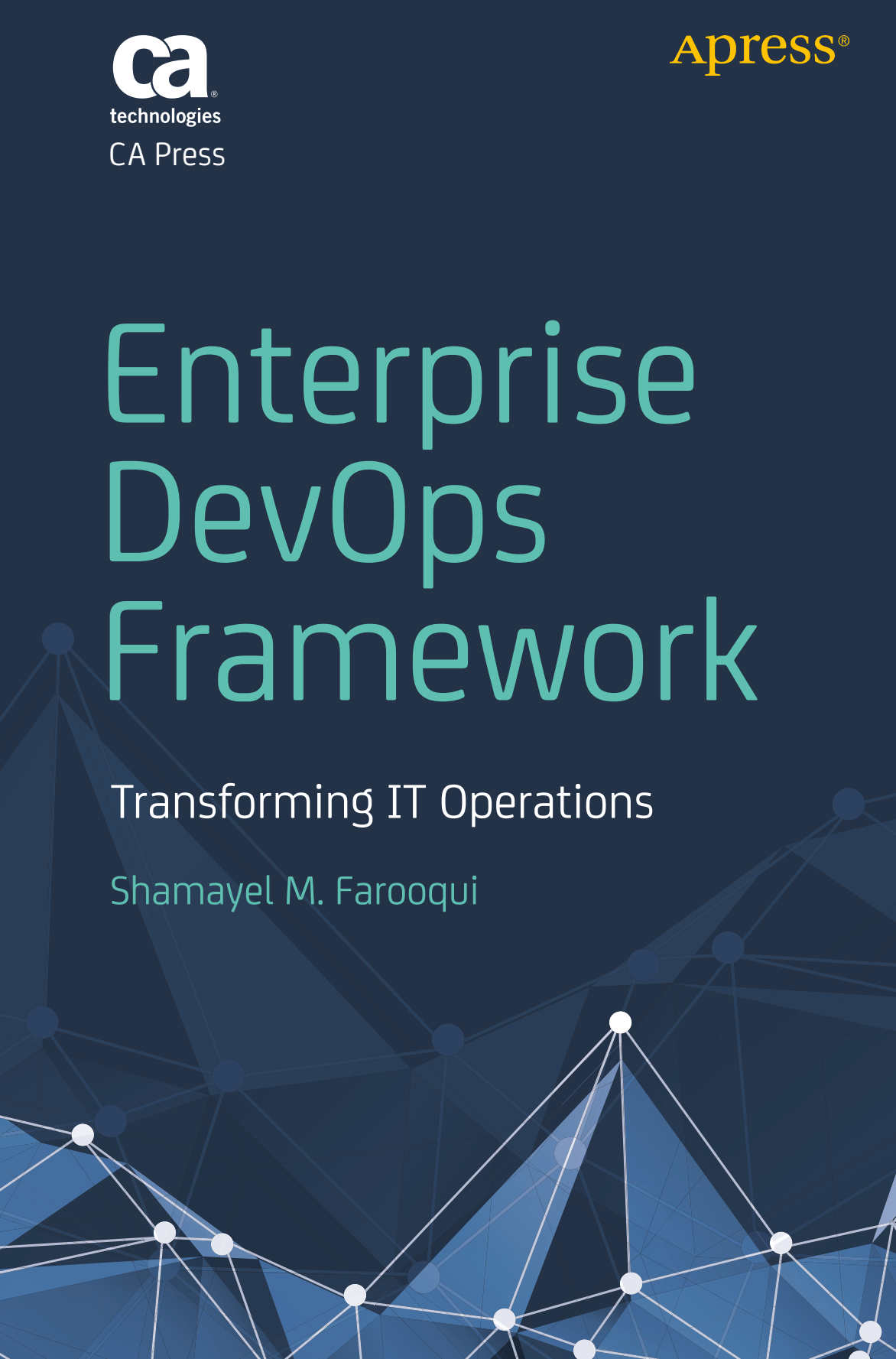
ca technologies
CA Press

Apress®

Enterprise DevOps Framework

Transforming IT Operations

Shamayel M. Farooqui



ENTERPRISE DEVOPS FRAMEWORK

TRANSFORMING IT OPERATIONS

Shamayel M. Farooqui

CA Press

Apress®

Enterprise DevOps Framework: Transforming IT Operations

Shamayel M. Farooqui
Banjara Hills, Telangana, India

ISBN-13 (pbk): 978-1-4842-3611-6 ISBN-13 (electronic): 978-1-4842-3612-3
https://doi.org/10.1007/978-1-4842-3612-3

Library of Congress Control Number: 2018951263

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Susan McDermott
Development Editor: Laura Berendson
Coordinating Editor: Rita Fernando

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484236116. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

I would like to dedicate this book to

My loving dad and mom, Dr. M. Jowad Farooqui and Mrs. Zeba Farzana, for making me the person I am today and for always making me believe that I am special.

My dear wife Zeeshan and my sons, Ashaz and Zidan, for always giving me a reason to smile.

My paternal and maternal grandfathers, M. Hafeez Farooqui and Mohammed Abul Hasan, for showing me how to lead a life for which you will be remembered for a long time after you are gone.

Contents

About the Author

Shamayel M. Farooqui is a serial innovator who thrives on building creative solutions to solve complex IT problems. He has successfully led IT teams that specialize in enabling DevOps within an organization by using the power of automation. The teams he has managed have been responsible for developing automation for business process automation and Cloud management, including application migration on public Cloud. He designed an innovation framework to simplify the process of driving innovation in an enterprise and has been at the helm of introducing a disruptive approach for modernizing IT processes and to ensure business is optimized by using these practices. He is well versed with Cloud technology and is at the forefront of the adoption of IaaS and PaaS within his organization.

Shamayel has also designed and architected the automation framework for implementing process automation solutions. He has delivered process automation as a service for his organization, which was used to drive multiple optimization initiatives successfully to reduce operational costs and risks for IT. He has been a champion for running lean IT and has played the role of a mentor by coaching many IT engineers on automation and innovation practices.

Shamayel has a master's degree in computer science from Loyola University, Chicago, and a certification from MIT's Sloan Executive Education, titled "Innovation of Products and Services: MIT's Approach to Design Thinking."

About the Technical Reviewer

Raghu Ram Burra has 22+ years of experience in the software technology, software services, and solution delivery industry. Raghu has led large global multicultural teams to deliver innovative technology solutions using agile management processes and tools. He has led enterprise digital transformation initiatives at large global enterprises that included Omni channel digital experience, API/PaaS solutions, digital modernization using micro services, DevOps transformation, in-memory computing, Big Data Analytics, AI, and robotic automation.

In his current role at CA as Senior Director, Global Information Systems, he is responsible for the public and private Cloud offerings including CA Dev Cloud (IaaS and PaaS Offerings) and CA AWS as a Service (AWSaaS) and partners with CA's product development teams to maximize the value from the applications/solutions used by our product teams, including operationalizing new PaaS and SaaS offerings that are foundational to help support CA's digital transformation.

Raghu received his M.S. in computer science from the University of Cincinnati, OH, and a Bachelors in Engineering from BIT, Mesra, India.

Foreword

It's easy and comfortable doing things the way they've always been done. However, not creating impactful change through continuous improvement often creates decline in performance and output in any organization. Over time, doing things the same way decreases engagement and creates errors that erode credibility and reduce productivity caused by repetition-induced lack of focus. In this reality, how do organizations that left to their own devices mostly prefer to operate in the clichéd modes of "why fix it when it's not broken" or to take a "not-invented-here approach" to create meaningful change. Evolving to a DevOps culture is a material change in how people in teams think, interact, and deliver outcomes. It's the evolution to a culture that is required for success.

Creating a culture that allows an organization to respond effectively to changing business needs while not making the journey onerous for the people who make up the organization is the goal of every leader. The vision of transformational change is materially enabled by the belief and progressive mindset of change agents who bring change alive and make outcomes better. Grit, commitment, and resilience are the core attributes of these individuals who make it happen. A DevOps transformation is brought about by a change in thinking and practices of team members and accelerated by the leadership of key individuals who embrace the vision and concepts. When this happens, the outcomes are better than what people thought could be accomplished at the beginning of the journey.

This book by Shamayel provides perspectives and lessons that are valuable in multiple scenarios and fit multiple personas.

- If the reader is someone responsible for creating the vision and delivering outcomes, the book will provide insights into what good looks like and into provide headlights into potential pitfalls they may experience in the transformation journey.
- For someone who is in the middle of a DevOps transformation, this book will provide examples that will validate and provide solutions to challenges they may currently be facing.

- If the reader is looking to be that change agent, it'll provide perspectives on approaches they could adopt to influence outcomes across and above from a organizational hierarchy view.

Using a golf tournament metaphor, this book provides an "inside-the-ropes" view into how influencing people and improving how work gets done leads to better team engagement, leading to strong business outcomes.

Mahendra Durai

Senior Vice President - IT, CA Technologies

Acknowledgments

My heartfelt gratitude to my colleagues and mentors Uday Bikkasani and Mahendra Durai for their invaluable input and support throughout the journey of writing this book.

Thank you, Raghu Ram Burra, for playing your part as the technical reviewer for this book and ensuring the book stays relevant to the current and future of IT.

Thank you, Apress team; you have made it possible for me to publish this book with all the guidance and support you provided.

I would also like to express my indebtedness to my alma mater, Little Flower High School, Hyderabad. The teaching staff at this prestigious institution has played a key role in laying the foundation and instilling the confidence in me to aspire for excellence in all my endeavors.

A big shout-out to my friend and ex-colleague, Archana Suresh, who spent long hours coediting this book during the initial stages to help me get started on the right foot.

Introduction

The IT industry in Its Current State

In today's world, a number of companies are focusing on optimizing their IT operations. Access to all forms of data, the ability to perform meaningful analytics on this data, and collaboration across different teams to make use of these analytics have become critical for running a successful IT shop. The IT services and the platforms on which these services are hosted have drastically evolved in the last decade with the adoption and enhancement of technologies like Cloud and containers.

How can an IT company stay relevant in this tsunami of changes that continue to hit the market? Although the new technologies definitely bring better features, it is no easy task to stay abreast while adopting them and get real benefits out of them. And this problem gets further magnified if you are an enterprise whose IT works the traditional way, running huge monolithic legacy apps supported by multiple processes on traditional hosting platforms, which are quickly becoming outdated.

There are a number of companies trying to upgrade their technology centers in order to brace themselves for the disruptions of the modern-day IT world. This is being done for many reasons, like reducing the cost of hosting and supporting their applications or business services, enabling efficient collaboration between globally distributed teams, to function with lean team structures and to be able to provide a good user experience.

This often means that these companies need to embrace modern practices of DevOps and Agile frameworks and strengthen these practices with the latest tools and technologies. By adopting this approach, they are able to convert the "good old admin" job into a more dynamic and modern-day "Reliability Engineer" who has a holistic view and understanding of how the different layers in IT are connected with each other. This in turn helps them deliver IT services with more speed and accuracy and improve upon the quality of the services they offer.

How Traditional IT Functions and How the Imbalance Has Crept In

Without doubt in any IT organization, there are umpteen number of business processes that are a part of the day-to-day functioning. The range of these processes could be extremely varied. For example, one of them could be addressing all the actions that need to be performed when an employee joins the company (e.g., the different kinds of access needed, placing orders and tracking the assets for the new employee, enabling his telecom and conferencing tools, etc.). Another example would be of how the service desk team handles priority tickets for a new acquisition, and yet another example could be of how an administrator manages a request received for building a new firewall to protect the critical resources of the company.

Generally speaking, these processes have been introduced at various points since the inception of the company and are shaped by multiple factors working along with the actual problem they are solving. There are always a number of constraints and influences that are taken into account before a process is built. These influences can vary from being financially driven to being determined by the people involved, geographic locations, current industry trends, and so forth. Many a time, it happens that a process that probably seemed very efficient at the time it was introduced seems highly obsolete or redundant when revisited some time later.

Unless revised often enough, over time some of these processes tend to become heavy. Factors like dependencies on a select individual(s) to execute the processes or collaboration between multiple teams executing steps that may no longer be relevant add to the weight of these processes.

Any organization that wants to stay relevant in today's fast-changing world needs to be highly efficient and proactive about circling back on its processes and methods and keeping them current. This is a very important trait of a company that is achieving continued and sustainable success.

These were some of the challenges that my company was grappling with back in 2013. Personally speaking, my employment contract with a large American enterprise was about to expire and this meant I was in a state of flux, professionally speaking. As a stopgap arrangement, I had accepted the role of an infrastructure automation engineer.

Thinking back now, at that point I was as clueless as an Eskimo in a desert, not really comprehending the magnitude of this change. Little did I know that this simple reassignment of duties would alter my professional destiny and snowball into the huge organizational transformation process that was to follow.

CHAPTER

1

How IT Operated: People, Processes, Technology

When I began my new position in 2013, IT was running in a traditional manner. That meant there were large teams focused on specific domains of IT operating independently. This led to them not fully grasping the overall impact of their work or aligning with the goals of the organization.

There was hardly any creativity at play and every task would end up being done in the most mechanical and staid way. Collaboration was more of an effort than the default practice. Teams would often work in silos and had very low visibility into the other teams' current undertakings and their roadmaps.

S. M. Farooqui, *Enterprise DevOps Framework,*
https://doi.org/10.1007/978-1-4842-3612-3_1

People

While speaking about various scenarios, I would also like to give you examples of the typical mindset of an employee back then. There were many employees who had been with the company for a very long time and had been doing the same kind of work for most of their careers. Many individuals owned a certain piece of some business function, which gave them a false sense of security about their jobs.

My colleague Ravi was the epitome of such a mindset. He had been with the company long enough to know critical details about quite a few processes. If ever you sought this information from him, his reluctance to share would be obvious. This led to frustrating and sometimes humorous situations on the floor.

For instance, before a scheduled automated process could run, he would manually complete the task. He would then smugly declare the automation ineffective! If anyone tried to challenge the effectiveness of the same, his favorite line in response would be, "This is how we've been working for the past so many years and if that were not good enough, how did we function well this long?"

In hindsight, his defensive behavior was mostly to reinforce his position as an indispensable piece of that project. We saw similar behavior being demonstrated by others across the employee spectrum. And this did not augur well for the healthy functioning of the teams.

Another instance that comes to mind is one of the conference calls I was a part of. There was an issue of very high priority with one of our services that needed immediate resolution and all the relevant employees were dialed into a call. When a request for certain information from a database was made by one of the meeting attendees, the database admin nonchalantly opened a search window and began searching for one of the simplest queries that was in turn projected on a screen for all to see. Needless to say one of the senior leaders was very vexed at the incompetence and the indifferent attitude to professionalism.

These instances illustrate how with time, different practices in an organization can become chaotic and dysfunctional. The worst thing that could happen at a workplace is for employees to stop caring about their organization. The moment people lose motivation, quality and productivity both take a hit. Innovation and creativity are lost somewhere within the negativity that spreads faster than mold on stale bread. It is very important for an organization to realize that any change they want to introduce needs to be enabled and blessed by their employees. Without this, transformation will be superficial and may not reach standards expected of a successful workplace.

It may be easy to state that those scenarios should not have reached the low they had or that fixing them should not have been an insurmountable task. But the fact was that over time with changing landscapes and multiple platforms being introduced, numerous hands operating in the environment, and hosts of tools deployed in the system amid highly dynamic market conditions, things got slipshod.

Processes and Technology

Work was getting done but it was not easy. For instance, I would like to share details about how the infrastructure change deployment process was executed.

Executing an infrastructure-related change required performing the following activities:

- Gathering the list of affected devices
- Determining the different variants in the list
- Understanding the business impact of the change
- Defining clearly the steps to implement the change
- Finding the right change window
- Determining the stakeholders and the impacted users
- Putting a test plan in place to validate the change and a rollback plan to address any failures

Each of these tasks seemed challenging with no proper support provided by any of the technologies that were implemented. With minimal automation implemented, most of these tasks were performed manually. As with most manual processes these tasks were time consuming, sometimes error prone, and not built to change the environment.

Incident management was another area where we had an improvement opportunity. Improving the configuration of the alerts to reduce false positives and capturing outage notifications consistently also required improvements. These gaps translated to an opportunity to streamline and improve the Network Operations Center (NOC) team without adding people.

To support the teams through these challenges, a number of tools were deployed. These tools had the capability to discover the current landscape, manage and support it, and detect any aberrations early enough before they caused any major impact.

However, we needed to improve the process and people flows to get the most out of these tools. For instance, high-end tools performing tasks like discovery of assets, monitoring system health, and collecting metrics from these systems

were all in place and required the right integration, and effective leverage of the data that was being produced needed improvement. The result was some outages on some business-critical services that could have been prevented.

Based on the demands of work, a lot of which was created by false-positive alerts, team members expended minimal effort on the right kind of work and couldn't find the time to shake things up and drive strong improvements. Innovation did not exist and team morale was low in this group.

This state of affairs required an overhaul. Also, the current structure was not sustainable, foreboding a potential crash if requirements were to scale up. And scaling was not the only concern—there were careers at stake. The industry was evolving, more so now, at a rapid pace.

The same admins who were much sought after in the industry four or five years ago were suddenly on the verge of becoming obsolete because they had not kept up with the changes and trends in the IT industry. It was not enough to be an expert on just one technology, tool, or platform. For example, in the present day, it is not uncommon to expect an MS Windows purist like an Exchange admin to be able to perform a basic level of troubleshooting for UNIX platforms.

There are many other similar examples where multiplatform knowledge is a must for individuals to survive in the current times. Without a compelling vision and strategy to drive continuous change, it's easy to be comfortable in the status quo.

A few leaders within the organization had assessed the situation and decided change was required. The leaders were savvy enough to realize that change had to start somewhere and were courageous enough to take those bold steps to make them happen. They were also ready to accept small failures if it meant there was hope of achieving some big success. This paved the way for the advent of greater things to come.

CHAPTER

2

Automation: Baby Steps Towards IT Transformation

As the organization realized the need for transformation, the focus shifted toward developing a strategy to get things started. Optimizing the way different teams operated felt like the first of many things that would need to be done. The way to achieve this was quickly decided to be driven through automation.

