiliency with the ability to use best-in-breed services from each cloud provider, it also comes with a number of drawbacks. With careful evaluation and design practices, you can avoid the following pitfalls.

Interoperability Between Multiple Clouds

This could be a challenge; in some cases, you may have to write additional integration software for your application, which may lead to additional build and operating overheads and a tighter coupling with a cloud vendor.

Lack of a Common Management Plane

The multi-cloud raises many more challenges, such as cloud federation, security, interoperability, quality of service, vendor lock-in, trust, legal issues, monitoring, and billing. Getting a consistent view of billing, services, monitoring, and so on across cloud platforms is not an easy task.

Application and Data Portability Issues

You can avoid vendor lock-in and portability issues by adhering to the open source principles and careful selection of products and services across clouds. However, selection of the right cloud products and services could be challenging, leading to users suffering from the paradox of choice.

Poorly Implemented Security and Privacy

You need to carefully evaluate security, audit, and compliance needs for your organization and ensure that each cloud provider is meeting those requirements. A challenge of the multi-cloud approach is achieving consistent security policies and practices, compliance, and management across different providers.

Inconsistent Day 2 Operations

In a multi-cloud environment, operations will be more complex. In some cases, you may have to build additional systems to monitor and operate systems efficiently in multi-cloud.

CHAPTER 9

Thinking of Cloud Platforms as a Software Problem

Rahul Gupta and Bharath Baiju,
Strategic Cloud Engineers

Who

- Engineers
- Technical Leads

Why

The traditional on-prem infrastructure world is often diversified if viewed from a software standpoint. Almost all the different pieces that constitute the "platform" came from different vendors, making them hard to manage in a consistent way or even orchestrate them as part of a single workflow.

This results in multiple handoffs between different teams, with each team using its respective ticketing system, leading to the deployment and infrastructure provisioning tasks spanning multiple days, sometimes even weeks. This whole cycle repeats itself for every simple piece of software that requires changes to these components.

Cloud platforms being completely API-driven offers a shift from this by enabling users to undertake all operations through automation and to apply software development best practices. This allows you to think of infrastructure, configuration, CI/CD, and delivery as a software problem—a problem that is solved by engineering, a common baseline of tooling, APIs, and best-practices-as-code collectively referred to as "the platform." This platform is built and evolved collectively with application teams to shift left, ship faster, and reduce toil.

There are three main added benefits of this. First, modernizing the traditional IT functions into a Cloud Center of Excellence (CCoE) team helps drive cloud engineering excellence throughout the organization by pushing best practices out from one group to all others. This brings an opportunity to move away from the traditional IT management practices and create a set of common standards, tools, and processes to tackle similar problems in a consistent way. This in turn creates a focused set of functions that can be scaled easily to support the entire organization while empowering self-service, reducing pitfalls due to human accidents, and ensuring compliance. Review Chapter 11 to learn more about continuous compliance and reconciliation.

Second, converting all your ClickOps-type of activities into GitOps-driven activities streamlines your deployment pipelines. *ClickOps* essentially refers to the traditional ticketing- or UI-based workflows that are user driven, whereas *GitOps* is a process-driven approach involving Git-based workflows, where code reviews represent feedback loops and pull request/merge approvals to replace ticket approvals. This creates an opportunity for organizations to design and develop for operating-at-scale by dedicating engineering resources to automate repetitive tasks and making their environments more reproducible and idempotent.

This also enables the platform teams to make fewer mistakes by adopting software best practices such as test-driven development (TDD) for infrastructure and configuration management. As platform teams adopt GitOps, application development teams can programmatically provision and tear down test environments and increase development velocity. In the end, it simplifies future audits to reviewing code changes rather than chasing down irreconcilable processes undertaken by different teams across the organization.

Last, the continuous nature of automation removes configuration creep and leads to considerable cost and time savings. Costly remediation processes can be avoided, as identifying configuration drift is no longer a laborious process, and reconciliation can be automated by observing the infrastructure provisioned with the intended infrastructure expressed as code. API-based provisioning allows for custom logic such as timeboxing for sandbox and test environments, which automatically deprovision infrastructure and thereby save on runaway costs. Using code and cloud platform state as the single source of truth enables the opportunity to deprecate or enhance some IT Infrastructure Library (ITIL) systems and tools. The monthly billing statements and cost projection for the next cycle provided by cloud providers also make it easier to more efficiently regulate the budget.

This essay intends to reflect on the increased struggle faced by software and IT operations teams as they strive to effectively and efficiently manage the rapid scale demanded by the growing businesses:

- Increased demand requiring a rapid, elastic scale of IT infrastructure is solved by using cloud APIs.
- Operational bottlenecks due to rapid scale and fragmentation, both at an organizational and a technical level, are solved by engineering a common baseline of best practices in the form of a platform.
- Disconnected feedback loops between software and IT teams are resolved by adoption of Git-based workflows for both teams.
- Manual errors that lead to compliance and regulatory issues are avoided by adopting automation and using code as the single source of truth.

How

In this section, we will share some techniques for reframing your approach to software so that you can migrate to the cloud. We will also show you some specific examples of tools that can make your migration process smoother, with fewer handoffs between teams.