Often, having a dedicated team to implement automation within an enterprise is thought of as a luxury rather than a necessity. In my company, Aditya, our VP of internal IT infrastructure and technology office, recognized the need for change. He knew that the time had come for this luxury to be made a necessity and he wanted us to focus on transforming IT functions. The immediate strategy to get things going was by aggressively automating our way out of the current state.

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_2

I was hired as a part of this strategy and was expected to drive automation for the organization. My expertise in different programming languages and the work I had done in my previous role had landed me this opportunity. I would later realize that taking up this new role was one of the best career decisions I had made. This role helped me tap a latent passion for driving optimization by means of automation. It also helped me hone a skill to identify areas where change was needed.

Looking back, after my high school, I knew that I wanted to be associated with software programming for the rest of my life. Back then, I was invariably attracted to the unlimited potential that good software could offer and equally intrigued by how complex it was to write a good piece of code. My choice of academic courses hence veered toward building those skills.

During my Master's program at Loyola University, Chicago, there were two courses that I really enjoyed. The first was the software engineering course facilitated by Associate Professor William L. Honig. That course left a lasting impression on me because it was all about how a team should function, which I had never really thought much about in the past. The course attendees were divided into five member teams and every member had to choose a specific role to perform on the team. I picked up the role of a developer as I was very passionate about programming. Other roles in the project were those of a typical development team—a team lead, a quality assurance engineer, a developer, and so on.

The second course that I enjoyed was called Extreme Programming, which was focused on the usage of XML and its integration with the Python language. This was the first time I had come across Python, but it has since stayed with me. This class was taught by Professor George K. Thiruvathukal, who has been an inspiration for me purely because of the energy and passion he infused in his job every day. We would be assigned very interesting projects during the course such as creating a "calendar" application, for example. Building these solutions from scratch would mean we had to wear multiple hats at various stages of the project. The challenges we would face and the solutions we would ultimately create would be extremely engaging and satisfying.

These two courses among the others left an impression on me and probably sparked that interest to develop solutions that would be fully focused toward solving a problem or addressing a requirement in an innovative way.

Shifting back to the story of my new job, I was part of a team that would manage implementation of different tools, and I was responsible for owning software that helped with process automation. This allowed me to see the big picture. I had an opportunity to closely inspect and understand how different teams and processes operated and the challenges different teams faced. I was the lone member responsible for implementing process automation to

help create capacity for all the other IT teams. The leaders had done their homework on creating the best possible strategy to drive this transformation. The strategy was based on their vision of what they wanted to achieve and also on how some other organizations across the industry achieved success in driving similar changes. How this strategy would work in our organization was not very clear though, as each organization works differently and what works in one organization is not guaranteed success in a different organization. Even with a sound thought process backing this transformation process, it was still an experiment to some extent, but one that was not too expensive or disruptive to perform. Achieving even limited success in this would far outweigh any setbacks or failures it might run into. The die had been cast and the risk seemed pretty low.

As far as I was concerned, the road toward introducing automation was not clear-cut as there were no internal references to fall back on. I was counting on the support and faith from management to back me up and the domain knowledge of the many experts I was surrounded with.

The initial challenge for automation was to be able to identify the ideal use cases. My development background did not equip me with the right exposure to understand the challenges in the IT operations world yet. I would soon realize the amount of effort that is needed by the IT folks working behind the scenes to allow an enterprise to function.

While identifying use cases, it often happened that what we thought was a good automation opportunity either turned out to be too complicated or was not feasible due to technical challenges. A good number of proof of concepts (POCs) had been performed by implementing hypothetical situations, but most of them remained in the conceptual state and never made it to production workloads.

Pressure was mounting on me, and most of it was self-created. Given that I had worked with product teams in agile mode in the past, a couple of weeks with no deliverables to talk about seemed like a highly nonproductive period. Thankfully, things changed for the better.

The Big Breakthrough

The service desk team had come up with a requirement to help them with managing the process for an employee who was moving out. This gave us our first major automation use case, which we called the "HR Exit" process. The process involved integrating a number of end points both internal and external to the company. Some of these systems included active directory, telecommunication system, ERP, internal applications, servers, and many more. Connectors to these systems were implemented by using various techniques

such as performing REST and SOAP Application Programming Interface (API) calls, executing queries on databases, and remotely executing commands and scripts on systems by using default operating system utilities such as WMI and SSH. Writing scripts for each end point was not too complex a job, but all of these had to be tied and orchestrated together.

It became clear that all these scripts that were written for automation needed to be managed efficiently. Ignoring this would mean that the automation, implemented to optimize and simplify processes, could complicate matters.

This is where the tool CA Process Automation came into the picture. CA Process Automation is an automation platform that provides inbuilt connectors and features for orchestrating IT automation solutions. With the help of this tool and some scripts that I implemented in a few popular scripting languages such as Python, PowerShell, and VB, one of the first automation solutions was born.

The manager at the service desk team assigned a subject matter expert (SME), Joe, for this process, and we worked in collaboration to deliver the first cut of the automated HR Exit process. Joe would later join me to form the very first version of what later became one of the largest engineering teams within my organization.

Although Joe was working with the service desk team and had little exposure to coding or programming, he had a very analytical and creative mind. Joe's skills would be of tremendous use in the future, because automation is often about thinking out of the box, and creativity figures high in the top skills to have in a good automation engineer.

Joe definitely exemplified this definition to the hilt. He would often instantly come up with ingeniously simple ideas to solve a complex problem that to him seemed the very obvious approach to take. This quality of his helped us in engineering the solution for processing the exit requirements of a departing employee. I still remember the very first demo we performed for my immediate team after implementing the HR Exit automation process. CA Process Automation did a good job in capturing the state of a process that was underway. The workflows that were running would display in sequence, the current step being executed marking each completed step as a success or failure.

The looks on the faces of my peers during the demo and the compliments that followed after were very gratifying. After all, I was a new member in a relatively unknown world, working with some very seasoned professionals. Getting recognized by my peers meant a lot. It felt like Joe and I had struck gold on our very first strike!

It was a little too soon to be celebrating, though, as the solution was still just a POC and would have to go through a series of enhancements and hardening before it could be made production-ready. But we had already made an impact. We had shown that it was possible and a lot of opportunities existed for us to look at more closely.

Having received a positive response, Joe and I worked together to swiftly convert the solution from being just a concept to a production-ready solution. This solution was then used as a platform and a reference for a number of solutions that were implemented later on.

More Use Cases

Soon enough, working with some other colleagues across multiple teams, I was able to identify the next set of solutions that would need to be automated. Many of these solutions were focused toward automating processes around infrastructure management and operations. These solutions were less processes and more utilities for teams that consisted of network and system admins.

One such solution worth mentioning was the P2V process (converting a physical server to a virtual server). For this requirement, we used the development toolkit provided by VMWare. Although the most complex part of this process was the actual server conversion itself, the scope was much larger than that. We also were required to manage the preconversion and postconversion process specific to our organization, which consisted of integrations with other end points such as the configuration management systems and domain controllers.

When the request for this automation came to us, the odds were certainly not in our favor! The scope of the work was undefined; no one was in a position to confidently state exactly what needed to be done and what was possible to achieve. I was in untested waters not knowing which way to swim. To make things worse, the solution was expected to be delivered in a matter of a few days.

I had partnered with Isac, a VMWare expert, for this and together we burned the midnight oil for quite a few days to get this implemented. The first thing we did was to define the scope for ourselves followed by understanding the feasibility of what was achievable. Together, we were able to pull off what seemed like an impossible task, and we gave a demo of the solution to the stakeholders well within the stipulated time.

The solution was appreciated by all stakeholders and both Isac and I were recognized for the passion and the skill that we displayed. Unfortunately, the solution was never really used as the priority and the direction for the project had changed and it was no longer required. The upside to this was that during

the implementation of this process, we gained a lot of knowledge that came very handy later on. Often, we learn a lot more from failures we face rather than the success we see.

Isac and I went on to become very good friends from there on as both of us had developed a mutual respect for each other's talent and our passion for work. It is moments like these that leave an imprint in your mind and form partnerships at the workplace that last forever.

By this time, Joe had improved his coding skills and was churning out good-quality code and quickly identifying probable candidates for automation, particularly within the customer-facing IT service delivery side. Aditya had taken note of our work and made a swift decision that had a long-lasting and game-changing impact.

Aditya worked with my would-be boss, Anil, and they let me build a team and gave Joe the option to join as my first teammate. Joe didn't bat an eyelid and partnered with me instantly. After a few more successful implementations, two more very talented resources were added to our team, Sid and Vince. The new team members were not only good at automation, scripting, and adopting new technologies, but were also highly motivated individuals who were looking for opportunities to prove themselves.

Sid was very high in his creative quotient, and he was very clever at authoring smart scripts and delivering solutions to the point. Vince was more meticulous in his approach: he was good at planning and also very quick in learning new technologies. As a group, we were unstoppable as each one of us brought different skills to the table and were collectively putting our weight behind a common goal. All of us were just about ready to ride the transformation wave that was set in motion.

The journey had begun!!

CHAPTER

3

Challenges Faced Early On

The automation team shaped up well and was geared up to delivering the promise of automation. But this came with its share of challenges. It would have been unwise for us to assume that every other team would be as excited and as motivated as the automation team about this transformation journey. We were too early in the journey to be able to share any success stories with other teams that could have given them the confidence to embrace this change.

Accountability and Ownership

We faced resistance from some teams when we brought up discussions around automation opportunities in their teams' work areas. It was extremely difficult for us to make them realize that issues like service outages and process bottlenecks that they frequently experienced could be resolved through automation. Also, for certain employees, as a result of working on repeatable tasks, boredom related to their work had crept in. This resulted in bringing down the motivation levels for those employees. The zeal to own a process and make improvements around it seemed to be dwindling. In certain cases when issues surfaced, responsibility would be deflected to other members within their team or members of the other teams. This attitude was contagious and had the potential to become endemic.

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_3

At times it so happened that, when there was an outage or a service interruption, the network team would be the first to be put under the scanner with very little evidence justifying this behavior. It was rather convenient to assume that network-related issues were at the root of the problem. "*The network seems to be slow*" was a line that often used to come up during the discussions around determining the real cause of the problems. This would create a new tangent for troubleshooting the problem, which resulted in a lot of time being wasted as the focus of fixing the problem would now be around improving the network performance, whereas the real issue could have been completely different. The issue would ultimately be resolved by rolling back a change made at the application level or at an infrastructure level. But, there would be limited visibility into why the issue surfaced in the first place and what exactly was done to resolve it, as there were many hands working in multiple directions to resolve it.

The reluctance to take responsibility by some employees was evident and was plaguing the work atmosphere. I would attend many meetings and conference calls where this behavior was displayed. When individuals were called out for not handling responsibilities effectively, not taking the initiative to improve their areas, or not doing enough to find opportunities to automate, they would get extremely defensive. There was disgruntlement about other employees trying to trivialize their work and not understanding the challenges and complexities of the respective teams.

Championing Automation

There was definitely a need to evangelize for automation, and we also felt that this was needed globally in our organization. Even though our organization operated in multiple global locations, in its early days the automation team operated from a particular office location, and this was limiting their area of influence. To improve the coverage of automation and to get the larger team on board, we decided to have Joe travel to our headquarters located in New York. The sole purpose was for Joe to connect with as many teams and as many individuals as possible and to be the automation champion. We wanted to position him as someone who could work with the various teams in this location and help them improve their work efficiency by driving automation in their respective areas of influence. As a part of his preparation for this trip, Joe did well by connecting with the different teams before his travel and setting up a series of meetings and discussions with many key leaders and influencers across the organization working from that location.

Change, when introduced, almost always faces resistance. It's the same in most organizations and we faced our share of resistance as well. As noble as our thoughts were when we sent Joe on this odyssey, the sailing was not as smooth as we had wished. Even with all the preplanning and arrangements Joe had

made, he was unable to get much traction from the teams. Session attendance was minimal and those who attended were close to hostile at times. The most common question we faced around automation was "*Will we lose our job once our work gets automated?*" This was a tricky question to answer. While there are no two ways about automation eating into jobs, the reason we were driving automation in our organization was much more encompassing than reducing head counts and driving savings from it. With automation, we were trying to drive efficiency into the system. In some industries, automation is a proven model to scale operations. Many organizations have increased the head counts of their skilled labor after introducing automation as they were able to scale up their business. Scaling up meant more manpower would be needed. The requirement for the employees would be to upskill themselves as the machines would now be performing the routine mundane tasks and humans would need to focus on the next set of challenges to be solved.

Driving this point home to a set of nervous and reluctant employees was an arduous task. The other common comment we would get was "*we have fully optimized our work and there is nothing more to achieve with automation*." Having said this, the teams would try to shut down the conversation about automation even before it really began. We had to be shamelessly persistent at times with certain teams to make any inroads into their areas of work. It would often be like trying to find a chink in the armor and make our way through that perceived weak spot. Often, the chink in the armor would take the form of one or two employees who supported the idea of automation and were motivated to drive that positive change in their teams. The other motivating factor for these employees was the opportunity to do something interesting and learn something new. We observed the emergence of a pattern of how the adoption of automation by different teams would need to be managed, and each interaction with the different teams was a learning opportunity for us.

Bullheaded Approach to Drive Automation

The organization was trying to keep up with the changing landscape in terms of the technologies hitting the industry and our adoption of those technologies. With little scope for expanding the teams because of budget constraints, creating capacity for the existing employees to start transitioning toward adopting newer technologies was emerging as a big challenge, one which we hoped automation would be able to help with. Our aim was to automate as much of the current work as possible to create capacity within the teams to start exploring newer technologies and accomplishing bigger tasks. This was the message we tried hard to convey to the teams with little success. By the time Joe concluded his trip to the New York headquarters,

he had spent time with many teams and individuals spreading the word on automation. He returned home with mixed feelings, not knowing what to expect next. I remember the disappointment he expressed to me when he later recalled some of his interactions on the trip. For quite some time, he was unable to fathom why automation generated such resistance from colleagues who were other wise quite easy to work with.

Even with all the resistance he faced, he was still able to get a buy-in for better collaboration over automation from a handful of teams. Immediately, we started cashing in on the little bit of momentum that was generated by Joe's trip and shortlisted the next set of automations that would be focused upon by the automation team. What worked in our favor was the fact that there were a lot of inefficiencies then in the system and the teams were mostly focused on keeping the "*lights green.*" Most of these teams had no plan for how they were going to create the required capacity in their teams to tackle the more challenging requirements that were expected of them. This meant there was plentiful work readily available for the automation team to get started. As a result, we were provided with a golden opportunity to make a long-lasting impact in a short period of time by automating the simple processes or the notoriously famous "*low-hanging fruits*" first and showcase the big returns from these automations. This was a chance that we would not let pass us by, and we delivered some solutions quickly to further the momentum.

Ignorance Can Be Bliss

As the automation team was not fully aware of or influenced by the complexities of the other teams' day-to-day work, the issues these teams faced did not appear to be too complex to solve. This irked many people because they felt that it trivialized the value of their work. The automation team proposing new ways to simplify the work these teams had been doing for years didn't sit well with most of them. Ultimately, the teams needed to open up and embrace automation wholeheartedly, and we needed to make them feel as much a part of this process as the automation team. We felt the need to collaborate even more and to connect with the teams in a more casual setting. We tried out a few things such lunch-and-learn sessions and brown-bag brainstorming. *Conversations started flowing well over a slice of pizza*. With the help of these casual interactions, we finally managed to break the ice and set up follow-up discussions. We often went into these discussions with data from our ticketing system and made the teams realize how much time they were spending on performing very rudimentary and repetitious tasks. Opinions started changing and people became much more receptive and appreciative of the helping hand that the automation team was trying to provide.

With time both sides, that is, the automation team and the teams we were working with, had matured in their perception of automation. The underlying tension eased as many employees realized that the intention behind this whole transformation effort was noble. Each one of us was trying to do the right thing to help the organization. We grew better at understanding how automation workflows integrated with existing processes, identifying the next set of possible automation opportunities got relatively easier, and the support from the teams was much stronger.

The Engagement Process

With the help of these interactions with the teams, we carved out a process to carry out our optimization initiatives powered by automation. It was important that the teams were involved right from the first step. This was often the key to have their buy-in to the automation of a specific use case in their areas. The process could be defined as follows, and as shown in Figure 3-1,

- One or more members of the automation team would reach out to the other teams to initiate the process. These team members would be interviewed about their pain points and asked to identify the top two or three processes that consumed most of the time within their team.
- An "Automation Consultant" would be assigned from the automation team to drive this process.
- The teams were asked to appoint a SPOC (Single Point of Contact) who would provide details of the process and drive initiatives for their respective teams.
- Together, the Automation Consultant and the team SPOC would define the requirements and identify the integration points to help understand the complexity of the solution.
- The Automation Consultant would then take the lead in designing the proposal for automation and get it reviewed by the Team SPOC and at times by the architects within the teams.

- This would be followed by the build process or the implementation phase. During the implementation, a number of connectors and other utilities would be built. These would then be classified into two categories:
    - *Generic components*, which could be shared by other solutions, for example a solution performing file-related operations or a solution performing activities on a remote system.
    - *Solution-specific components*, which were specific to a particular requirement. For example, a connector for a telecom solution or to a particular network device.
- The different components that were built would be orchestrated within the CA Process Automation (PAM) tool.
- The solution would then be tested in the development environment and signed off by the respective team members before being pushed out for production.

Note It did take us a good number of iterations to come up with this model, and the model has never stopped evolving to this day.

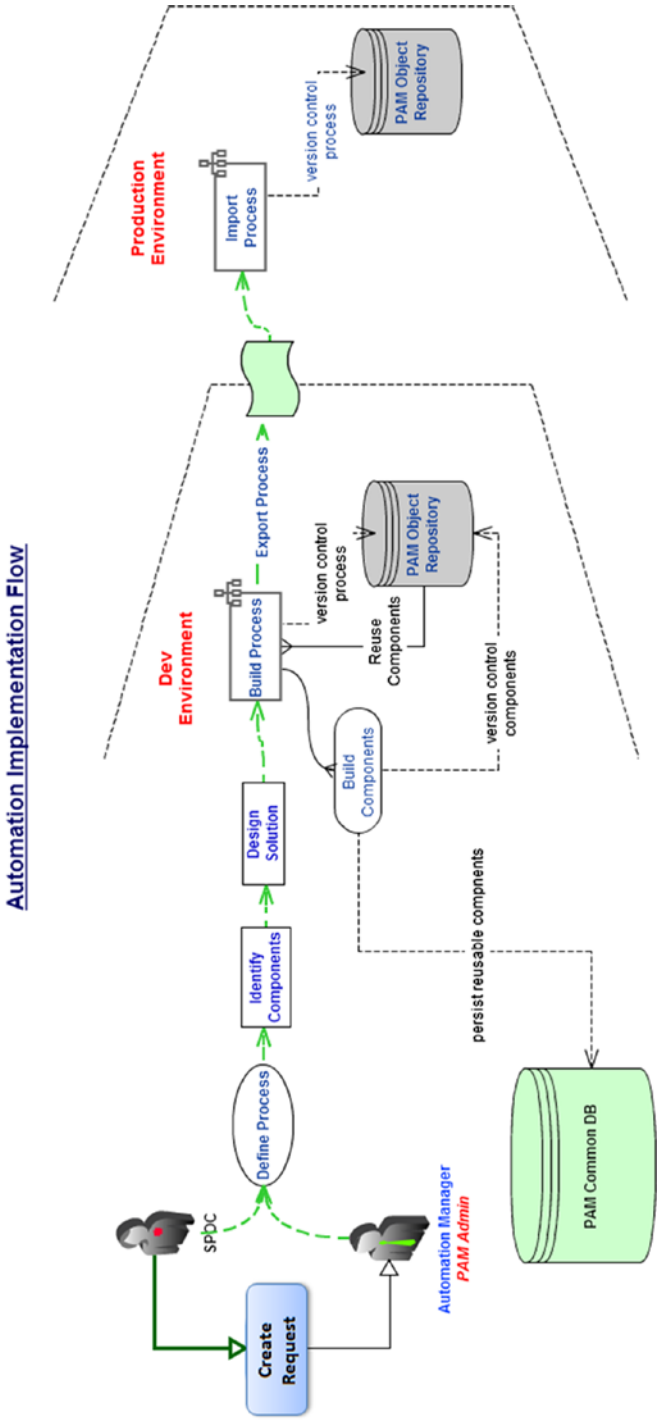


Figure 3-1. Automation framework

This exercise had its share of challenges. Initially, the teams were uncomfortable revealing the challenges they faced. As I mentioned earlier, there was an early phobia of automation among the teams as they thought it meant a reduced head count. A few others in the teams thought the tasks they performed were far too complicated to be automated. They would undermine the power of a good piece of code and the orchestration technologies out there. But to be fair to them, they had not seen real automation in action within their field to display full faith in automation just yet.

I remember a team lead telling me: "*My application is different from the others; there is nothing that can be automated here.*" Ironically, a few months later, the very first instance of a "code deployment automation" that we implemented was for that particular application! It later became a reference solution for many similar implementations. That particular colleague is now a very good friend of mine and she is a true believer in automation and also has become an outspoken evangelist for automation and its benefits. I sometimes wonder if she distinctly remembers those very words she spoke! I do not bring it up, lest she gets embarrassed about it. Together, we have now deployed umpteen number of automations and I admire her for the way she has transformed herself and adopted to the new way of doing things.

Without full support from the teams, we would get only pieces of information. Incomplete data often kept us guessing on the full scope of the automation opportunity. This resulted in multiple follow-ups on our end to gain useful data points from them. A technique we had mastered with a good success rate was to identify that one resource/team member who was actually excited about being a part of the whole automation process. He would become our go-to person and would provide us with starting points and guide us toward asking the right set of questions to the right people. Once the automation was delivered, we would make sure that we highlighted the contributions of this individual and get him his due recognition. This approach would encourage others to be more participative in future automation opportunities. The momentum kept on building, and most of the teams and individuals started aligning with the automation strategy of the organization.

We did fall back on the leadership team on occasion to propagate the importance of automation and push their teams to start collaborating with us. Short programs had been launched that encouraged and rewarded employees who were pioneers of automation within their teams.

Bullseye

Some areas where we succeeded in delivering quick value were the service desk support team, the NOC, the systems admin team, and the application support teams. The use cases that were automated ranged over a large set of areas. The following are a few of those use cases.

Use Case 1: Incident Management Automation

Problem Statement: The monitoring systems currently in place were detecting a large volume of issues in the environments. The operations team in place was not able to keep up the pace in resolving these incidents reported by monitoring systems. This was resulting in a delay in resolving the incidents and was adversely affecting the business services.

Solution: A number of autofix solutions implemented by the automation team were able to fix a large number of incidents reported by the monitoring systems without any human intervention. Most of these solutions ran in real time and hence were fixing the issues as soon as they surfaced. This resulted in a highly efficient operations model, and the MTTR (Mean Time to Recovery) was greatly reduced, resulting in happy customers.

Use Case 2: New Server Build Process

Problem Statement: When a request for building a new server was placed by our organization's customers to host their application, the turnaround time from our provisioning team was proving to be a major point of dissatisfaction for the customers. There were multiple steps involved between when a request was received by the provisioning team and when it could be handed over to the customer, and almost all of them were manual.

Solution: A number of optimization opportunities were identified in the new server build process by the automation team. The process was first broken down into four different phases:

1. Acquire
2. Deploy
3. Manage
4. Retire

Each of these phases was then brought under the automation radar. The end solution that was delivered was highly optimized and proved to be one of the most successful solutions built by the automation team. More details on this solution are provided in the later chapters on DevOps and Cloud.

Use Case 3: Managing a Load Balancer

Problem Statement: During a planned (or unplanned) change or a maintenance activity, usually executed during weekends, application teams often had to reach out to the Network team. The Network team's on-call resources would help to disable and enable the respective application servers from the load balancers to perform the activity.

The Network team would be paged a couple of times during this activity, which resulted in a cost and time loss. This was an expensive approach to a simple requirement. Besides, there were challenges involved in getting a quick response from teams and all this had to be coordinated by the NOC team.

Solution: The automation team built a solution that would empower the NOC team as well as the Application support teams to perform basic operations like adding or removing a server from a load balancer without any support or help from the Network team. The solution was delivered in the form of a simple web form to make it user friendly. As this eliminated the need to page the network team and have them perform the activity, it proved to be popular with the Application support teams as well as the Network team. The solution reduced the time and cost involved by a whopping 15X times!

Use Case 4: Code Deployment

Problem Statement: Developers would reach out to deployment teams to deploy the latest code on the app servers. This process involved a formal request that was created and addressed by the support teams. Precious time was lost between requests getting processed. The support teams had to drop what they were working on and pick this up as any requirements from the developers would be a high priority item.

Solution: The solution that was delivered would intercept a new deployment request and perform an automated deployment of the latest code, as well as notify the respective parties of the progress and the end status. Again, all of this was achieved with zero human intervention and had a high ROI (return on investment) as the time lost between submitting a request and someone picking up the ticket to resolve was completely eliminated.

What we delivered then was not the most efficient way of implementing an automated delivery solution, which we learned about later in our DevOps journey. Nevertheless these kind of solutions made the adoption of DevOps much easier as the solutions and teams were primed for better efficiency by these automations. This was our first take on continuous deployment and with time these practices evolved into a highly advanced solution.

Automation Train Well on Its Way

Building these kind of solutions meant that work that used to take a few hours or days to complete, would now be completed in a few minutes or maybe even in seconds. This efficiency was achieved by empowering the support teams. A simple user interface would be tied to the automations that needed on-demand execution. This utility was then handed over to the actual team

that was requesting the activity. The wait time was thus eliminated and the processes became lean and fast.

These solutions became very popular as they provided wheels under the feet of teams who were now able to meet their deadlines faster. There were occasions when due to change in processes or technology the automation would deprecate, but in between cycles of being built and being deprecated, the solution often delivered huge value, which justified the time invested in building those solutions.

After adding two new members to the automation team, the speed with which solutions were being implemented improved significantly. Sid with his expertise in Perl and Python coding along with good exposure to UNIX, was able to contribute on solutions being developed for multiple platforms. Vince was an expert in RedHat platform and had excellent shell scripting knowledge and Cloud expertise. These were very apt skills to have when our immediate focus was on infrastructure-related automation.

Our team consisted of four members then, each with different skills that spread across a wide range of technologies. What most of us had in common was expertise on the UNIX platform and good coding/scripting skills. These traits had become the differentiating factor for the team in an organization predominantly had admins and support resources who, although highly proficient in their areas of expertise, had little to no coding or scripting skills.

Evolving into a More Advanced Automation Team

The automation team was delivering solutions in many different areas, both technology and process oriented. An important gain from delivering these solutions was the positive effect on the mindset of employees. When a new process was introduced, participation and acceptance from the teams were now significantly higher. All involved in the process were committed to making sure that the automation initiative was as comprehensive and efficient as possible.

The tables had turned, and the automation team no longer had to reach out to other teams and ask for work. As a matter of fact, too many automation requests began coming in, which meant that the team had to be scaled up in a short period of time.

Initially, we kept track of the work coming our way by use of traditional ways like MS Excel or plain text files. For each new request we would enter information like the summary, point of contact in the team, automation consultant representing the automation team, ROI (Return On Investment), ETA (Estimated Time of Arrival), comments, and so on. Although quite basic, this information helped us track the effort in a somewhat organized manner. We started publishing dashboards with this information to various leaders and stakeholders.

Automation, a Simple Four-Step Process

As time passed, we gained more experience with automating various kinds of processes and moved toward using a more organized approach toward automation. We began evolving into using more processes around managing the entire automation life cycle rather than focusing on just implementing a solution. This was derived from the automation engagement model that was discussed in detail earlier in this chapter, but was simplified into four key steps.

The steps were as follows and as shown in Figure 3-2:

- Identify
- Evaluate
- Build
- Govern

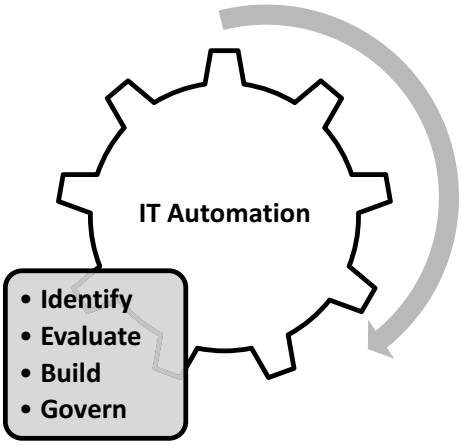


Figure 3-2. Steps followed during automation of processes

Identifying the right opportunities was the first step in the process. This often proved to be the most difficult step as it involved a lot of time and effort. Doing groundwork to identify which teams needed help and what processes needed optimization was often the first stage.

This was sometimes accomplished by analyzing data emerging out of service desk tickets that had data around change management, incident management, and request management. Also, interviewing different team members helped in this phase.

Evaluating the value of automating a shortlisted process was the next step and often the most important one. Once an opportunity was identified, a few check boxes needed to be ticked before it moved to the implementation phase. This was based on the business value that would be realized by automating the solution.

At a high level, the business value of the solution would be determined by understanding the capacity created for the teams and the quantum of risk mitigated.

The number of hours saved for each process, the complexity of implementing the automation, the count of manual errors that were eliminated, and the number of changes made though the automation became factors that contributed toward coming up with an ROI of automating the process. This would determine if going ahead with the automation was viable for business.

The **Building** process pretty much remained the same as explained a little earlier in the engagement process. Early on in the process, we anticipated the need for building certain solutions that would be required across multiple use cases and made accommodations for reusing those in the build process. This proved to be really helpful as we started implementing more and more solutions.

Governing the solutions became highly critical. As the automations started taking over, the dependency on these automations increased heavily. Teams now started focusing on tasks that were not yet automated and assuming that automation was doing its job. Every once in a while we would have an instance when an automation job failed or did not work as expected. If proper alerts and monitoring were not implemented for these solutions, the failures would not be detected and there would be a huge impact. We needed to maintain proper documentation around each automation solution, which helped us identify the owner for that solution, its impact, and the troubleshooting steps in the event of the solution failing. Also, a manual failover option had to be identified in case the automation was experiencing any challenges.

Managing the Automations was the next step. Once a process for building automations was determined, creating new automations became relatively easier. For the automation team and for the platform teams for whom the automations were being developed, it also became critical that they understood what role they had to play in the process of solution development and supporting the solutions. A few of us from the automation team put our heads together to bring some clarity on this, which is depicted in Figure 3-3.

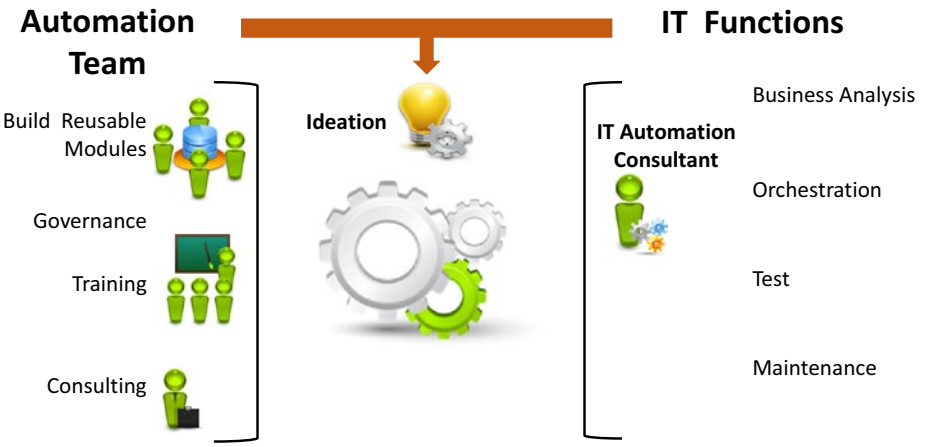


Figure 3-3. Roles and responsibilities

Roles and Responsibilities

Most of the solutions that the automation team developed involved two main stakeholders. One was the automation team itself, as they would be building the solution, and the second was the team that owned the process that was being automated.

Automation Team Responsibilities

The automation team was tasked mostly with creating the new solutions, identifying any reusable modules that already existed that could be used in the new solution, and identifying and marking any new modules that were to be built for the new solution as reusable modules. To facilitate this, the automation team was tasked with assigning one automation expert for every process to be automated.

The automation team was also responsible for creating awareness and spreading automation by acting as trainers and consultants for automation. Multiple training sessions were held by the automation team around the process for automation and the tools and technologies that were involved in building the automation.

Process Owner Team

The teams that owned the process that was being automated were tasked with assigning a SPOC from their team to lead the automation initiative. The SPOC would be the interface between the two teams and would be responsible for defining the requirements for automation and for bringing in the subject matter expertise requirements for the process being automated.

The SPOC would also be responsible for ensuring that the testing of the process was complete and provide a sign-off once the solution was developed. Finally, once the solution was in production, the SPOC would need to ensure that support for the processes was provided in an effective manner. This piece was often challenging, and the automation team used to play a major role in this phase of an automation life cycle as they were the most well equipped for debugging and identifying any ongoing issues with automations in general.

It took time for many of our users to understand that even though the process was automated, the ownership of the process still lay with teams for whom the solution was developed and not with the automation team. Although defining the roles and responsibilities did not fully solve the problem of ownership, it did help in putting some basic structure in place that could be built upon as time passed. We definitely improved on this aspect with time, and this was mainly due to the change in the mindset of the people.

Making automation more accessible for the various teams who were now increasingly adopting automation and the housekeeping activities around these automation solutions were becoming increasingly challenging. Collating the solutions so that they are easy for the end users was a major challenge. Managing the access to these automation solutions also became very important. By automating the processes we had simplified the processes to a great extent, it had now become possible for team members who were not experts in a particular technology and who did not fully understand the impact of the change to be able to perform complex tasks. Domain- or function-based segregation became important for all the automations that were rolled out so that one team did not have access to solutions built for and owned by another team.

Summary

In this chapter, I have called out a number of challenges that we faced while trying to increase our impact with automation. It is quite evident from our experience that most of the challenges that we faced were nontechnical. In fact, when automating a process, the one aspect we would be least worried about would be the technical feasibility. This was because technical feasibility was very easy to determine, and unless the technology involved was primitive there would be one way or another we would figure out to automate the process. What was challenging was to get a process in place that would drive the maximum value of these automations that we were building.

Mastering the art of effective automation comes with experience, and there will be a number of challenges that any organization will face when they set out on this journey.

In the next chapter, I touch upon how we evolved into a more advanced stage of automation and how building solutions like ITBot became table stakes for the automation team.

CHAPTER

4

Era of the Bots

As the automation team's expertise on building automation improved, we started spreading our wings even further by building more creative and complex automation workflows that were better aligned with the user demands and requirements. Many opportunities were identified while interacting with other teams for building solutions aimed at improving operational efficiency. The automations were grouped into larger solutions based on the different problems they were addressing and were termed as the automation bots by the team. Most of these automations were running as workflows in CA Process Automation, which did a great job in helping orchestrate the automations.

ITBot

Though CA Process Automation was great at orchestrating the solutions and had a shortened turnaround time compared to scripting solutions from scratch, we had identified one major gap in this tool that was hampering the adoption of automation. We needed to simplify the means by which these solutions were accessible to the users. At that point in time, the users would have to navigate through multiple pages and make quite a few clicks before they could reach the exact solution they were seeking. This would deter some users from using the automation and instead, they would go back to their old manual means of executing the desired task.

In order to overcome these challenges, we built a web interface that we called "ITBot."

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_4

ITBot was developed as a simple HTML wrapper for the different automation solutions that were implemented in CA Process Automation. We simplified the UI shown in Figure 4-1 to address the challenges mentioned in the preceding and positioned it as a one-stop automation portal where different teams came to execute their automation. We further enhanced it to incorporate the ability to send notifications and to be able to report the status of automations.

Figure 4-1. ITBot: the automation portal

Just building automation solutions is not sufficient; the value of the solution lies in the consumption of the solution and this is where the delivery and the accessibility of the solution plays a key role.