Prepare for Platform Modernization

Identify exactly what is needed by the organization to successfully move to cloud while incorporating the practices of DevOps. Revisit Chapter 1, keeping software-driven automation in mind.

Consider creating a platform services organization that maintains and operates the “platform.” An existing infrastructure org can also transform into this. This team creates and offers a common set of tools and services that can be leveraged by other parts of the organization, such as application developers, security and compliance, legal, and so on. One example of the services could be a self-service tool that can bootstrap the initial environment setup, using a cookie-cutter routine for quick application onboarding, in conformance with the architectural best practices and standards. Another example could be a centralized build system that provides base images that are already hardened and compliant with organizational policies, so that the application teams no longer need to maintain their own OS images.

Depending on the size and complexity of the platform, this team could also be tasked with maintaining the foundational components, such as identity and access management, billing and chargeback setup, and audit logging, to enable other parts of the organization like the accounting and security teams to perform their respective duties.

Make Decisions on Consistent Tooling

Create a decision map of tools that should be used to perform the individual tasks. The key idea here is to keep consistency in mind as much as possible. For example, using Git as the source code repository might make sense if it is already being used by the software teams. This also reduces the learning curve for the teams. Also, when evaluating new tools, consider the possibility of integrating with existing systems that may have some relevant use cases, like compliance, billing, or account management.

If you plan on leveraging multiple cloud providers, consider choosing tools that provide abstraction across your cloud providers/hybrid cloud. One example of selecting a tool for deploying infrastructure-as-code may be to prefer a tool like Terraform over Google Cloud Deployment Manager or AWS Cloud Formation to be platform independent. Another example is Kubernetes Config Connector (KCC), which uses the Kubernetes Resource Model (KRM) to allow a consistent, cloud-native developer experience. KCC makes use of provisioning controllers that are responsible for asynchronously actuating Google APIs to provision resources and to reconcile drift between observed state and intended state.

Boundaries Defined by Software

Creation of project boundaries and segregation across environments to create logical silos can be automated.[1] For instance, using a project creation template allows segregation via billing, chargebacks, cost centers, and so on to be baked in. Complete environments can be defined that can comprise different components such as Identity Access Management (IAM), folder/project hierarchy, VPCs, security boundary rules, and so on, which significantly improves reproducibility. As environments are defined as code, they can be closely mapped to application releases.

Let the Tools Do All the Heavy Lifting

The creation of a change management process tailored to GitOps optimizes existing processes and integrates with existing change management systems such as ServiceNow, using plug-ins and hooks. Separation of ownership is also enforced in a Git-based model, as product teams may have code ownership over simple product infrastructure code, while a firewall or networking change may require additional PR approvals from InfoSec and Networking teams.

Automated testing workflows consisting of linting, dry runs, and integration tests ensure that the majority of checks are verified before a human gets to review the changes. This may lead to an evolved stage in which some changes are auto-approved by robots based on the severity.

Enforcing that all the changes are made through the process defined above and not manually by humans is critical for maintaining a single source of truth. This can be ensured by restricting access to only view resources in the pipeline-driven environments so that all changes are funneled through the robot accounts used by pipelines and by creating alerts for user actions to ensure that all actions are audited.

Performing more frequent and smaller changes rather than bigger changes over longer gaps makes the changes easier to review, fix, and roll back in case of issues. Creating non-production and production environments that are separate but almost identical to each

[1] See Chapter 7 for more on the topic of project boundaries.

other also allows us to roll out and test changes gradually. Refer to the Google Cloud solution "Managing infrastructure as code with Terraform, Jenkins, and GitOps" for a reference implementation.

Keep Optimizing for Everything

The methodology described in the preceding section can be adopted gradually, allowing organizations to slowly transition to a software-defined automated model. As you begin to realize the benefits of time savings, cost savings, and risk reduction from adopting this model, creating a cost/benefit analysis helps to quantify the value gained and identify additional opportunities to optimize in the near future.

Pitfalls

While modernizing the various IT practices for cloud platforms is a really impactful change, it is worth considering some misconceptions that might bubble up during the transition.

It Will "Cost" More if You Do It Right

Cloud can easily appear to be more expensive if you don't consider the new realities of the cloud platform and how it differs from your existing fleet. For example, consider how the shift from CapEx or OpEx influences your technical decisions, and try to identify the opportunities of scaling up and down with demand.

Although updating IT processes and introducing new tools and practices into the mix may seem to be a costly process, those costs are outweighed by the resulting impact. Trying to optimize everything for cost is not the right approach. There are many hidden costs of time and labor that are sometimes not visible if those things are done otherwise.

But We've Always Used This Tool!

While there may be multiple tools available to do a particular task, and though you may have become comfortable with doing the task in a particular way on-prem, don't do it the same way in the cloud just because of personal comfort. For example, a system config management tool such as Ansible (or Puppet or Chef) is not really the tool of choice when it comes to infrastructure management, whereas a tool like Terraform may be a better option due to its focused approach to creating infrastructure-as-code and the significant vendor contributions that have made it a leading approach to instantiating. Do your research before selecting the right tool. Refer to the article "Deploy a simple website with Terraform and Chef on GCP" to see the differences in action.