ITBot happened to be the first of the many other bots and automation applications that the team later implemented. The delivery of this solution had lit a spark in the team and they started looking at the delivery piece of automation solutions in a totally different light.

ITBot was not an instant success in improving automation user experience (UX) as I had hoped it would be. Many employees felt this was a redundant solution because it just provided an alternate means of accessing solutions that could otherwise be accessed from the CA Process Automation directly. The pain that I thought ITBot was providing relief for was not felt as acutely by others. Users who were accustomed to the existing solutions continued

accessing the solutions through the CA Process Automation portal. But this also highlighted a bigger problem. There were not many people depending on automation yet or really using the automation solutions that were being put in place to help them out. Part of this can be attributed to human nature. As humans we are generally hesitant to change the way we work even if the current process is time consuming or complex. Familiarity with the steps involved in executing a task and the reluctance to break away from that set pattern comprise perhaps one of the major hurdles to introducing a change. Any new process is almost always assumed to bring with it complexity, and people tend to avoid adopting it until there is no other choice left.

ITBot had a modern look and feel, and we had done a decent job of segregating the solutions for individual teams by providing each of the teams a "tile" to access their team-specific solutions. This helped them to easily navigate through the multiple solutions on the portal and save time. ITBot became popular for accessing the newer solutions that we were rolling out but was not immediately accepted for the ones that were already in place.

We did not want to force people to use the automated solutions by shutting other means of performing the task. Our focus shifted toward improving adoption by making the new processes as easy to use as possible by addressing their UX aspects. This was done by engaging the users more, observing the way they executed a task, and identifying the latent needs that could be addressed by the solutions to make them more appealing to its user base. By doing this, we were able to drive more people toward using the automations and we observed a steady growth in the adoption of automation. More adoption meant that the dependency on automation was also growing and we had to build solutions that were highly resilient and accurate. The ease of use of the ITBot portal and the comprehensiveness of the solutions it provided attracted more users toward it. This helped with the overall adoption of automation in the organization.

Soon enough, almost all the users of the automation solutions started using ITBot, and it became the de facto portal for automation. We kept on adding new solutions to the portal as and when we built them. We also realized that building ITBot for just our organization's consumption was not enough as there was a broader community outside our organization that was using CA Process Automation, and our knowledge could be beneficial to them as well. There was already a program in place that was the brainchild of a few key leaders in the company that created a channel to encourage collaboration between the IT department and the product development teams. A team was put in place in the IT segment of the company working dedicatedly toward driving more product sales for the organization. One of the ways this team was driving the program was by focusing on improving the internal adoption of the products our organization was building to sell externally. The idea was to test

out the products internally first and find any bugs or identify any enhancement opportunities for the products so the required quality improvements could be incorporated before the products were sold externally.

The automation team felt it was important to pass our observations and challenges to the product development team of CA Process Automation. We started engaging them and performed a demo of ITBot for them. The product team immediately acknowledged the significance of a functionality like ITBot within the product. They put this requirement in their backlog and prioritized it high. The very next release of CA Process Automation came with a new add-on utility, "Solution Suite," which mimicked the functionality of ITBot. The automation team was thrilled that not only were we driving value in the organization by optimizing process with automation but we were also able to influence the revenue-generating stream of our company by providing inputs to the product teams. This success story further strengthened the credibility of the team and proved that we were heading the right direction.

FixIT

Hot on the heels of the success of ITBot, we rolled out a self-help end-user solution named FixIT. FixIT targeted resolving end-user problems in a more efficient manner by providing a self-help website. The service desk team was getting bombarded with a number of calls reporting issues with end-user systems. The problems were wide ranging, from system slowness to access issues.

Most of the time, fixing these issues was quite straightforward but the process put in place to manage this was not the most efficient. The process worked as follows:

- On experiencing any issues with his/her machine, the user would reach out to service desk by phone or initiate the process by creating a ticket.
- This would be followed by a service desk analyst addressing the concern by having a phone conversation with the user to start the troubleshooting process.
- The analyst would often request remote access to the user's machine. This was needed to diagnose the problem and to download and install a package on the user's machine.

This process was highly time consuming and was a ripe candidate for automation. The service desk team managers had to constantly stay on top of their teams' available capacity based on a number of such requests coming their way, and this was turning out to be a costly affair.

In order to solve this problem, the automation and the service desk teams did a couple of brainstorming sessions and came up with a design for a self-help solution. Together, we performed some analytics on the requests received by the service desk team to understand where the volume was with respect to the issues being reported. The top ten solutions were selected based on this analysis and then addressed in the very first release of FixIT. The before-after state of introducing the FixIT solution in the organization is captured in Figure 4-2.

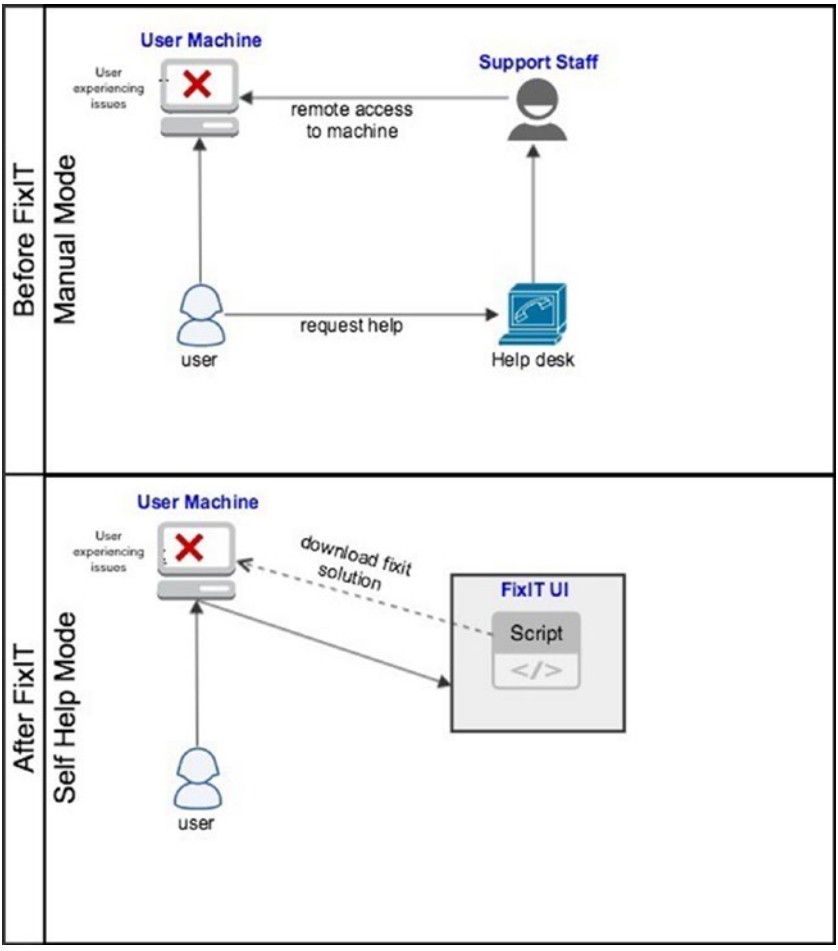


Figure 4-2. FixIT before-after state

The users were now able to resolve most of the common issues themselves by simply visiting the website (refer to Figure 4-3) and running the executables listed on the site. The service desk team was now redirecting the users to FixIT when they received a call and the issues were getting resolved much faster. With time, more and more solutions were added to FixIT and the user adoption of FixIT kept on improving.

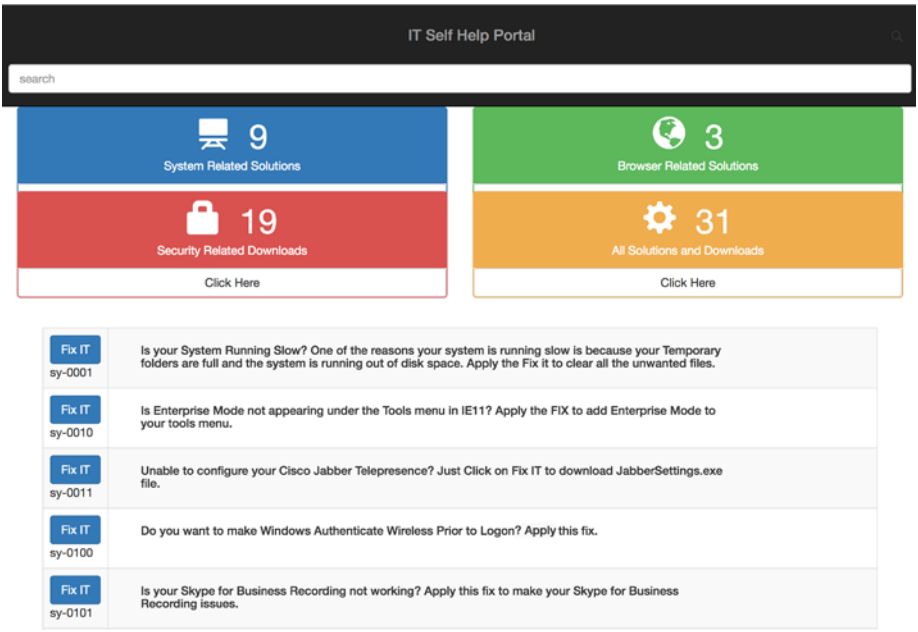


Figure 4-3. FixIT portal

Also, as a practice whenever a new service or a product was rolled out for the employees, the service desk team and the service/product release team came up with a simple three-step process to improve the success of the new service being introduced. The process would address any challenges with user experience and also address the challenges around adoption of the solution. The process was as follows:

- Identify what changes are expected for the end user. For instance, do they need to install new software or require a license to be deployed on their systems?
- Create an automation package for the change that needed to be deployed.
- Proactively push the package on the end-user's machine using remote management tools; in cases where this was not possible, the solution would be added to FixIT and the end users would be informed about the details of this solution.

Based on what I have seen, when developing a self-help portal, it is very important that you closely observe the journey of the users experiencing the problem, measures he or she usually takes to resolve it, and the complexity of delivering a solution.

A self-help solution should be driven by two important factors:

- Improving the user experience
- Improving the efficiency of the service teams

If either of these boxes is not checked then the value of providing the self-help solution is greatly diminished and needs a retrospection immediately.

InfraBot

In 2014, Aditya and I traveled to Las Vegas to attend the annual conference of one of the biggest Cloud vendors. For me, the event stood out because of the participants. My experience until then made me think that a conference like this would attract a lot of people from the management and leadership teams who would be making business decisions, and hence it was important for them to attend such events. But this event was different. We saw an army of young participants in terms of their experience in the IT field who showcased a completely different approach toward thinking about solving traditional problems. It was evident that their motivations were beyond just keeping lights green for their organization.

Each of the participants supremely confident, focused, and ready to take the challenges of modern IT head on. New technologies meant more opportunities for them and the Cloud platform had a whole ecosystem of solutions and challenges that needed to be resolved and designed for, which appealed to their basic instincts. Most of them were either developers trying to leverage Cloud to their advantage by building modern-day applications or from back-end operations teams looking for opportunities to simplify the challenges of running datacenters.

Aditya and I attended some of the breakout sessions about new Cloud offerings and got a chance to interact with quite a few participants. These interactions left an impression on both of us. We followed a divide-and-conquer approach where Aditya attended the business tracks and I attended the technical tracks. We did this for two days and would have discussions on how some of these sessions were relevant for us and how we should be adopting the new technologies out there.

Aditya was one of those leaders you come across rarely in your career. If I had to describe him in two words, I would use "positively disruptive." He had worked his way up from the ground of the IT chain, starting his career as a database admin. He had a very good understanding of how business should be built and supported from the bottom up. He had an acute understanding of picking up what new-era technologies were going to be relevant in the future and needed to be invested in. Seeing a young creative mind would always excite Aditya.

Anil (my manager at that time) and I were mostly on the receiving end of Aditya's creativity and had some memorable brainstorming sessions together. He would just pull us into his office and start shooting all these crazy challenging ideas, most of which would initially really scare me as I knew it meant I would be the one implementing them. But soon, I realized that these were the discussions that became the inspiration and the seed for most of the good things our teams would later be implementing and be remembered for.

Though the Cloud conference had lasted for only two and a half days, it had left a lasting impression on us. The event energized us with a craving to pivot and improve the efficiency of our own organization by leveraging these attractive futuristic offerings. Our focus moved toward identifying the biggest challenge we were facing at that time.

Swiftly, Aditya narrowed it down to managing changes on infrastructure efficiently, as this was something that had been bothering us for a long time. There were multiple occasions when a business service was interrupted and the root cause was zeroed down to a change that was pushed on a server by an admin. Questions like "*Was the change tested in a development environment effectively to test its effectiveness?*" and "*Was the impact of the change properly evaluated?*" were very uncomfortable for the teams, as there would be no clear answers. The biggest challenge we observed was the different configurations of the servers, which were a major hindrance in deploying any changes to test environments and relating the results with the production environments.

Configuration Drift

Although every new server was built from a standard hardened base image created by our engineering teams, over a period many hands would operate on the server to make these small, incremental changes. Also, these changes would be tracked or correlated effectively. Within a few months, the configurations would start deviating heavily from the expected standard state. This led to complications with understanding how a server or an application would behave when a change was required to be made to the server. This is how we understood and experienced **Configuration Drift**.

Configuration drift is a term that describes a process where changes are implemented to software and hardware components in datacenter environments that are not tracked or recorded properly. Over a period of time, this introduces a lot of complexity in managing datacenters when making any more changes to the components as the impact of making the changes cannot be gauged accurately.

Figure 4-4 illustrates how configuration drift results in configuration differences between servers that were created with the same baseline.

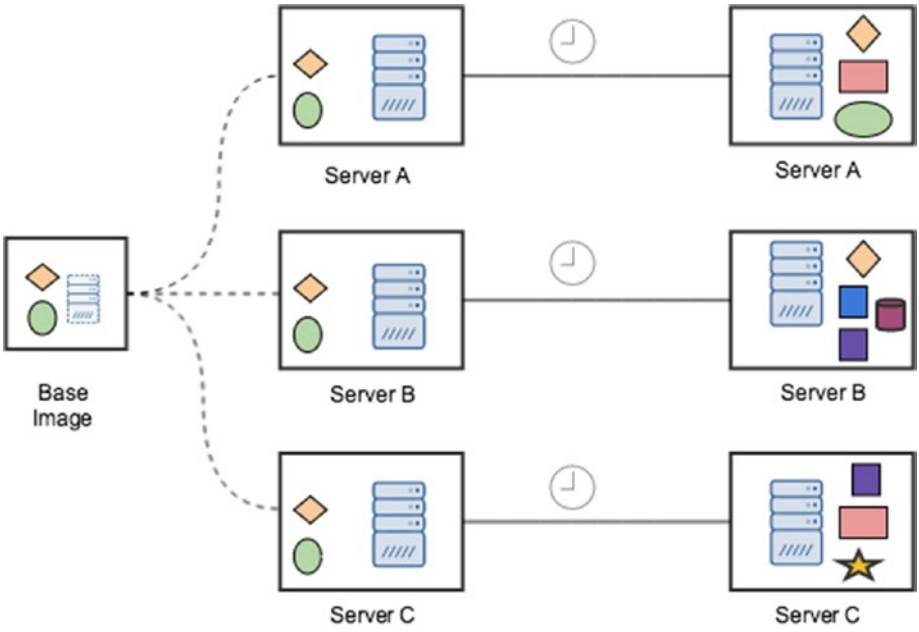


Figure 4-4. Configuration drift

Recollecting all these challenges the teams were facing, Aditya challenged the team with a new requirement. He suggested that we implement a modern-day solution that would have two key high-level features: Discover and Validate.

- **Discover**: Provide real-time information on the current state of the datacenter. This meant providing answers for the following questions:
  - How many servers are running on the physical datacenters managed by our organization versus on Cloud?
  - How many of these servers are running on Windows OS, UNIX, or any other platform?

    - What is the environment distribution for these servers? Determine how many servers are running production services versus any QA or dev services.
    - Determine and map the relationships between the servers and the applications. This would need to be highly dynamic and much more real-time than a static CMDB (Configuration Management Database) in order to keep up with the changes being introduced on a frequent basis in the environment.
- **Validate**: Continuously measure all server configurations in our datacenter and detect whenever a deviation from the desired state was introduced.

Aditya and I spent the evening after the Cloud conference talking more about the solution. We discussed the challenges this solution would be addressing, what the solution would look like, and also how pleasant life would be once it was implemented. Later that night, after I had put some thought into the approach we would need to take for delivering this solution, I made a call to my team back in India, who had just started their day at work. I updated them on the discussion Aditya and I had and the new challenge he had thrown at us. We had a long discussion on the solution design and talked about how we felt the solution could work.

Vince and Sid agreed to drive this effort and engage the rest of the team. Even though the team was not clear on why this had suddenly become a priority, they were excited enough to commit on delivering a POC within three days. I remember not sleeping well that night, but it was not stress that kept me up, it was the surge of thoughts that were coming to my head on how we can keep enhancing this solution and how it could be the one solution that solves most if not all of our problems at work.

That is one evening that I remember very clearly, especially the discussion I had with Aditya and the phone call that I made later. It turned out to be one of the defining moments for the automation team. The idea turned out to be the seed that grew into something really big and impactful. With more discussion we were pretty clear on the requirements for this solution. The team started working on a detailed plan for its implementation. The first thing we did was to go to the drawing board to finalize the design for the solution. Refer to Figure 4-5 for the solution design.

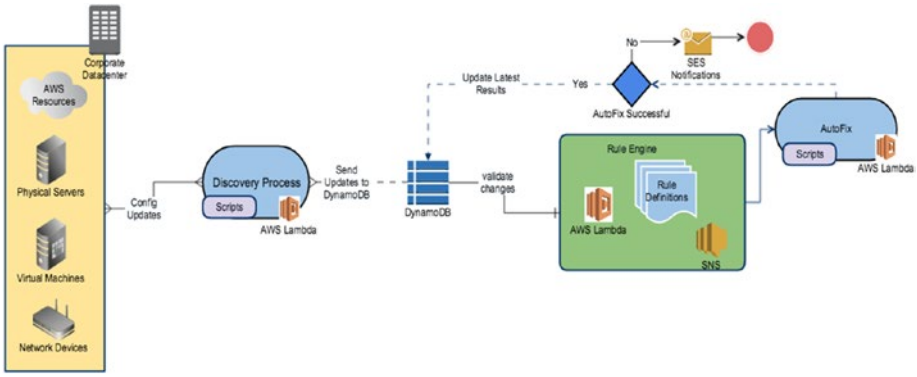


Figure 4-5. InfraBot architecture

At a high level, the solution was designed to work as follows:

> **Baselining**. Identify the baseline values for all properties we wanted to measure a system's health on.
>
> **Discovery**. A data discovery mechanism was put in place to obtain the current state of these properties from each server.
>
> **Rule Engine**. A rule engine was designed to compare the current state of the properties with the baseline and highlight the inconsistencies.

Since the solution helped us automate the management of our infrastructure, we called it "*InfraBot*." The most critical piece of this solution was the rule engine, which would perform the comparison between current state and desired state. Multiple rules were created to check the health of a server from different perspectives like its performance, security outlook, alignment to standards, and so forth. A "rule" would basically define what parameters should be compared between the discovered set of data and the baseline data. For each property we wanted to compare, we added a new rule. A rule would determine what properties from the server would be required to be discovered. The discovery process was initially performed at a 24-hour interval but was later optimized to an impressive 30-minute interval.

Case Study: Antivirus Coverage for Datacenter

The following is an example of how the InfraBot added value in the organization.

Problem Statement

Protecting all servers in a datacenter with antivirus is a must for an organization. There are different ways in which an antivirus coverage for a server might be ineffective. While the vendors provide a means of checking the health for its agent, it is not always fully integrated with the processes of the organization and can pose a challenge to manage the antivirus coverage for the thousands of servers present in the datacenters.

Solution on InfraBot

In order to solve the preceding challenge, we created the following within InfraBot.

- A new rule was created on InfraBot that checked the expected state of the different system services and the processes running on the server related to the antivirus software.
- A provision to include multiple parameters apart from the service checks like the configuration of the software and the connectivity (or heartbeat) of the antivirus agent with the central server.

These checks would give us enough information to depict a clear and complete picture of the health of a particular application, which contributed to the overall well-being of the server.

Multiple times a day, these checks would be performed on every server where the antivirus was expected to work. Any of the checks failing would trigger a notification to the team managing the antivirus solution.

Business Value

Proactive detection of antivirus health is very critical to prevent any security-related incidents and is an obvious advantage of having a solution like the one described in the preceding.

The solution also helped in highlighting the gaps before they turned into an incident captured by a monitoring solution and needed to be addressed as a critical failure in the system. This approach would give the teams addressing these gaps a breathing space to fix the solutions at a comfortable time rather than on a tight deadline measured on SLAs (Service Level Agreements).

Expanding InfraBot

As the automation team did not have domain expertise across all the required areas in IT to comprehensively determine the checks needed to be performed on a server, we started interviewing other teams to understand if they wanted to add rules for their particular areas of expertise. The solution was received well, and we got a pretty good set of new rule suggestions from the teams we had reached out to. A rule was added in InfraBot for each new requirement that came our way. While implementing a rule, we would also associate the rule with an owner. An owner would be the team responsible for managing that particular configuration of the server. The owner of each rule would now be responsible for ensuring that any failures in the rules they owned were addressed and dealt with in a timely fashion.

Based on the inputs we received from multiple teams, within a short period of time we ended up creating more than sixty different rules. The rules would perform different types of checks on a server like the capacity usage, operating system patches, service state, user account configurations, integration with configuration management system, firewall settings, and many more. This was a proactive way of dealing with anomalies in the system rather than reacting to those anamolies after the damage was done.

During the implementation of InfraBot, we realized that we were not fully equipped with the right tools in our environment to deliver a complete solution and also to be able to scale it. To overcome this shortcoming, we adopted some open source configuration management. The first version of the InfraBot was rolled out within four months from the time the idea was conceptualized. Since then, the solution has been highly valuable by proactively highlighting the gaps in the datacenter and helping to resolve those gaps before any service interruption or outage occurs. Measuring the full impact of this solution was never going to be easy, as not all findings would have resulted in an outage. Nevertheless, *with each failure caught and resolved, there was one less potential vulnerability in the environment to worry about and that itself was a huge success for the solution.*

With time, this solution became a "*household utility*," so to say, for most of the operations and support teams. Almost all the data that the teams wished to have access to for a server or an application including the existing gaps on them was available on InfraBot. Most of the teams would visit this site at least once a day to get this information and to define their work backlog for the day. Improving the quality of this solution was a continuous process, and the solution kept on maturing over a period of two years with inputs from the various teams that were using it.

ScoBot

As we tightened our grip on the efficiency of operations in the organization, one area that was constantly emerging as a threat to derail all the good work by different teams was the constant threat of failing the compliance requirements for the organization, as these requirements are pretty intense. As any large organization, we were also bound to abide by the guidelines provided by some widely adopted governance standards. While adhering to these standards appears to be a mean task to accomplish, most of them have been put in for a good reason. If an organization plays according to the rules of these standards, then automatically this ensures that most of the risk and vulnerabilities in the system are eliminated. The challenge lies in how effectively this can be accomplished.

For an enterprise like ours, there is a lot of complexity and diversity in the technologies and processes in the environment, which makes it extremely challenging to follow these high standards to the hilt. There is never a clear-cut black-and-white distribution of the overall environment with respect to the existing infrastructure, services, and data, which could make life easy when applying the expected standards. There are different levels of criticality defined for different applications based on parameters like the users of the applications and the sensitivity of the data residing on these systems. Based on this classification, different governance and compliance standards needed to be applied in these systems.

Depending on the industry an organization falls under, there are different compliance regulations that apply. Some of these standards are as follows:

- SOX Compliance
- SEC
- PCI Compliance
- HIPAA Compliance
- FERPA
- GLBA
- FISMA

SOX 404 or the Sarbanes-Oxley Act Section 404 was one of the standards that our organization was expected to comply with. The emphasis of SOX controls is mostly along the lines of protecting access to sensitive data pertaining to financial transactions. Although SOX compliance was already an area of high focus for us, managing the requirements required a good amount of work and was challenging. The overall focus of an audit is to evaluate the measures in place for the effectiveness of the controls defined by SOX.

Any findings meant that there was risk associated with the environments we were managing. Also, as these findings have to be reported externally for a publicly held organization, they had the potential to damage the credibility of the organization. Aditya and his manager Mike were very keen on eliminating any gaps in the environment and bring down the audit findings to zero.

Mike was a no-nonsense leader and was highly respected across the organization for his success rate with delivering on his promises. He was a leader who would make sure that he provided you with all support needed to deliver on your commitments and be successful in your endeavors. What this also meant was that this left no room for excuses by anybody later on. He would expect you to call out any challenges or roadblocks you were facing as soon as they were encountered and do his utmost to help resolve them. Mike and Aditya had put down eliminating SOX audit findings as one of their top priorities and committed to this as one of their goals for the next fiscal year. Both believed in putting their money where their mouth was and invested in forming a new team, IT Compliance & Governance (C & G), to help achieve the target of zero compliance findings.

The C & G team started their work by first putting effort into understanding more about the different controls that applied to our organization. They also started analyzing the challenges faced by the teams in the past to determine any shortcomings in the environment. Soon a pattern started to emerge. In a number of cases, the gap was not in the process that was defined for implementing the control. The gaps that emerged were mostly in the practice or execution of these processes, and this was because there was a dependency on humans following the defined processes. Although most of the time people did follow the defined processes, we had to eliminate even the slightest chance of the process not being followed. Not always is a human error because of bad intent. On the contrary, many times a human error occurs as people are trying to be more efficient by finding ways of doing things faster like providing a solution to a customer in the shortest possible time. Unfortunately, this sometimes results in employees not following all the steps defined for the process. This results in a violation of the set process and ultimately will be considered an audit finding.

A SWAT (SOX Workflow Automation Team) that had members from the C & G and the Automation team was put in place to eliminate the human dependencies in following the compliance guidelines. Joe led this effort from the automation side and was expected to collaborate with a number of other teams who were stakeholders for the defined processes. During their evaluation of the different processes, the C & G had also determined the extent of manual intervention required for each of the controls and the scope for automating those steps. The SWAT quickly got into action and started designing automation processes for each of the controls that were applicable. As the work gained momentum, most of the jobs were now controlled

by scripts operating remotely, and this resulted in tightening the access to different systems and applications for the admins who would have performed those steps manually earlier. The requests for elevated access to any of these critical systems was better controlled. All sorts of guardrails were being put in place by implementing monitoring solutions in the servers and application and integrating the solutions with the service desk portal to map access requests to these systems. In order to manage these automation solutions effectively, a web portal was designed by Joe along with the C & G. Sticking to the theme of bots, this portal was called "ScoBot," SOX Controls Bot.

The scope of ScoBot was restricted to a select few sensitive applications that fell under the SOX guidelines.

When automating a process, keeping the scope as small as possible is always a good approach since this helps in delivering the solution faster and more accurately. Measuring the impact of a new solution is also easier if the scope is well defined and well contained.

Eventually, with the help of ScoBot, the management of compliance requirements was simplified considerably and the environment was hence secured further. The solutions were slowly scaled up and covered a much larger landscape than initially was the case. The goal of eliminating the complexity associated with adhering to the compliance requirements was almost realized but for a few hiccups we would face at times, which would result in further enhancing the automations covered in ScoBot.

Summary

The mentioned bots were not the only ones that were built. This practice turned out to be contagious and influenced other teams. Some of these teams had started building their own bots and portals, which helped them make their day-to-day work more efficient. Teams that did not have the means or resources to build their own solutions would reach out to the automation team to help them with their requirements. In such cases, we would determine if these requests could be served within any existing bots or if it needed a new solution to be put in place.

The introduction of the bots in the environment was a crucial phase in the transformation, as the bots not only improved the efficiency of the overall operations but also helped in creating a culture of automation and innovation across the multiple teams.

Not all the bots survived the test of time, as some of them were more wishful than practical, fading away as their adoption rate and hence business value they were generating proved to be low. InfraBot and ITBot were the two bots that proved to be the most valuable of the first generation of bots. But these would soon be challenged with the more evolved solutions that would be built in the not-so-distant future.

The success with the bots helped create capacity in the teams and gave us the confidence to aim higher. We were chugging along on our journey toward transforming into an organization that operates on modern-day practices and technology. The next step for us was to rapidly adopt the technologies in the market that would help us compete with the best, and our adoption of Cloud was a big step toward this.

CHAPTER

5

Hopping on the Cloud

The momentum that started earlier had set the ball in motion and times were changing. Automation was now the flavor of the season. Many teams started realizing the potential of automation and were now more swift in their approach toward identifying and automating the areas that were plaguing their respective teams. Also, the automation team had developed a good reputation for itself by delivering high-value solutions. There was enough work identified for the automation team and the next few quarters were mostly blocked for the requests that were received from the different teams. More requests would pour in with each passing day. The automation team members, with their varied skill sets and creative mindset, were able to tackle requests for most of the IT functions. There were a lot of reusable modules that were built as a part of earlier deliverables that turned out to be of great value in implementing solutions faster.

Each member of the automation team brought something different and new to the table. Every individual had his/or her own approach and methods of implementing solutions, which made work enormously interesting and enjoyable for the rest of the team. A culture of helping others in the team reach the same heights as one has achieved had developed in the team. Teammates would make sure that the others in the team were succeeding and would provide all the support that was required to make that happen. A lot of cross-training happened quite frequently within the team where experts in one area would be training others in their areas of expertise.

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_5

Also, there were a lot of brainstorming sessions that would be conducted in the form of simple stand-ups or extended meetings in conference rooms. During these sessions, a lot of good ideas would be exchanged and discussed, which would benefit all the participants. I have always believed that *a sign of a healthy team is when debates are frequent, but conflicts are rare*. By these parameters, we definitely displayed the attributes of a healthy team and this trait of the team augured well for its further development.

CORE: Cloud Operations and Reliability Engineering

The team was enjoying exposure to a broad spectrum of work, solving the various kinds of challenges that came our way. But just like any other exciting story, a twist in the tale was soon to come. A new organizational strategy announcing the migration to Cloud from an on-premises datacenter for hosting business applications was revealed. The direct impact of this decision to our area of focus was not something the automation team had anticipated by any measure.

In order to meet the demands of the current times, our organization had decided to move ahead with a *Cloud-first* strategy. At a high level, this meant that all new applications would now be deployed on the Cloud instead of the on-premises datacenter, and a migration path would be created for the existing applications to move them to the Cloud as well. The long-term goal was to shrink the on-premises datacenters as much as possible. A decision as big as this sent mixed signals across different teams. The teams supporting the current on-premises environment were getting nervous. They were not sure what the future would hold for them if the company moved toward adopting Cloud.

An immediate need was felt for a team that would be able to support the Cloud platform. Not many resources within the existing teams had the skills and expertise to manage Cloud, and hiring external resources would mean a large investment. During those times, finding the right resources in the market in a short period of time who could support the demands of managing the Cloud was next to impossible. Cloud technology had just hit a purple patch in terms of its adoption. Cloud engineering and support-related skills were one of the hottest skills in the market. Getting good recruits would mean a sustained effort for a long period of time, and we did not have much time on our hands.

This is when a smart move by both Aditya and Anil turned out to be a masterstroke. This move would not only make the transition to Cloud possible but also make the Cloud adoption strategy an overwhelming success.

Aditya had done his homework on the Cloud adoption requirements for an enterprise. In fact, he had a head start on most of the other technical resources in the company in terms of understanding the usage of Cloud and any related technologies. He not only had done intensive research in terms of how other companies have adopted Cloud but also had registered for a personal account with one of the popular Cloud vendors and had started experimenting in the environment. After going through some of the best practices across the industry and understanding how other organizations successfully transitioned to Cloud, he paired up with Anil and played his move.

A tactical team was formed by hand-picking some of the best of the employees from various existing teams. The focus was to try and cover all the technical areas that would be needed for running a Cloud-based datacenter. I was lucky to be not only selected for this team but also given the privilege and added responsibility of leading this team of experts. At that time, though, it did seem more of a challenge than a privilege. Going ahead, there were just too many unknowns and challenges, whether the technology we were going after or the new members within the team I would be working with. Almost everything seemed new. All of a sudden, from leading a set of employees who had no more than three or four years of IT-related experience, I was now supposed to lead a set of highly seasoned professionals who were much smarter and more experienced than I was in most of the areas. This definitely added pressure on me, but at that stage in my career I was looking forward to a significant role and prepared myself to give my best shot given a chance. Apart from the existing members of the automation team, the team now comprised resources from the network team, the systems team, and the application support team. We also recruited one experienced Cloud and configuration management expert. I was now leading two teams, the CORE (Cloud Operations & Reliability Engineering) team and the automation team.

The CORE team was responsible for managing the Cloud datacenter. They were expected to manage all tasks related to supporting the Cloud environment, all the way from L0 to L4 (basic operations to advanced engineering requirements). As a part of our transition to Cloud, in collaboration with the education team, we worked out a training program to upskill the different teams on their Cloud knowledge. There were basic trainings offered for all employees across the IT teams, and the CORE team members were sent for advanced training once they completed the basic track. Working with the other experts in the organization, the CORE team had put the high-level as well as the detail-level design in place to address areas like the network segmentation, security requirements, and the application deployment needs. Reference architectures were being developed to help with migration of applications to the Cloud depending on the type of the applications.

Smog Around the Cloud

Very early during the transition phase, we realized that not all complications and challenges could be anticipated up front. While the Cloud vendors did offer a highly efficient platform, the onus was on us as Cloud users to account for some of the most important architectural components that were expected to be table stakes by the application teams moving their applications to Cloud.

The architectural components, which I also will refer to as **hazards** in a *burst and hops* model context, were as follows:

- Availability
- Resiliency
- Security
- Elasticity
- Scalability

I refer to them as hazards because addressing the challenges associated with the hazards would often slow us down in our migration to Cloud. The high expectations from the application owners meant that we had to think out of the box and wear a hat that was different from what we were wearing as a part of the operations team earlier for an on-premises environment. For instance, DR (disaster recovery) on Cloud was a gray and confusing area. There were so many different combinations of things that could go wrong in terms of the global distribution of the services offered on Cloud. We had a number of discussions just to understand what should be done to provide a DR capability on the Cloud. Each option that we discussed had its own set of challenges. While there was clearly a best way to implement a solution in terms of it being the most reliable or secure solution, not always can the best be defined from just one perspective. We had to offer the best solution in terms of addressing the requirements associated with all the hazards listed in the preceding and do this in a short period of time. Also we had to steer clear from the pitfall of overengineering solutions, as there was enough resiliency built into most of these Cloud offerings. In order to address this, we followed an approach that I later started referring to as the *burst and hops* model.

The Burst and Hops Model

The approach that got us moving ahead on Cloud migration while keeping an eye on the multiple requirements that needed to be addressed was to look at the criticality and the value associated with each hazard. We would often keep moving ahead with deployments to the Cloud, making good ground in a short period of time by focusing very little on the hazards. Our aim would be to make sure that we are offering a certain capability to our customers

within a short period of time and not slow down that process just to address the requirements that hazards would bring forth for us. The scope of these hazards would be different for different use cases. Also, depending on the application, the criticality of the hazards would change. For example, if an application was stateless, in the sense that it did not have dependency on storing data, it was not important to create a data replication solution for it. Similarly, the level of elasticity that was needed for different applications would be different depending on the variance of traffic that was expected. For any new migration to Cloud, we would analyze these aspects and create a delivery based on an initial **burst** (Figure 5-1).