Software Engineering Is "Just Writing the Code"

Just writing code for infrastructure is not enough; apply software development life cycle (SDLC) best practices such as requirement analysis, architecture design, test criteria, release planning, version control, and so on. Consider adding upgrade strategies when writing deployment code so that you don't have to manually intervene when it is time to upgrade the resources. Infrastructure changes are mostly linear to software changes and require updating in the same ways, but do not necessarily follow the same pattern.

CHAPTER 10

Making Security Policies Fit-for-Purpose

Omkhar Arasaratnam, Engineering Director

Who

- Engineers
- Business Functions

Why

Deep in the heart of the security department are many tombs of wisdom. On the front covers of these tombs are inscribed the hallowed phrases "ISO 27001," "NIST," and other well-recognized and revered phrases. However, you wouldn't be remiss if you overlooked these tombs, which are under a thick layer of dust that is disturbed only when the security department is called on to "assess your risk," or worse, under the inspection of a regulator. Often in such cases, the interrogating party inflicts the will of policy on the accused (engineering department), demanding "proof of execution," "sample evidence to spot-check," and all "exceptions."

It shouldn't be like this. There has often been a vast separation between the intent of policies as expressed in prose and how distributed systems operate. Long gone are the days in which a spot check of a system's controls can be considered indicative of the system's current state. With many enterprises leveraging hyperscale technologies using consistent control planes like Google Anthos, we have an opportunity to codify prose into software-based controls at build and runtime. In doing so, we can make security policies fit-for-purpose.

How

The most effective security policies are developed hierarchically and have traceability from their lowest levels of technical implementation to their highest levels of principles. In Figure 10-1, we present a hierarchy of security concepts as a guide for transforming principles into practice.

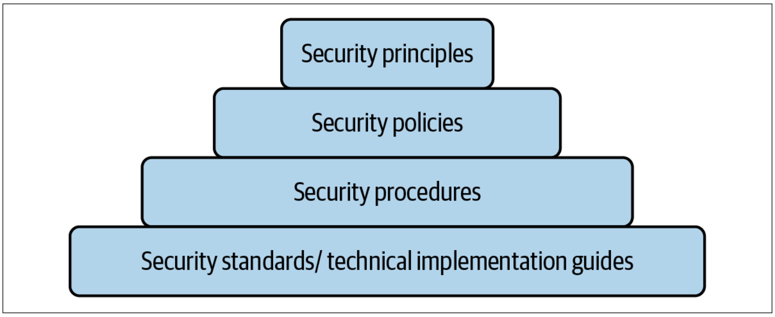

Figure 10-1. A hierarchy of security policies (adapted from Google Original Art)

At their highest levels, security principles provide aspirational statements regarding the intent of the security posture of the organization. Progressing downward, the policies themselves may be logically subdivided into topics of interest or domains. Popular taxonomies for these domains include the NIST Cybersecurity Framework or ISO 27001:2013. These domains can provide guidance regarding particular concepts that are core to security, such as those regarding Identity and Access Management or Asset Management.

Further, security procedures elaborate on the specific sequence of events required to achieve the security controls that are articulated in the policies above. Closely married to the procedures are the security standards and technical implementation guides. These inform the organization of the correct technologies and security configurations required to achieve the desired security outcomes.

Also bear in mind that the procedures and standards need to be oriented around both build time and runtime control requirements.

Where possible, security requirements, considerations, controls, and evaluation should occur early within your SDLC, and consistently throughout. At runtime, your system design should allow for visibility into the performance of the controls and validation against new security threats that are discovered. As with other forms of testing, testing positive and negative outcomes is key, as is the consideration of advanced techniques such as mutation testing.

Each step of validation, whether at build or at runtime, should be preserved as an artifact with both traceability and lineage. This provides the organization traceability from the ephemeral prose of principles down to actual runtime build artifacts.

Where possible, using policy-based methods of validating technical conformance such as Forseti is key to being able to secure environments at scale.[1] Attempting to hard code security controls within configurations or code leads to brittle systems, which are slow to react.

Consider the following example.

Principles

A security principle might state, "All systems will maintain Confidentiality, Integrity, and Availability." Principles should rarely if ever change.

Security Policies and Procedures

A correlated *security policy and procedure*, focused on access control, least privilege, could offer more specific guidance, such as topics found within NIST 800-53 Access Control:

Policy: Privileged Accounts

Privileged accounts, including super-user accounts, are typically described as system administrators for various types of commercial, off-the-shelf operating systems. Restricting privileged accounts to specific personnel or roles prevents day-to-day users from having access to privileged information/functions. Organizations may differentiate in the application of this

[1] Forseti is a continuous-compliance tool for cloud platforms. See Chapter 11 for more details on this approach.

> control enhancement between allowed privileges for local accounts and allowed privileges for domain accounts, provided the organizations retain the ability to control information system configurations for key security parameters and as otherwise necessary to sufficiently mitigate risk.

It may also describe processes by which to obtain, in this case, a privileged account—for example:

Prerequisites

- Only people within production support may apply.
- Only people who are already in the unprivileged group for the system may apply.
- An enhanced HR background check is required.