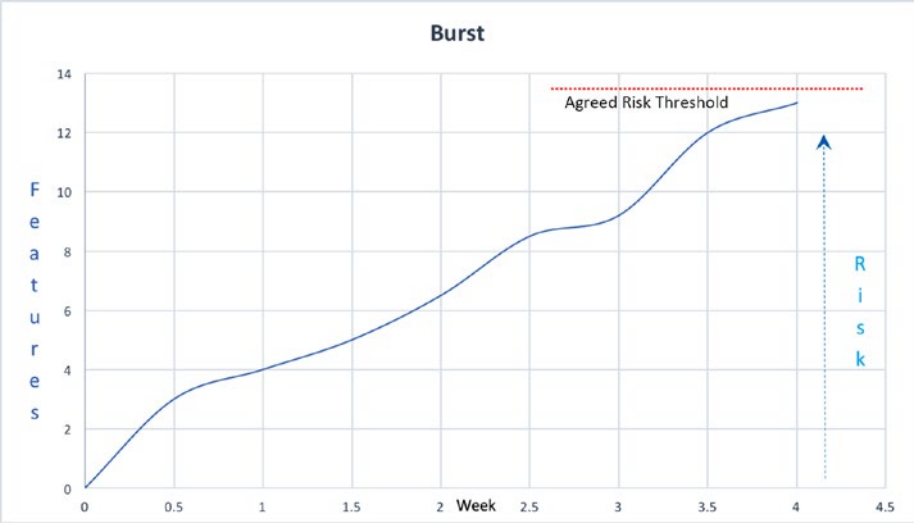


Figure 5-1. Burst

The burst would basically be a defined duration of time during which we would be going ahead with the migration of one or more applications to Cloud, giving little importance to the hazards along the way. This would enable us to define the scope of the migration upfront with high accuracy. The risks associated with ignoring the hazards would be called upfront by us. The scope of a burst addressing some of the hazard-related requirements would depend on what is an acceptable risk to the business on a temporary basis. A typical length of a burst would range from three to nine weeks, during which we would have migrated or deployed one or more applications to the Cloud. Once the deployment was successful, we would then start addressing the hazard-related requirements for the deployment to decrease the risk index associated with it. The scope of this activity would be very well defined in terms of it addressing a specific requirement of a hazard. This is what I called a **hop** (Figure 5-2).

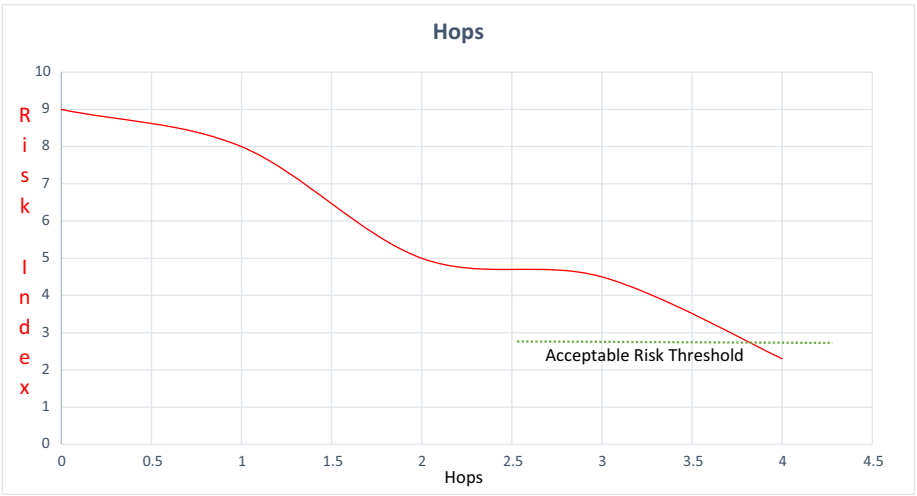


Figure 5-2. Hops

A hop would ideally last one or two weeks, at the end of which we would make sure that there was a significant decrease in the risk associated with the application based on the requirements of an associated hazard. This meant that the risk index of an application was basically an equation that took into account the length of the burst and the number of hops that were performed. Generally, this meant that *the shorter the length of the burst and the greater the number of hops, the lower the risk*. With each new deployment, we would create a reference architecture that would help with future deployments. This greatly helped in reducing the risk that would be associated with the future bursts as addressing some of the hazard-related requirements would now not be too complex since we had solved for it in one of our previous hops for a previous burst. Figure 5-3 shows us a series of bursts and hops. This process can be used for delivering a project on time and with good quality.

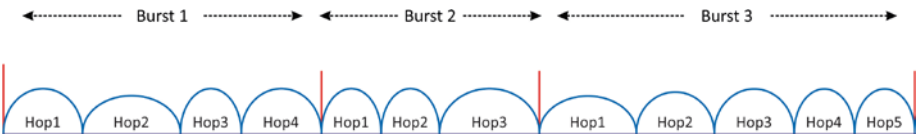


Figure 5-3. Bursts and hops

The burst and hops model was pretty much the de facto mode in which the CORE team operated for a long time without actually calling it that during that time. The idea of having a CORE team had clicked, and the migration to Cloud was mostly uneventful in terms of introducing any major service

interruptions. Managing the first few applications gave us good lessons, and in a short period of time we were able to perform a lot more migrations along the way optimizing the environment for better security, capacity, and governance.

CORE, from a second wing of the automation team, had slowly become more of a parent team and the automation team slowly got fully absorbed within CORE. The combination proved to be a bonus, as there were a number of opportunities realized for automating operations on Cloud and the team was able to churn out those solutions at a rapid pace. This process was further accelerated with the help of the extensive API support provided by the Cloud vendors. A number of operations were simplified with automation and were transitioned to the respective operations teams so CORE could focus on the next challenging task. The model worked pretty well, and soon CORE's involvement in Cloud operations was left to supporting only advanced requirements, as the simple and repeated tasks were moved to the operations team. This created bandwidth for CORE to continue with their exploratory and innovative work.

Migrating to the Cloud brought a number of advantages, most of which provided more agility for delivering our services. Before the adoption of public Cloud in our organization, a typical request for a new server would take at least a few weeks, and in some cases even months if hardware needed to be procured from external resources. This period would be very frustrating for the application teams, who were ready with their code and would be waiting on the servers to deploy it. Now, with Cloud in the mix, similar requests for servers would be addressed in a matter of a few hours. Eventually, we were able to deliver a fully hardened server addressing the requirements of the users in less than an hour. This kind of speed was unprecedented and really added wings to building the DevOps momentum for us, which by the way was picking up steam in the organization. What this transformation also brought along with it was streamlined processes and a transparency with respect to what can be expected upon submitting a request. This meant overall reduction in the time spent in supporting and troubleshooting the infrastructure area and in nursing customer needs. These were welcome side effects and helped a number of teams focus on further strengthening other areas of operations and eventually increase service efficiencies.

The CORE team had developed a good reputation across the organization, and most of the other teams now wanted to align with our work and get a piece of the new technologies we were working on. From being a team that was isolated in the past, we had turned into a group that other teams wanted to be associated with. This had its repercussions too. Some of the other teams felt that CORE was overshadowing what their teams were delivering and was also receiving a lot of attention from the management.

Because the CORE team was comprised of experts from various IT areas, a lot of times the skills required to deliver a particular solution were all found internally within the team. We would seldom reach out to the other teams for help. This further deepened the chasm that was being created. Concerns were raised by different teams and their leaders about how they felt the need to be included in the solutions being worked upon by CORE. As this concern started growing, I had to make some changes in the way we operated and make sure that we were more inclusive of other teams while implementing solutions.

We soon realized that this was one of the traits that a successful team had to incorporate in the modern world of DevOps. The more collaborative you are, the more chances you have for overall adoption and growth toward DevOps. Also, the additional domain expertise these teams brought in was tremendously useful. Since many of them were old hands in the company, the specifics of each of their areas were best known to these teams, and by collaborating with them, we were able to develop formidable solutions to the exact needs of the organization.

Celebrating Small Wins

In all the hustle-bustle, one thing we made sure was that we never forgot to celebrate the small victories. In fact, most of the time we did not need a reason to celebrate! There are a number of published articles that talk about why it is important to celebrate every step and every progress that is achieved in your journey toward a long-term goal. For me, the importance of celebrating small wins was instilled by both Anil and Aditya. Every time we announced the roll-out of a minor automation by the team, we would be immediately recognized by these two leaders. This would further motivate the CORE team to deliver more and would also encourage other teams to start working in an incremental fashion like the CORE team did.

The fun did not stop at just being recognized by the leaders. As a team, we would go out for luncheons and dinners when we felt we delivered something worthwhile. We would also have team outings at fun places, which would prove to be highly successful in building team spirit and would contribute to improving collaboration within the team. These events helped to ease the tension among individuals who had developed any kind of friction from being part of a high-performing team and subjected to the constant pressures that come with it. These events would also at times bring to light some hidden talents of the individuals and give them a chance to showcase it to their peers.

The time spent by the team members outside the work environment was very helpful in building comradeship. The individual personalities would be on full display during these events, and this would help the team members to learn more about their peers, whom they would otherwise look at very differently in a work environment. The bonds built and the trust formed during these times would result in greatly improving the teamwork and collaboration at work. This always made me encourage the team to plan such events and activities.

We almost had unsaid roles defined for the individuals when the time came to plan and execute these events. Vince among most others would make sure that the event was planned in an orderly fashion and that every detail was taken care of well in time. Joe would entertain us with his musical talents during the outing and also make sure that everyone had fun by creating a list of activities and conducting them successfully. Sid with his witty one-liners would bring in the humor.

I also remember one of our teammates then, Chris, who was very fond of food. Personally, it seemed to me that the scientists at NASA were wasting their time looking for black holes millions of light years away when there was one right inside my dear friend Chris's belly! Chris loved food and could eat quite a lot. We would jokingly say that Chris uploads all the calories to the Cloud and does not keep any storage on-premises.

Another memory I recollect of a fun team outing was when the team was playing a game of truth or dare at an offsite location. Being the daredevil that I thought I was, I had chosen to go with a dare and was instantly served a curve ball by the team. I sheepishly chickened out from the dare and as a punishment for that, was made to improvise a towel into a cape and run around the room pretending to be Superman. This dare for which I could not muster up the required courage was basically to make a phone call to my wife and tell her that I have had enough and wanted to walk out of our marriage! There was no way I was going to do that. I am still not sure if I was more scared of saying that to her and upsetting her or if it was my own fear of the response I would get after saying it. Either way, one thing I was sure of was that I would not be executing that dare. My manager, Anil, however proved why he was the boss and took it up. He lucked out, however, as his better half did not answer the call or so he said. We'll probably never know the truth about that one.

Anil had joined us as a midlevel management resource and was responsible for managing the infrastructure of our organization along with the different platforms built on this infrastructure. He swiftly started making his mark by identifying a number of loose ends and tying them up. He came in with a reputation of being extremely focused on removing any roadblocks that prevented things from getting done. He was perceived as an aggressive leader who did not hesitate in talking tough and taking even tougher actions when required

to keep things moving. He was a good idol to have, and I wished to emulate him in terms of his collaboration and delivery skills. He was one of the first managers who had a positive influence on me in terms of imparting knowledge of management skills. Until then, most of the lessons I had picked from my previous managers pertained to what a good manager should not be like! In that sense I have no second thoughts about saying that Anil was the first "leader" I had as my direct manager.

Anil and I were on the same mental wavelength on many fronts and connected well on what the end goal should be like. Our methods and thoughts often did not sync completely, but both of us accepted that. In fact, the reason some of the ideas reached fruition was because we'd have healthy debates and challenged each other's perspectives many a time. There were times when at the end of those debates I would end up wondering if I did the right thing by getting into these energetic discussions with my manager, but Anil made it clear that he never took those conversations personally. He always appreciated my passion and motivation behind those thoughts. I remember quite a few of his coaching moments where he taught me how to handle certain tricky situations to get the job done and how to balance aggression with compassion. He went on to advise me on how to help your listener understand your point and have him realize his responsibility toward enabling me in delivering solutions for the organization.

I have realized that for an organization to function effectively, diversity in all aspects is important. We have heard about diversity in terms of gender, region, and so on. Another important aspect of diversity we need is in terms of the way one works or what I call **aptitude diversity** (see Figure 5-4). This can be explained in the following manner.

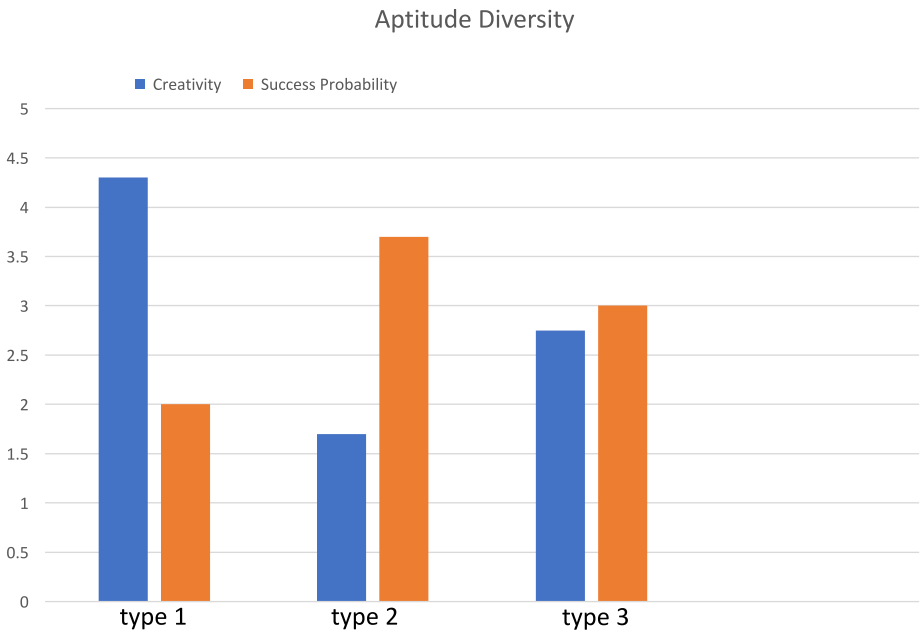


Figure 5-4. Aptitude diversity

I have always believed that the one thing that keeps me excited about my work is to have the freedom to innovate. I believe this is an inherent trait for some, and if this aspect of their professional life is not fulfilled, then there are chances that they will very quickly get demotivated about work and eventually either quit or fade away into oblivion and become a highly ineffective contributor to the team. Now, what having this trait means in terms of work is that if there is a requirement to be met and one of the ways of solving this is by delivering a solution that is extremely high on its creative/innovative quotient but is risky in terms of the guarantee in success, an individual like myself (depicted as type 1 in Figure 5-4) would perhaps still lean toward opting for it. There is definitely a higher chance of failure in terms of ROI for this approach, but it also means that at the end of the delivery there are a number of lessons learnt by the team involved that will be helpful in the future.

On the other end of the spectrum, we have individuals (depicted as type 2 in Figure 5-4) who are highly focused on getting the work done and would like to eliminate any risk that they feel could hinder them from delivering the solution. While this might seem the most effective option for an organization, often what happens is because the focus is so high on not failing, new approaches are not explored and the age-old practices are persisted with. In the long run, this could hamper the organization in evolving as a new-generation company and also could have a deterring effect on the creative individuals in the company.

Summary

I have just defined two ends of the spectrum based on only one of the dimensions in this multidimensional world of aptitude diversity. While not all personas that fall in this diverse area might be effective, there is no one persona that is a best fit. An organization to be successful needs a healthy mix of these different personas. And this is not something you need to hunt hard for while hiring. Humans by nature are varied, and in a given set of individuals you will always by default find a good mix of these varied aptitudes. What we need to do to achieve success with a diverse team is the freedom for the individuals to work by sticking to their aptitudes. If we force someone to adopt a different aptitude, you are challenging their inherent nature and that creates a lot of friction and often ends badly. This theory made sense when we applied it to the successful partnerships we would see in our organization between individuals. For example, in the partnership between Anil and me, one focused on delivery (type 2) and the other tried to squeeze in some aspect of innovation in the process (type 1). Also in this spectrum was Aditya, who was more toward the center, trying to balance creativity with guaranteed success toward the business delivery (can be considered as type 3), making sure he is not creating roadblocks for either of these personalities he saw in the organization.

CHAPTER

6

Mastering the Cloud

Adopting public Cloud was a major step in the transformation journey for our organization. The decision to move to Cloud was based on driving agility and optimization in the overall infrastructure operations space. But merely moving to Cloud does not guarantee these benefits. Cloud technology needs to be utilized in a highly organized manner to seek its benefits; otherwise there is every possibility that Cloud can become a cost and a security burden for an organization.

Early Days on the Cloud

Within just a couple of months of starting our journey of Cloud migration, we had been successful in migrating quite a few applications to the Cloud. Some of these applications were highly critical to the organization, and any outage on them would have a big business impact. In fact, the very first application we migrated was one of the most critical applications, which was the Single Sign On (SSO) application. The thought process behind picking this application as one of our pilots was that while deploying a complex application on Cloud we would encounter a number of challenges that would need to be solved. Once we were able to successfully deploy our SSO on Cloud, the next migrations would be somewhat simpler. This was because the challenges that we would face with the next migrations would have most probably been encountered earlier and we would have found the solutions to them.

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_6

The SSO application was serving more than 20 other important applications, and if anything went wrong with this service, most of the other critical applications would be affected. The stakes were quite high but we were quietly confident about delivering what was needed. The initial days after the migration to Cloud were rough and we did have a few hiccups in running the services in a stable fashion, but these challenges were not big enough to cause any major concerns.

We realized that a lot of our future adoption of Cloud was dependent on running this application smoothly on the Cloud and ensured that the team was available to support this 24 × 7. While the CORE team was responsible for enabling the platform to support this application, a bigger role during the migration was played by the Application support team. If there was any other team that was keeping pace with the CORE team in terms of learning new technologies and practices for successful adoption of Cloud, it was the application support team. This team consisted of members working from different parts of the world who were extremely talented and proficient in their areas of expertise. Most of the time, the first team that would be paged when anything went wrong would be this team. The individuals in this team were very prompt in responding to any issues and were ably led by Jay at one location and two other engineers, Abe and Tim, at the other location. Jay was considered one of the most hardworking and valuable employees on Aditya's team, and even though he was in a management role leading a team of more than 20 employees, he was still always the first to check in during emergencies and was able to lead the calls on the technical front.

Abe and Tim were more thinkers than engineers, and their job was to stay one step ahead in terms of emerging designs and technologies. They had an very good chemistry between them in terms of the approach they would take to problem solving and coming up with creative solutions to challenges. I always looked forward to seeing them whenever I traveled to their office locations. I have spent really good times brainstorming as well as partying with them. Our conversations would be quite weird in terms of the times when they would pop up. We could be having fun partying late in the night and suddenly the conversation would drift toward how some technology or design pattern had emerged and what we would need to do to start using it. The rest of the group would wonder what was wrong with us. With talented and interesting individuals like them, we felt that we could tear down any obstacle that came our way and have fun while doing that.

In a way, going with SSO as one of the first solutions migrated to Cloud was also a feasibility test for Cloud adoption. If we were able to run SSO successfully from Cloud, then it meant we would be able to run most of the other applications also as they were much simpler in terms of their architecture and other requirements. This was not the normal strategy for moving to Cloud adopted by other companies. Many companies would follow

the crawl-before-you-walk model by deploying the simplest applications to the Cloud first. But there were not enough references that we could obtain that would justify one approach over the other, and to us that did not appear the right way forward. We went ahead with what we felt was apt for us, and indeed while migrating some of the complex applications first there were a lot of lessons we benefitted from.

For example, we understood how to create a globally distributed architecture that would ensure that users from different parts of the world are not facing any latency issues in the applications deployed on Cloud. Also, some of these applications required an on-premises footprint along with their Cloud presence. For this requirement, we had to put up a design where some parts of the application would work on Cloud and the rest on-premises. Designing and executing each of these migrations taught us something more about Cloud that would be very useful in future.

Scaling on Cloud

The momentum around migration seemed to be gaining steadily. We were already building our roadmap of the next applications to be deployed on the Cloud while the first few applications were stabilizing. Every time a new application was deployed on Cloud, the operations teams made sure there were enough eyes and automation in place to resolve issues or outages as early as possible. For newly migrated applications as well as existing applications on Cloud, the teams made sure all related services were up and running at any given time and there was enough control and automation in place to make sure the best IT standards were implemented. This often meant keeping a constant vigil on all the changes that went in and having the ability to make sure that our processes and implementations were still relevant and optimized to the best of their capabilities.

However, with the handful of applications we had migrated to Cloud, we had our hands full deploying and supporting them. We realized that Cloud is only a viable option for running services if we could use it at scale in an efficient manner. Although by this time our team had grown with a set of five fresh college graduates it was highly impractical from a business point of view to expect the operations team to expand in proportion to the ballooning environment. The only way this seemed possible was to manage the increasing operational demands with the help of automation.

We started exploring the various options around automation. There were some references across the industry that proved to be useful and we started adopting those. Then there were certain requirements that we felt were not addressed by any solution readily available in the market. For these requirements, we started building our own automations.

Top Focus Areas for Cloud Management

From our experience so far with managing Cloud, we felt that operating on Cloud at scale had some fundamental requirements to be taken care of. The key focus areas that surfaced were as follows:

- Governance and security
- Cost and capacity optimization
- Backups, recovery, and reliability

Most of the operations and engineering work that has to be performed on the Cloud was driven by one of these three areas. The following are some details on the focus of each of these three items.

Governance and Security

Enforcing governance standards on Cloud is a major challenge. There are usually many hands operating on the environment and making changes to the environment. Not safeguarding the environment by putting proper controls in place means that there is every possibility that new vulnerabilities and risks are being introduced. Also, a number of business processes depend on following certain practices like tagging resources on Cloud properly and segregating the different application environments properly. Failure in managing this effectively could have a major detrimental effect on the quality of the overall Cloud environment.

Cost and Capacity Optimization

One of the major drivers for Cloud adoption is the agility that Cloud provides. But this agility could prove to be a two-edged sword if there are no proper capacity management controls implemented. Because it is extremely easy to create resources on Cloud, it often means that there is every possibility that people might go overboard in their zeal to create resources fast on Cloud and get careless with the cleanup required with wrongly provisioned resources as well as resources that are no longer needed. This is a very common trend that is observed with Cloud users and needs to be managed proactively.

Backups, Recovery, and Reliability

Most of the Cloud providers have extremely evolved options when it comes to backing up your environments and recovering from them. However, their philosophy is that they provide the means for you to take backups and recover from those, but you as consumers would still need to manage the schedules

of those backups, as most organizations have their own custom policies with respect to the frequency of the backups and the retention periods associated with those backups. Also, applying proper policies that safeguard the backups and periodically testing the quality of the backups is the responsibility of the Cloud users.

For addressing the requirements of each of the preceding categories, we built multiple automation scripts and would schedule them to run at specific times or trigger based on certain events that occurred. These scripts were initially written in Java and Python, which utilized the API suite provided by the vendors. We later evolved into using more of NodeJS for these needs, as that fit well into the web-based interface we wanted to integrate for these solutions. For a time, the scripts served their purpose and helped manage the scale on Cloud to a good extent. However, these different individual automation scripts built by the team were now themselves getting difficult to manage and monitor. While the team was churning out solutions at a high pace, we were faced with the challenge of managing these solutions. Their number kept on increasing and we were losing track of what solutions were in place. In order to address this challenge, we decided to start consolidating them into the high-level categories they served. As we had already identified these categories the solutions fit into, we felt it would serve our purpose well to group them accordingly. By doing this, we would be able to better manage the solutions and also have an understanding of the value these solutions were delivering.

Right from the onset of the Cloud adoption, the CORE team was inspired by a few companies that had managed to place themselves as the technology pioneers in the Cloud world. These were the companies who had been highly successful in leveraging the benefits of Cloud. They had done so by very innovative means. One such organization was Netflix. Netflix was able to run its huge business operations by successfully utilizing the full potential of Cloud. Netflix had mastered the art of Cloud usage by writing small automation scripts to address the individual needs of core operations on Cloud. They then started packaging these automations into a suite of solutions they called the "Simian Army" and had open sourced these solutions. The Simian Army was a bundle of smaller automation utilities that were popularly known as Netflix monkeys. Users could install each of these monkeys individually and use them to assist with their automation needs around areas like resiliency, security, and some others. We actively used two of these monkeys, "Graffiti Monkey" and "Security Monkey." We also used a few other open source solutions developed by other Cloud experts. Although not all of our needs were served by these solutions, they were very helpful in some of the areas. But, there was still a huge void in terms of the automation requirements we had for our specific needs and the solutions readily available on the market.

As I mentioned earlier, we were filling these gaps by building our own automation scripts and by now had built quite a few of them. Drawing inspiration from the free offerings such as the "Simian Army" and from Hollywood, our team started bundling these solutions into different packages, and together all these packages were called the "Cloud Dragons" within our organization.

Dragons in the Cloud

I developed the first codebase for the dragons based on the initial automation requirements that we had identified. The code was mostly written in Java, and I created the libraries for each of the dragons. Vince and I worked diligently for a few weeks to develop the solution. I would write the code based on the priority of the requirement, which was often decided based on discussions between Vince and me, along with some other members of CORE. Vince would do the deployment of these scripts and perform a thorough testing of the solution. Working this way, we were able to roll out quite a few solutions within a matter of weeks. As the team grew, our coverage of operations with automation kept growing as well. Once the new members joined us and were done with their share of training, they would be ready to start contributing. I transitioned the ownership of the dragon codebase to one of the new joinees, Paula, who proved to be an excellent choice for the job. She was very quick in grasping the current code and was almost instantly successful in adding a lot more capabilities and resiliency to the dragons.

We based the theme of these dragons on the popular movie, *How to Train Your Dragon* (2010). Based on the three categories we grouped these solutions into, we created three dragons:

- Penny-Pincher Cost Management Dragon
- Snaptrapper Cloud Governance Dragon
- Cloudjumper Recovery Dragon

Penny-Pincher Cost Management Dragon

One of my responsibilities as the product owner for Cloud platform within the organization was to ensure that the spend on the Cloud was highly optimized. The Penny-Pincher Dragon was focused on managing the cost and capacity on the Cloud. On Cloud, you pay for what you use, but if you do not put focused effort toward leveraging this behavior of Cloud, you could end up wasting a lot of money. We implemented quite a few solutions that helped us on this front. Each of these solutions was now packaged under the Penny-Pincher Dragon, and we continued to extend its capabilities with time.

Snaptrapper Cloud Governance Dragon

Snaptrapper was the name of a dragon we borrowed from the movie *How to Train Your Dragon*. In the movie, this was a multiheaded dragon who had a reputation as a fierce and aggressive creature if provoked. We picked this name for our compliance and security automations, as we had to look at governing the Cloud platform from multiple perspectives, which was symbolized by the many heads of the dragon. There were many stakeholders for the Cloud and we had to make sure that all the best practices and security measures were being implemented by all hands operating on the Cloud. We created a new role in the organization to help with creating policies that were required for governance around the Cloud environment and had intentionally not made him a part of the CORE team. We wanted someone from outside the team to keep an eye on how we operated. This employee would provide us the requirements for the features that would need to be added to the Snaptrapper.

Cloudjumper Recovery Dragon

One risk that we always felt threatened with on Cloud was if our Cloud account were to get hacked or if our data were somehow lost. This could happen due to multiple reasons, such as a Cloud admin not being vigilant enough about how he/she secures his/her access on the Cloud or an accidental action taken by an admin. Even though the likelihood of something like this might seem low, it does happen. And it happens a lot more frequently than you would think. There have been examples of companies shutting down or incurring heavy financial losses due to these kinds of activities. While Snaptrapper's job was to make sure that our Cloud environment was highly secure, we decided right from the early days of Cloud adoption that building in a recoverability solution in the Cloud from any disasters that could strike would be high on our list of to-dos. We were very clear that we wanted to build in enough redundancy in the Cloud services for ourselves and provide our customers a highly resilient environment. We came up with different flavors of automated backup to ensure that we were protected from such events. Also, to give us the confidence in our backup solutions, we would rigorously test our ability to recover from them. Cloudjumper would take daily backups of our Cloud accounts and create copies of the data and the configuration of the account. This data would be persisted at different locations, so we made sure that we would not get easily compromised. We automated our solutions to the extent where we would be able to recover from those backups at the click of a button.

Table 6-1 provides a detailed list of the features offered by all the dragons.

Table 6-1. Features Offered by the Dragons

Feature	Dragon	Best-Fit Environment
Smart Shutdowns	Penny-Pincher	Nonproduction
Detect and Eliminate Unused Resources	Penny-Pincher	All
Right-Sizing Resources	Penny-Pincher	All
Reserve Instances	Penny-Pincher	Production
Bulk Usage Discounts	Penny-Pincher	All
Configuration Backup	Cloudjumper	Production
Data Backups Based on Org. Requirements	Cloudjumper	Production
Automated Recoveries	Cloudjumper	Production
Automated DR Solution	Cloudjumper	Production
Financial Reports	Snaptrapper	All
Server Provisioning	Snaptrapper	All
Internal Chargeback	Snaptrapper	All
Compliance Audit	Snaptrapper	Production
Vulnerability Management	Snaptrapper	Production
Periodic Access Review	Snaptrapper	Production

The dragons would work all night and day ensuring that the Cloud is at its best at all times. They helped in creating capacity for the CORE team and the other Cloud operations team, and we were able to keep moving ahead with the next milestone we wanted to achieve. The dragons soon became quite well known and were heralded as one of the success stories around innovation for our organization. Our confidence was further bolstered when during our visits to Cloud conferences across the globe, we got to interact with some of the pioneers of the Cloud world. Upon exchanging ideas with them, I realized that we were doing pretty well with our Cloud services and were on the cutting edge of automation and innovation in the Cloud space. We took heart from all of the positive vibes we were getting for our innovations and kept enhancing our solutions and continued adding more features to the dragons.

Along with the many bots we implemented, the dragons were now adding to the operational efficiency that automation was bringing to the organization. We also built the solutions in such a way that they would complement each other and integrate easily when needed. For example, the dragons would be feeding off of each other's strength to provide more holistic solutions. Snaptrapper would make sure that all solutions were integrated properly with

the Penny-Pincher and Cloudjumper services and call out any discrepancies found in this. The data on InfraBot was helping manage some of the services that Snaptrapper required. All the features of the dragons were created as workflows and forms created in CA Process Automation and were exposed to end users through ITBot.

Often in the serious space of the corporate world, it helps to add some fun and color to the work. The dragons and the bots gave us a chance to do exactly that. The team was now not only writing code but was also thinking how they could get more creative with naming their solutions and designing logos for them. This was helpful in bringing down the stress levels of the individuals and provided a welcome distraction for them. Work was getting more enjoyable and that's not something you see often. Courtesy of the bots and the dragons, our team was now looked upon even more as the one forefronting the innovation wave. We definitely felt proud of this reputation that had built with time and were keen to hold onto it.

Cloud Requirements at an Enterprise

If you are working for a large IT or software organization, chances are that there are multiple teams having their own Cloud accounts and environments that they are managing themselves. The simple process offered by the Cloud vendors that allows the use of one's own personal or corporate card to start a new independent Cloud account is often taken advantage of by these teams, as they want to keep moving ahead with agility and try to avoid slowing down by internal processes of the organization. As a result of this, organizations end up operating multiple accounts with the Cloud providers. Over time, this could lead to inefficiencies in managing the Cloud services and also cause confusion, as the standard process is not followed.

We landed in a similar situation. While CORE was busy optimizing the one account that we were supporting, we learned that there were more than 50 different Cloud accounts that our company had been using for some time now. This came as a surprise to us. But it also provided us with an opportunity to share the practices that we were following to govern and optimize our account with these teams.

We reached out to a few of the account owners and had limited success obtaining details on the usage of the account. We did manage to get access to some of the accounts, and upon reviewing the design and practices associated with these accounts, we quickly realized that most of them needed to be optimized in almost all the three areas I have mentioned earlier in the chapter. For the first couple of accounts we got access to, we worked with the account owners within our company and were able to reduce the spending by up to 40%. Also, we were able to identify a number of improvement opportunities that would further eliminate chances of any critical security incidents occurring

in their environments and worked with the account admins to eliminate those risks. This bolstered our belief that the practices and solutions that we had developed for managing our Cloud accounts were highly beneficial and needed to be applied for all the accounts that were operating within our organization.

We worked toward consolidating most of these accounts under a single umbrella, which was overall governed by the CORE team. Small teams were put in place to work on each individual account to identify opportunities for optimizing and better securing them. This exercise was quite cumbersome as we did not have right access to the accounts upfront and had to work our way toward obtaining the access from their respective owners and their permission to scan through these accounts. At times there were apprehensions by some of the teams who managed their own accounts. They would be concerned that we might come in and impose certain processes and controls on their teams that would affect their efficiency.

While our intent was noble, we needed to improve the way we were being perceived and needed to find a better way to align with these teams. We decided that this could be achieved if we could be as unobtrusive as possible and empower the teams to manage their own accounts efficiently. Handing over automation scripts to them and expecting them to execute those effectively was not a practical way to achieve this.

We decided that instead of handing over individual automation scripts and expecting the account admins to run and manage these scripts, we would need to build an easy-to-use, self-help-styled solution. We started consolidating these individual automations into a Cloud solution suite that we called the CloudAscent solution.

CloudAscent: Serverless Approach to Cloud Management

CloudAscent is comprised of all the features that each of the dragons contained (refer to Table 6-1) and is designed as a SaaS-modeled (Software as a Service) multitenant solution. About the same time, AWS (Amazon Web Services), which was one of the primary Cloud providers that we used and was a market leader in the Cloud technology, rolled out a new set of services that they called the *serverless* offerings. The serverless design was to use a set of Platform as a Service (PaaS) resources to build an application. By doing this, you could completely eliminate the need for managing servers at your end, as AWS would be responsible for that. This was an extremely efficient approach for us as an IT organization, as it eliminated the need for running large datacenters. The nature of building a serverless application appealed to us instantly, and we went on to design our first self-service Cloud management solution, CloudAscent, using the serverless architecture.

The backbone of the serverless architecture was these individual compute services called AWS Lambda functions, which would be built as a microservice, providing a small functionality for a larger application. A new terminology was coined by the IT industry folks, FaaS (Function as a Service), to refer to services like Lambda. CloudAscent used as many as 95 different functions and was offered as a SaaS offering to all the different account owners within our organization to manage their Cloud environments. We worked in a typical startup mode during the entire implementation of CloudAscent. We defined pseudotitles for each of the team members working on this project right from a CEO to a VP of Development. The solution had its own internal website that allowed users to register and start using the solution. A screenshot of the website is captured in Figure 6-1.

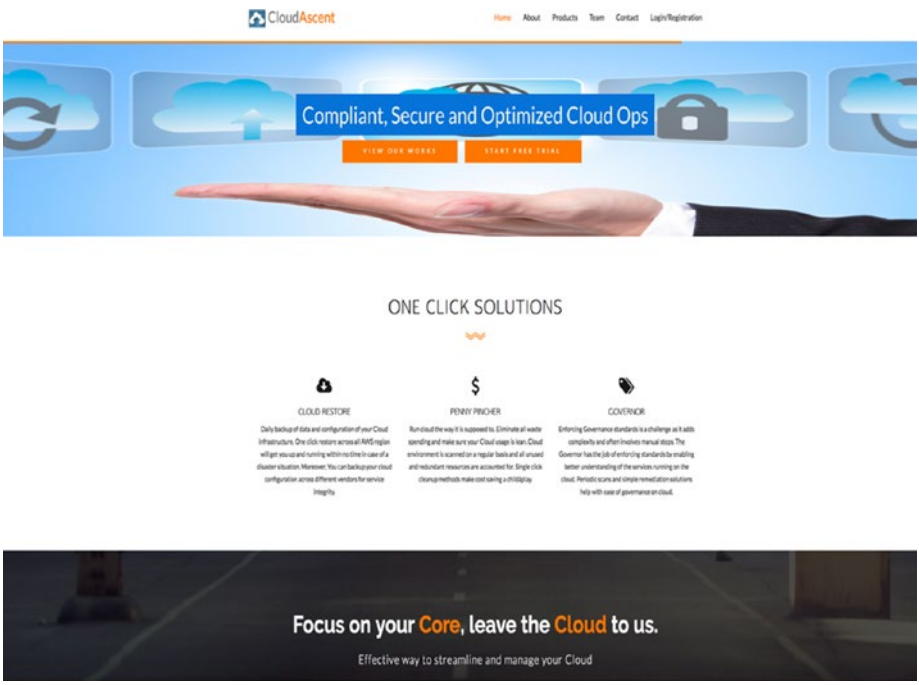


Figure 6-1. CloudAscent

We followed a number of startup practices like developing in small increments focused on delivering the MVP (most valuable product). Practices like frequent deployments with testing integrated at multiple steps during the process were also followed. We benefitted from internal users for this solution, as we were able to obtain the right feedback and direction from them in a timely fashion for the solution to mature. The MVP of our solution captured the most critical features that the dragons offered. We adopted the SOA (service-oriented

architecture) and built our solution writing one microservice at a time. By doing this, we were able to incrementally add more features to the application as and when they were ready without causing any disruption to the current functionality of the application.