Procedure

1. Request access.
2. Manager approval.
3. System owner approval.

Conflict check: Requester should not grant manager approval or system owner approval. If either condition is met, route to CIO for approval.

A complementary procedure may be a detective control to determine whether the expected behavior occurred in production. This may be a process to handle exceptions to the expected standard above—for example:

> Exceptions: CIO approval required.

These processes should be relatively static, changing only in tandem with major organizational changes.

Security Standards/Technical Implementation Guides

An *implementation guide* is the technical or solution-specific guidance that helps technical personnel in implementing the relevant technical controls for a particular application or technology. In the case of this example, it might provide guidance for configuring your request management system to route the appropriate approvals for privileged access. Or it may contain instructions for how to configure the Google Cloud Platform IAM Roles and Policies to achieve the prescribed outcome.

Furthermore, standards and implementation guides may provide correlation and input into advanced analytics platforms such as Chronicle Security Analytics Platform to design alerts that inform the detective processes.

These implementation guides/standards may change frequently, as versions of the API/software change.

Conclusion

As technology evolves, the principles and process provide guidance as to the underlying systems that shall be configured. This ultimately allows consistency of intent and policy across domains. There are evolving technologies such as Open Policy Agent (OPA) that seek to homogenize the translation of intent to technical implementation.

Pitfalls

Revisiting these sacred policies is a crucial part of your cultural transformation. It signals a desire for an understanding of how you will operate differently. However, there are pitfalls.

Moving at the Wrong Speed

Often, to meet the illusion of speed, organizations sacrifice the aspects of policy design that are viewed as more academic. While some of this preamble may seem laborious, properly planning security policies from the highest principles allows for clear, concise, and, most importantly, consistent implementation in technology.

From the perspective of risk, compliance, and regulatory organizations, it also provides a clean and linear traceability from intent to implementation. Furthermore, when done correctly, it provides a method for organizations to prove continuous compliance to control objectives with minimal effort.

Internal Adversarial Behaviors

Too often, there's been a significant gap between the well-meaning intent of risk professionals and the distributed systems created by engineers. By taking a policy-based approach, organizations can ensure they meet their risk obligations in code throughout the SDLC as a matter of continuous compliance, rather than approaching security as a one-time, adversarial interrogation that is trying to match eloquently written policies to the underlying technical implementation.

At one company, this was especially pronounced. Historically, the development team had envisaged the security team's accountability as, to put it generously, "box tickers." Conversely, the security team had personified the developers as "unruly cowboys."

The constant strife led to a significant reticence to comply with security requirements. This resulted in a very adversarial relationship between the two teams, as well as constant deprioritization of security requirements. These lapses ultimately led to a Matter Requiring Attention (MRA) by the regulatory authorities that supervised the company.

The security and engineering teams used this as a change agent to revisit their engagement. The security team started at the top, defining the organizational principles and policies. The security team then cooperated with the engineering team to develop the procedures, standards, and technical implementation guides, leveraging the principles and policies as control requirements. By doing this, the engineering team was able to articulate the most feasible manner in which it could technically achieve a control outcome, without feeling like it was being subjected to bureaucratic inquiry. Conversely, the resulting mappings allowed the security team to better understand critical controls and its influence on the overall security posture.

The company ended up closing the MRA in a timely, sustainable manner, and the security and engineering teams enjoyed a more constructive relationship going forward.

CHAPTER 11
360-Degree Continuous Compliance

Kieran Broadfoot, SRE Director

Who

- Engineers

Why

Developers are naturally curious and are therefore likely to have existing experience in your chosen cloud platform. Harnessing their enthusiasm and passion is absolutely critical to driving cultural transformation within your organization. This enthusiasm, however, can create tension, as developers are likely to request broad access to a wide variety of services without significant enterprise control. This leaves us with a challenge: how do we apply our security policies in a consistent and reliable fashion?

A common anti-pattern we've seen regularly is to build or buy a brokerage layer—a piece of software that abstracts one or more cloud platforms into a common set of APIs. This is regularly framed as solving two problems:

1. Enabling longer-term arbitrage opportunities between cloud providers
2. Providing a common control point at which policy can be applied

The first problem is dealt with elsewhere in this report, where we argue that enabling only common services substantially reduces cost-efficiency and velocity in favor of a future *potential* opportunity.

This approach of wrapping the providers' APIs also has a dangerous side effect, whereby you substantially change the developer experience (DX). Any knowledge your developers have already garnered in operating cloud technologies will be underutilized. Moreover, it unintentionally stymies growth for your engineers beyond their current role and requires you to create a range of new custom training. This training needs to cover the use of your new facade but also needs to reflect the range of architectural patterns/blueprints you consider "safe." We believe that leveraging best practices and patterns already published by the platform vendors, industry groups, or the open source community generates sublinear benefits.

A better approach would be to utilize common application frameworks (such as Anthos), which enable higher-order application architectures across multiple cloud platforms (see Chapter 8).

However, the second argument for a brokerage is particularly pertinent, as it starts with the implicit assumption that control is applied only once, at the point of provisioning. We would argue that this assumption is false.

First, it presumes that you have a watertight control regime where, over time, configuration creep cannot enter the environment. If this were the case, you likely have built a platform that is not suited for innovation and business change. We'd recommend reviewing Chapter 10 to learn how we think about defining your lines of defense.

Second, it fails to recognize that security is an ongoing, continuous process built into every stage of the software life cycle.[1]

[1] You can learn more about "shifting left" in "DevOps tech: Shifting left on security".

Third, it presumes that any additional layers of defense have a linear cost profile—that is, every new infrastructure component has an associated nonzero cost required to secure it over time. What would this look like if the cost to secure any component was (near-)zero?

This is what we mean by continuous compliance. In other words, we need to find a way to make the cost of applying policy trend toward zero over an extended period of time. Once you reach this point, you can apply control constantly and consistently—a fact that your business, regulators, and partners will appreciate.

How

In an environment where you can monitor *everything* and react to *anything*, your policies can now be formalized and continuously applied. But what might you monitor, and how would you do so?

Our recommendation is to utilize software agents that are built to constantly monitor and react to changes within the environment. These agents can be built to handle a multitude of situations:

- Removing insecure configurations (example: block storage created without using a customer-managed key)
- Driving cost efficiency (example: halting VMs running in Dev projects outside of business hours)
- Providing architectural insights (example: choosing MySQL when Datastore is the preferred choice)

Building these types of software agents is achievable by virtue of three key cloud architectural promises:

1. Cloud platforms provide a consistent and timely log of all state-changes occurring within your environments.
2. Wiring/routing log streams to compute resources is a trivial endeavor.
3. Serverless architectures enable highly scalable models for responding to stream events.

For a practical example of this continuous compliance architecture, let's look at a contrived scenario. Consider an enterprise that wishes to understand where *personally identifiable information* (PII) data is being stored. To achieve this, they have decided to enforce labeling on every persistent disk.[2] Their first line of defense would be to apply control via infrastructure-as-code[3] or API policy applied via Resource Manager. In both cases, this ensures the correct labeling at the point of creation.

However, we can also apply a second line of defense by ensuring that the label is always present, even if an employee intentionally removes it. This automated application of policy makes certain that the organization remains in compliance with its security requirements. However, as with any software development effort, edge cases must be considered. For example, what should happen if the label is removed rather than altered? In this trivial example, we default to a "fail-safe" mode by presuming a worst-case scenario and applying a PII label. Clearly, this naive solution produces false-positive outcomes in some situations.

Here are the approximate steps on GCP to ensure the label is always present:[4]

1. Manually review the audit logs that are available and make sure you understand how they represent state changes.
2. Create a new pub/sub topic, such as `audit_log_responder`.
3. Set up a log sink (`audit_log_sink`) to post audit logs to the `audit_log_responder` pub/sub topic:[5]
    a. Filter the relevant audit logs that you need: (e.g., `resource.type="gce_disk" severity="NOTICE"`).
    b. Set the destination as a pub/sub topic: `audit_log_responder`.
4. Create a new Google Cloud Function (`audit_log_disk_labelling_check`) in your preferred region, using the `audit_log_responder` topic as a trigger. See the sample code in the Appendix.
5. Test: creating a new, persistent disk without any labels causes the function to execute and apply/fix if missing. The same

[2] `label: "datatype", values: [ pii | no_pii ]`

[3] Examples include Google Config Validator or Hashicorp Sentinel.

[4] This model is applicable to other cloud platform vendors with minimal changes.

[5] Bonus step: consider setting up a second log sink with a target of a locked GCP storage bucket for long-term retention and review.

occurs if the labels are removed or changed at any time in the future.

The preceding example intentionally operates from first principles to highlight how this type of continuous compliance works. However, a variety of platforms are available that provide much of the scaffolding needed to build your own policy model, including Forseti, Cloud Custodian, and so on.

There is one final area of opportunity that results from this mode of operation—policy as code. Chapter 8 spoke about the requirement to revisit security policy to ensure it is fit-for-purpose. However, it is feasible to extend this idea one step further by documenting your policy as code, or as pseudocode, depending on the current skill set of your organization. Articulating your policy in this way removes ambiguity and ensures that edge cases are given due consideration by subject-matter experts.

Pitfalls

Building a continuous compliance framework can be extraordinarily powerful, but there are a few risks worth considering.

Doing Something Just Because the Cloud Makes It Possible

Introducing new policies is cheap and easy, but the complexity of interactions among policies grows quickly. Without careful review and understanding of these interactions, it's possible to create problems that are difficult to debug and that may lead to unintended consequences, such as a reduction in velocity. Ensure you maintain a review process for new policies, to confirm they are fair, reasonable, and aligned to the organization's overall goals.

Avoiding Teachable Moments

The example in this essay automatically and continuously actuates its policy. However, it does not provide any mechanism to explain "why" it did what it did. Consider ways to expose the logic of these policies to those impacted by them, so that your organization learns from its mistakes.

Hasty Continuous Compliance Architecture

At scale, executing a Google Cloud Function (GCF) in response to every log entry may be cost prohibitive, although doing so is likely cheaper than the business impact of not running a continuous compliance architecture. Think carefully about your log sinks and logging regime (filtering) to find the balance between cost and value. It may also be appropriate to host your GCF policy functions in a distinct project, using more advanced IAM and log routing to improve separation of concerns within your cloud environment.

CHAPTER 12
Safely Operating at Scale

Stephen Thorne, Staff Site Reliability Engineer

Who

- Engineers
- Enabling Team Leads
- Technical Leads

Why

Once you have an application running on a cloud platform that your users care about, you need to consider its reliability. After all, it doesn't matter how good it is if your users can't depend on it. This is why we advocate a user-centric approach—will your customers care that the product has exceptional design and great features if they can't trust that it works?

At Google, we consider reliability an engineering discipline equal in importance to software engineering itself. This work, however, is typically undertaken by engineers who demonstrate a particular proclivity toward large-scale architectural and systems thinking. Furthermore, we intentionally think of all these aspects of engineering as an intrinsic cost of delivering our services, and therefore they are intentionally long-running. This regular cadence of improvement and reduction of technical debt ensures the longevity of our services. We believe this is a critical element of any cloud transformation journey.

How

Once you decide that you must care about the reliability of a specific service, measuring and assessing the user experience of that service is the first critical step you should take. We know there are a myriad of ways you can quantify reliability. On the web, the simplest starting point is to measure the error rate and latency as seen by your users.

The fundamental requirement is to measure the experience your users actually perceive. Your users do not care about your CPU utilization on your database, or if a batch job takes 20% longer. They care whether their applications load quickly and whether their app shows the right information when they need it.

The measurement of user-perceived reliability is something we call a *service level indicator*, or SLI. What you want to do as a business is make sure that this SLI doesn't dip so low that a user notices. So you put a threshold on it—we call this threshold the *service level objective*, or SLO. A good SLO might be that over a 28-day period, 99.9% of your customer's requests respond with an HTTP 200 OK in under 200ms.

To be clear, your SLO is not a binding service level agreement—in other words, when it's compromised, you don't give your customers their money back. Rather, the SLO is intended to enable your business to balance reliability against feature velocity and to set objectives and milestones for your engineering teams. Targeting your SLO at a point below 100% means you have a budget for small service disruptions. Your aim is to keep the feature development and service running but respond to problems when the reliability dips below its objective.

Once you have your SLOs in place, you have to start doing the hard work: discovering what the biggest danger to the system is, in objective terms. Only experience with your own service and setup can tell you what is actually your biggest concern.

Learning from your experiences is extremely necessary. When there is a large incident—such as that time you ran out of memory on your database server and no engineer noticed until the customers reported the site was down, or that time you pushed a new release and the payment gateway treated every transaction as fraud—you need to learn from the ordeal.

When we have an incident, we write a retrospective document we call a *blameless postmortem*. This postmortem aims to explain what happened, how it affected our customers, what went right, what went wrong, where we got lucky, and what we want to do next time to stop it from being as bad, if it ever occurs again. As we've discussed elsewhere in this report, mistakes are inevitable; how you respond to them is a quintessential part of your cultural transformation. Each action undertaken during the operational failure must be considered appropriate given the current situational knowledge. Therefore, a postmortem is a teachable moment for the organization.

The actions we might take are to *detect*, *mitigate*, or *prevent* future outages. In a hypothetical situation in which a database server has exhausted memory, we might:

- Improve detection by adding monitoring to the database server for a high memory threshold or probe the server to make sure it's running fast
- Improve mitigation by giving incident responders better tools for reducing traffic to the database server or increasing memory to any of the servers quickly
- Implement some form of automated prevention, such as automatically provisioning a higher memory machine when required or adjusting our load balancer to not send queries to a server replica that is overloaded

As you can see, some of these are easy, and some are hard. The goal is not to implement them all but to choose what to do so that next time we have fewer errors that our customers see, and we spend less of our error budget.

You can learn more about the SRE mindset, including topics such as SLOs, toil, automation, release/risk management, and monitoring, from our freely available and widely read books at *sre.google/books*.

Pitfalls

You can avoid these common anti-patterns in order to operate safely at scale and to provide your organization room to scale up easily.

Require Reliability

You can imagine that if you require a team to keep the product reliable, they will do the simplest and easiest thing: change the way the system is measured until the numbers meet management expectations.

You must incentivize the team to report where the customer experience is deficient and to do the engineering work required to give your users the experience they deserve. In this way, you will get more honest monitoring over time, and your team will know that it is never a problem to show where the system was less reliable than everyone thought it was. It will give them an opportunity to do work they will be rewarded for.

Blame

Be careful when writing any kind of report on failures. A good incident retrospective is a true and frank accounting of what happened. Your staff must be given enough psychological safety to be able to write down every contributing factor, so that you can work out how to make sure the best lessons are learned and the best protections are put in place.

With this in mind, you must never blame one person or system. There is no such thing as a "root cause"—you must consider every contributing factor the failure. If you blame one person, feature, service, or bug, fixating on it is very easy. You must not seek to blame anyone or anything but instead must consider everything that led to the failure. Only then will you be able to assess the best ways to make sure the failure never happens again.

CHAPTER 13

Developers—Go Faster, Go Together

Titus Winters, C++ Libraries Lead

Editor's note: In the introduction, we promised that we would refrain from preaching the value of cloud. Well, what's a rule without a willingness to break it once in a while? This final essay takes a different approach: it looks different and has a different structure. This essay is for developers everywhere, reminding us that once in a while, a technical change may, in and of itself, be an extremely powerful motivator for unleashing the latent talent in your organization.
With love, Google.

Disruption Arrives: Let's Move to Cloud!

Recommendation: *don't underestimate how much a change in resource provisioning can impact your day-to-day.*

It's a cliche in the tech industry: your project is humming along, everything seems to be normal—and then the Monday morning meeting happens. Management has discovered a new truth, the perfect new tech strategy. Today's shift: *we're moving to cloud!* You sigh and mentally throw out the plans you had for the next few quarters. Disruption has arrived.

Good engineering leaders know that there is some truth to this—disruptive new mandates have a cost. The tech industry is certainly prone to fads and buzzwords, so you have to think very carefully when a new fad arrives on the scene: is this really going to improve outcomes for you? In the case of shifting to cloud, we believe there is a lot of benefit—for your project, your team, and even your happiness as a developer.

Some Benefits Appear Quickly

Recommendation: *use cloud and on-demand resources for development as well as production.*

Once your organization starts using on-demand and scaled resource availability for production, it's easy to see the bottom-line benefit: you pay for what you need, when you need it. No more lag time to get a new server in the mix. With an eventual move to "cloud native," your compute resources become cattle, not pets; it isn't "the server"—it's just interchangeable resources that can be swapped in at will. Kubernetes deployments, serverless designs, and cloud-based storage mean that no single machine matters; it's all a haze of resources. The industry has been talking about this for decades. "Utility computing" and "the network is the computer" foreshadowed this, even if the paradigm shift to cloud took a while to manifest.

Still, make no mistake: this is a paradigm shift, and the extent to which that paradigm will change things is still unintuitive for most of us. For you as a developer, this change will undoubtedly come with new things to learn but also with some great improvements to the developer experience. For instance, cloud adoption in the developer workflow may lead to great boosts in productivity and perhaps team harmony. Consider: in many development shops, there is an "elephant in the room" question: "Does the build take too long?" The math here seems simple: if an average developer costs the company, say, $100,000 a year, a 3% productivity boost justifies buying a big expensive developer machine for $3,000, and that productivity boost pays for itself quickly. There are many professional environments in which more than 50% of the workday is spent waiting on the build...and in cases like those, minor improvements to build speed can make a big difference to productivity. Just as cloud makes sense for production, a build server or build farm is even more effective than one expensive machine for each developer. For development as much as production, you want pooled resources for efficiency.

More Benefits Accrue over the Long Term

Recommendation: *shared build resources and artifact-based build systems improve consistency, communication, and productivity.*

Once your team starts thinking about shared resources for the build, you probably start with your continuous integration (CI) systems like Travis or Jenkins. Even this small step is an improvement for many teams: with a centralized source-of-truth for the build and tests, the argument "It worked on my machine!" can finally fade away. Once the build runs reliably with your cloud resources, a forward-thinking team may start to wonder, "Why don't our local builds match the results from the CI build?" This may cause you to adopt an artifact-based build system such as Bazel. A good build system and distributed build machinery (whether on-prem or in the cloud) can feel like magic: no more habitual "make clean," because you can actually trust the build. Incremental builds are nearly instant. Large, complex builds with hundreds or thousands of dependencies don't matter. Hermeticity means no more "It worked on my machine!" The same build resources and isolation can even be used to run your unit tests.

With shared build infrastructure and CI, the question of "Is the build green?" stops being so critical. You don't have to hunt around to find a commit to the repo that happens to work; you can sync up and get a working version most of the time. CI systems give your whole team a shared ground-truth awareness of how the repo is doing. If you're tired of that dev down the hall breaking the build before going for lunch, shared infrastructure can give your team a shared understanding and a real chance to improve communication and collaboration. Investing fully in this modernization will get you lower build latencies, more reliable builds, caching, parallelization—improvements that mean you and your team can spend more time developing productively.

Shifting your production workloads to cloud may be the impetus that gets management thinking about the concrete tradeoffs involved in development: how much you're paying for compute resources, how much you're paying for developers, and how much time those devs are able to be productive. Compute resources take away arguments like "I didn't run the tests because they take too long on my machine." Shared infrastructure takes away lots of engineering team friction. And that is the real personal appeal in cloud:

instead of everyone being silently off working on their own machine, looking at a local copy of things and hacking on an isolated version of some component, a shift to cloud makes it much more immediate and obvious that your team is working in a shared environment with a shared fate.

The Transformation Benefits the Organization and the Technology

Recommendation: *shared resources and understanding will make you and your team more effective.*

If you want to go fast, go alone. If you want to go far, go together. When everyone is focused on having one powerful machine under their desk, they are shackled to that private pool of resources and will inherently gravitate toward isolated practices. The industry has seen what happens in that world: we get isolated "hero" developers. But most of us aren't heroes and don't want to resort to heroics just to get our jobs done. When you are unshackled from that local, isolated thinking, then your team can really shine. You'll get more done, you'll collaborate more, you'll learn more, you can be better at your job—but it starts with a disruption. There's a new way to do things—some of it will be chaotic, and everyone has to think it through. In the end, we'll find new and better ways of developing software.

In other words, our message is clear: helping your organization navigate the complexity of cultural change also provides a clear opportunity to bootstrap new practices within your own development teams. Doing one without the other is a fool's errand.

Conclusion

Kieran Broadfoot, SRE Director

There is no doubt that successful cloud transformation is difficult—extremely difficult, even. Therefore, it was clear that helping you take the same transformational leap should be a priority for us. Providing guidance to other organizations as they seek to achieve sublinear growth is recognized as a key moment in our careers. We fundamentally believe that cloud enables digital transformation and generates outsize benefits.

Every organization is inherently a society with its own norms, customs, and values. Therefore, any technical transformation is entirely predicated on understanding—and, where necessary, changing—this culture.

As we've mentioned numerous times, this type of cultural transformation is challenging insomuch as it requires significant introspection. It requires an organization to honestly reflect on who it is, what it wants to be, and how it wants to operate going forward. This is a blameless endeavor that demands uncomfortable conversations at every level of the company.

Here's the crux of the matter: transformation generates tension, and tension is inevitably challenging for everyone involved. It's often seen as creating "winners" and "losers." It frames individuals as "old" or "new." It doesn't have to be this way, however.

That's why we wrote this treatise on the philosophy of change. We cannot know your unique situation, but we can ask the challenging questions to help you reflect on your own journey. We genuinely hope we've helped in this regard.

Throughout our essays, we've shared a number of key tenets that underpin the change you need to instill in your own organization. Distilling these rich thoughts down to a small number of clear actions is challenging, but we hope you use the following list as your call to action:

1. Understand and document the key constituents necessary for your transformation.
2. Create a psychologically safe culture in which you can grow together.
3. Define clear objectives for the organization. Document measurable steps toward these goals and understand that each step must, in and of itself, deliver value.
4. Review your existing organizational behaviors and set principles/policies that influence and direct every future decision related to your transformation.
5. Use your new culture to refine how decisions are made and provide meaningful autonomy across the organization.
6. Build structures that empower practitioners to share best practices and solve common problems. Use these structures to accelerate others.
7. Build guardrails into your cloud platform that support transformation, at pace, without negatively impacting others. Support safe experimentation.
8. Understand the types of cloud platforms that are the most appropriate fit for your business needs. Choose carefully.
9. Recognize that *everything* is now software, and understand what this means for your existing IT infrastructure functions.
10. Don't be afraid to revisit existing, hallowed security policies. Making them fit-for-purpose is crucial.
11. Continuously measure and apply your new security policies through software.

12. Be bold; build a new way of operating your business products with a customer-centric perspective.
13. Love your developers.

Good luck!

APPENDIX

Sample Code

Sample Code

This code uses Google Cloud APIs to illustrate how you can ensure labeling on every persistent disk created for your project, applying principles from Chapter 11.

```
import googleapiclient.discovery
import logging
import base64
import json

def set_datatype_label_as_pii(client, labels,
    fingerprint, project, zone, resource):
    # update current label set to include our specific key/value pair
    labels["datatype"] = "pii"
    # build request body for setLabels API
    label={
        "labels": labels,
        "labelFingerprint": fingerprint
    }
    logging.warn("Adding datatype label to disk: "+resource)
    request = client.disks().setLabels(project=project, zone=zone,
        resource=resource, body=label)
    response = request.execute()

def check_disk_label(event, context):
    """
    On receipt of an GCE Disk audit log fragment (received via log
    sink + pub/sub), this function applies continuous compliance
    by confirming a "datatype" label has been applied with the
    value of "pii" or "no_pii".
```

```
"""
body = json.loads(base64.b64decode(event['data']).decode('utf-8'))
if "resource" in body:
    disk_id = body["resource"]["labels"]["disk_id"]
    zone = body["resource"]["labels"]["zone"]
    project = body["resource"]["labels"]["project_id"]
    # build API to communicate with GCE
    service = googleapiclient.discovery.build('compute', 'v1')
    # request full disk information, including labels
    disk = service.disks().get(project=project, zone=zone,
        disk=disk_id).execute()
    # save labelFingerprint as we will need to provide them it
        to the API if we update the labels
    fingerprint = disk["labelFingerprint"]
    # check labels are attached to this disk
    if "labels" in disk:
        # setting labels is an atomic update, so save existing
          labels for updating before returning
        current_labels = disk["labels"]
        # check for the existence of our datatype label
        if "datatype" in current_labels
        and current_labels["datatype"] not
            in ["pii", "no_pii"]:
            # key found but value is not in valid set
                - reset to PII
            set_datatype_label_as_pii(service, current_labels,
                fingerprint, project, zone, disk_id)
    else:
        # no labels found - add and set to PII
        set_datatype_label_as_pii(service, {},
            fingerprint, project, zone, disk_id)
```