With the help of CloudAscent, we improved our coverage and control of the multiple accounts in the organization. And by doing this, we made considerable improvement in the overall effectiveness of our team, which was responsible for managing Cloud platform for the entire organization. Not all teams using Cloud would enable all the features offered by CloudAscent. Each team had its own requirement on Cloud and they were able to pick and choose what applied to their environment. CloudAscent was an important milestone that we achieved in our Cloud journey, and multiple teams and organizations benefitted from this innovation that the CORE team delivered.

One important factor that contributed toward the success we were getting with our adoption of Cloud was having the right partners. Looking at the rate and scale at which we were adopting Cloud, our primary Cloud provider put up a support team and structure for us that was able to support our advanced requirements. A number of times, the actions we were performing on the Cloud were the very first time someone had ever tried them, and there were lessons for everyone involved in the process. We were unable to get the same level of traction from some of the other Cloud providers, and hence the adoption for those vendors remained low and they never really came close to the rate and scale of our adoption of the primary vendor. This is a critical piece of the puzzle during a transformation process. When companies start moving to Cloud, one of the reasons they do so is to benefit from the agility Cloud provides. If the partners in this journey are able to keep up the pace, then the process becomes that much easier and the partnership strengthens with each day. And if a partner is unable to provide the required support, then they will be left behind.

While CloudAscent was quite successful in optimizing the operational work for a Cloud account, there was another challenge that surfaced. There were a number of new accounts that were cropping up in the organization created by employees for their business needs and we did not have a proper framework in place to manage these new accounts. This meant that with each new account created there was the possibility of a nonstandard and potentially unoptimized account getting introduced in our system. While we fixed the gaps in the existing accounts, we now also had to ensure that any new accounts that were getting created had to be managed effectively. To address this, we introduced ClaaS.

ClaaS: Cloud as a Service

Since the CORE team had gained enough expertise in managing Cloud environment, we felt we were now in a position to address another critical requirement for our organization. Across the industry, the awareness and adoption of Cloud technology was growing, and with this, the demand for owning a Cloud environment by the different teams within our company also kept increasing. We knew that if we did not provide a service around this, there would be a lot of haphazard and nonstandard implementations of Cloud in the company and at some point we would be tasked with bringing this chaos under control. As it was, there were multiple teams managing their existing Cloud environments by themselves, and governing those was getting quite complex. We were adamant on making sure that the situation did not aggravate further and quickly formulated a solution for this.

We launched a service we called as ClaaS, which would enable employees within the organization to request a new Cloud account. A new website was launched with provided details on the service. A user would come to this site and fill out a quick form that captured information around that Cloud provider's account they were requesting, the expected spend on the account, and if the request was approved by their business heads. Once a request was received, we created new account templates that would be used to create a new account and would forward the details to the user. With this service, the user would also benefit by having integration with CloudAscent and would be able to manage their accounts effectively. We also offered more help in securing and optimizing their accounts by lending one of our Cloud experts' services to consult on the design and operational aspects for each of the accounts.

While ClaaS was helping solve the problem for new accounts, we put a roadmap to also bring the existing accounts in the organization under the common umbrella of ClaaS. This exercise was very challenging and took us about three or four quarters before we could get decent coverage of the existing accounts under ClaaS. Most of the time, the employees who were already managing their accounts felt that they were managing their accounts effectively and did not need any help. There were a few initially who let us in and provided us access to their accounts. We were immediately able to make an impact and in some cases brought down the spend by 60%–70%. These initial wins came in handy as references when we approached other Cloud account owners, and we were able to build credibility as Cloud experts across the organization.

CSS: Cloud Security Standards

Though CORE was implementing the security on Cloud, based on the industry best practices and from their knowledge and expertise in Cloud, there was still scope for improvement. We were not sure how to measure the security of our environment and determine the risk the environment was vulnerable to at any given point. This started becoming even more important as time passed on and more and more critical applications and services started moving to the Cloud. While we deployed a third-party solution to perform a continuous audit of our accounts based on the industry best practices, we often would run into queries by the application teams and by some of our leaders on certain security-related aspects that were very specific to our organization. The generic standards that the third-party solutions offered would not always provide answers for those questions.

In order to overcome this challenge, the CORE team partnered with the cyber security and vulnerability management teams in the organization and came out with a Cloud Security Standards (CSS). The CSS was authored specifically to take into account our company's IT policy, and it was based on the NIST Cybersecurity Framework. The CSS was developed keeping in mind certain aspects relevant to our IT needs. For example, it had guidelines on how to handle any sensitive data on the Cloud, the access management policy for Cloud, PaaS security guidelines, and so on.

To start with, we put in a manual process to regularly audit our Cloud accounts against these standards and determine the risks in the environment. Once the risks were found, we would work toward resolving them on priority. In the next couple of months, we worked toward automating this security scanning process in order to be able to perform these audits on a more frequent basis. As we were managing multiple accounts, a vulnerability that surfaced in one account would often surface in some other account. We identified these recurring kinds of vulnerabilities and started automating their remediation processes.

Having a baseline is very important when you are trying to assess any system. Without this, we would never be able to understand what quality parameter we are comparing against. Putting the CSS in place was a critical step for us in assessing the security of our Cloud environment. The CSS itself was a running document, as any time the dynamics around Cloud security changed because of vendor-side changes or any new risks that were identified, we would update the CSS.

Managing Multiple Uses of Cloud

Multiple Cloud environments in the organization meant that there were many teams who were using the Cloud technology. Each of these teams had their own need for or requirement from the Cloud. Some teams were running production workloads on Cloud, while others were using Cloud for testing and nonproduction purposes. There were a few teams from nontechnical functions such as sales teams and education teams who would use the Cloud temporarily for demos or as labs. With all these varied use cases, we realized that a single governing system would not be ideal for such a wide user base. The various governing standards that we put in place that assessed the security, capacity management, and configuration management practices for Cloud had to take this aspect into account as well.

In today's world, one of the prime parameters for measuring the effectiveness of a service or a solution is the UX (user experience) it offers. Even though it is important to put controls in place when trying to impose standards or better secure an environment, you need to empathize with the users and make using the service a pleasant experience. For us, providing Cloud services to the different customer bases, a one-size-fits-all kind of a solution did not make sense. We had to customize our solutions based on the type of Cloud usage we were catering to. Securing an environment that supports critical production application would need a lot more stringent policies than securing a noncritical environment. Applying strict policies could affect the UX, as it often comes with extra controls and restrictions. While the users of a production environment might be alright to live with those constraints, if the same policies were applied to noncritical environments, we would be creating a lot of dissatisfaction for the users. To avoid this, we came up with the idea of identifying separate personas of Cloud users and mapping each persona with a specific set of governance and security standards that would apply on them.

We called this as the Enterprise Cloud Governance Framework, and it went hand in hand with the CSS I mentioned earlier. The framework would propose a set of questions that the Cloud users would need to answer for themselves. This would help identify which category their Cloud environment would fall into. Once the category was determined, then we would understand the compliance and security standard required for that environment. We also came up with another standard that would determine the risk a Cloud environment was prone to if certain guidelines in the CSS and the Enterprise Cloud Framework were not met. With this approach, our influence on the environment owners improved because they would be warned upfront about the vulnerabilities to their environment. It would not be in the best interest of the account owners to be aware of risks and not act upon them, and hence they would comply with the recommendations that CORE would provide. Thus, with the help of these processes and frameworks that we put in place, we were able to drive efficiency and accountability in the Cloud management segment.

Summary

The adoption of Cloud is a major transition for any organization. The success of this transition depends on many factors that are both technical and cultural in nature. Having the right team led by the right set of individuals happens to be one of the most basic requirements. As the Cloud provides agility and flexibility inherently, it has the potential to scale up very fast. The challenges associated with scaling on Cloud need to be addressed in a timely manner. Automation happens to be at the forefront of scaling up on Cloud. In any midsize or large organization using Cloud, multiple teams can end up with their own Cloud accounts. A comprehensive insight into these different uses as well as a common strategy across the organization for Cloud adoption helps in reaping the full benefits of Cloud. The organization I was working at had a lot of these things in place and hence was able to make steady and stable progress in its adoption of Cloud.

CHAPTER 7

Innovate or Perish

At different phases during the transformation journey, there were situations and instances where the need for innovation in day-to-day work conducted within the organization stood out. Following the age-old practices without causing any disruptions to set processes seemed like a comfortable and a safe approach for most. The momentum driving innovation within the organization could use some acceleration, as it would help challenge the status quo and could drive improvements by challenging these set practices. Though there was innovation happening in certain pockets, there was still a lot of ground to be covered on this aspect. Innovation as an inbuilt DNA trait needed to be further inculcated in teams and in individuals.

Creating a culture of innovation and sustaining innovation practices needed a carefully thought-through strategy and a high level of commitment by employees at all levels.

> *We cannot solve our problems with the same thinking we used when we created them.*

This quote, sometimes attributed to Albert Einstein, perfectly defined the challenge that the organization was going through. I remember an incident during the early days of Cloud adoption in our organization that illustrates this. Automation was still gaining momentum and we were identifying use cases that could have a significant impact on the day-to-day work of the employees within the company.

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_7

Rip and Replace

During a visit to one of our global offices, I was invited to a meeting where a group of employees was going to discuss a chronic issue, "*improving the quality of CMDB*," that was plaguing our organization's service quality. CMDB typically becomes the backbone of a number of processes, as it is considered to be the master source of inventory and the configuration of the items within the inventory. Any asset present in the datacenter is expected to be captured in the CMDB. This data is then used in a number of processes to drive overall operations.

Attending this meeting turned out to be a novel experience for me, one that helped me understand the current mindset of some of the employees. At that time, my understanding of CMDB and its significance to the organization was limited, hence I chose to be a silent participant for most of the meeting. I sat my chair listening to others in the room during this revealing discussion among six or seven colleagues who either owned CMDB or had some stake in contributing to its quality. The discussion was mostly around applying bandages to the existing process rather than addressing some core points that could bring major improvements to the process.

I walked away from the meeting pondering what I had just witnessed. I had very little to contribute to the meeting at that time, as I felt I had limited knowledge to recommend anything yet and also was not sure how to share my honest opinion around what I felt about the process to the individuals who built it and lived with it for a number of years now. Based on the discussion I sat through, I felt little hope that there would be any major improvements we would achieve if we stuck to manual processes. There were definite opportunities I identified to automate the process during the discussion and I was determined to bring improvements to CMDB by understanding more about the current process and CMDB's role in the organization.

As my understanding on CMDB grew, it appeared that if broken down properly, the process to manage it could be systematically automated by implementing the following steps:

- Identify the different data sources and extract data from these sources to feed into the database through scripts
- Ensure there are controls put in place where the only option is for a human to enter the data
- Constantly validate the data in the database against an asset's (configuration item's) current state
- Update any delta/gap found in the data automatically in real time

While defining the process was easy, shifting the mindset of the employees and having them agree to this proposal was a task in itself. We had to roll out our solution in a phased manner, which moved at a snail's pace. We would face challenges like getting access to the end systems to update the data through automation, and we would be constantly questioned about the quality of the data we were pulling from the systems and had to prove our solution's reliability multiple times before gaining trust from the stakeholders. We implemented most of the solutions addressing the preceding steps, but getting a nod from the respective process owners and the stakeholders was extremely challenging and would slow us down. We ultimately did deliver a solution that made a considerable impact in improving most of the quality challenges in the CMDB. There still were some pieces that needed to be addressed, which we did in due time.

The preceding example illustrated the importance of sometimes giving the process a complete overhaul and not applying bandages to things that are beyond repair.

Building an Innovative Team

The CORE team size had more than doubled after we hired a bunch of fresh college graduates to help us with the increasing demands of automation and Cloud engineering. The key skills we focused on for recruiting CORE team were as follows:

- **Proficiency in at least one programming language along with sound computer science fundamentals**. The programming language often talked about by the interviewees would be either Java or C++, as these were the two languages that were a part of the curriculum in the local universities.
- **Analytical mindset**. We often shortlisted the candidates on this skill by giving them simple problems to solve that would help us understand their approach to problem solving and the aptitude the individuals displayed when posed with a challenge.
- **General technology awareness**. We would be very interested in learning if the individuals we were recruiting have done any research or projects apart from the course offerings in their colleges or universities. This would help us gauge their passion and also help us in the process of shortlisting candidates from a large pool of individuals that was provided to us by our recruitment team.

This process of recruiting worked well for us, as the first set of five candidates we hired turned out to be extremely proficient. Not only were they good in their technical knowledge but they brought a burst of fresh energy into the team. They were very keen to get started and were highly inquisitive about how things worked in an enterprise and had an undying hunger for learning more about the areas they would be contributing to. There was buzz around the office corridors with the new recruits in place, and they would leave a positive impression on almost everyone they interacted with. Without much effort, we were quickly able to identify their individual strengths and mapped them accordingly to the work areas they would be focusing on.

Innovation, a Survival Skill

When you are leading teams of highly talented individuals, it is essential that you nurture and harness their creativity!

One of the strengths of the CORE team that set us apart from many other teams was our ability to adopt new technologies. A common challenge with adopting new technologies is the dearth of useful references to help understand the technology and how to adopt it. We were able to overcome this challenge by thinking out of the box. The team would conduct brainstorming sessions and implement small, fun projects using the new technologies that would help them learn. Once there was enough understanding of the technology, it would then be applied to solve business-related challenges. This approach became quite infectious in the team and almost everyone was getting trained in using the creative part of their brain to a good extent. I also believe that if we keep innovating regularly, a momentum is created that automatically fuels more innovation, and this is what CORE was experiencing. A number of other teams were also in the process of adopting newer technologies in their respective areas. They realized that they needed to up their game with respect to successfully delivering services using these latest tools and solutions available in the technology space.

Earlier approaches such as engaging the services teams from the vendors or reading the manuals of the products was just not sufficient any more. The application of these products to solve real-world business problems was more important than appreciating the long list of features offered by the solutions. While an attractive GUI (graphical user interface) for an application definitely helps, the real engineering and integrations these days are happening behind the scenes. Most of the modern-day products offer APIs or CLIs (command-line interface) that help in integration with other systems. There is a basic level of coding expertise that is needed to be able to make effective use of these offerings. The individuals and teams that realize this are successful in making a fast-paced transition toward building more holistic solutions that

are well integrated with all the required touch points and are not limited to operating at a task level.

Integrating systems with each other is a key step when we are trying to automate processes end to end. To be able to achieve an optimum state in terms of automation, it is important to look at the process from multiple perspectives and focus on finding creative ways to solve the problems effectively. The teams that are able to do this thrive as they are able to move beyond the mundane tasks that eat most of their time and look at bigger challenges. Teams that are still caught up in the trap of keeping the "lights green" and are not innovating tend to be left behind in terms of staying relevant in the organization as well as keeping the team members motivated.

Innovation helps an organization flourish, as it can affect various aspects of the business. At times, global organizations need different versions of processes to be implemented across the different global regions they operate in, and this needs out-of-the-box thinking. A practice that works with the culture and the governance rules within one region might be totally unsuitable for a different region. Also, with the right innovations, an organization can lead business transformations by entering new fields sooner and can capture markets early. Companies expanding based on acquisitions of other smaller companies also require a lot of creative thinking apart from great business acumen. Considering multiple perspectives to a possible acquisition opportunity helps in determining the long-term benefit of a merger or an acquisition. Innovative ways of keeping customers engaged in the build process and in the sales cycles are proven ways of driving customer loyalty and generating long-lasting revenue streams for a company.

Often, innovating is misunderstood to be a characteristic associated with elites and can be confused with a quality displayed only by a genius. This kind of thinking adds a lot of pressure on people and acts as a road block when driving innovation in a company. The closer the masses feel they are to a quality, the greater the chances of adoption. We need to help people realize that a lot of times they are already innovating in their daily lives without putting too much thought into it. For example, parents often come up with their own creative ways to manage their children when raising them. There is no set practice to bringing up kids, and each family has come up with their own ways to make the process as simple and as efficient as possible. Within a family, too, what worked for one child might not work for another, and they have to keep evolving and thinking out of the box to handle many situations that arise in this process. Innovating at work is no different.

Another misunderstanding associated with innovation is that innovation means building something big and new that needs to have a huge impact. This is notion is wrong and needs to be dismissed. Innovation is often identifying the small opportunities and making each step a little more efficient. It is about

observing closely the minute details of a process and evolving toward building solutions that make the process that much easier to execute.

Frameworks and technology are mere tools to achieve the desired state: the real value lies in the thought process of the individuals designing the solution.

ShakeUp: Ideation in Action

Considering all of these perceived benefits of innovation, it is essential that an organization should always strive to drive a culture of innovation by making innovation one of its core areas of focus. Our organization was also beginning to realize this and started investing toward driving innovation as one of the DNA traits of the company. Several programs were introduced at different levels and there was a lot of mentoring and coaching provided for individuals to start developing their innovation quotient and bringing it to their day-to-day job. Innovation had slowly become a quality that was no longer expected to set you apart but more of a table stake. If someone were not trying to innovate, then he would definitely stand out, but not for the right reasons.

Design Thinking

A common practice that the CORE team followed when they were faced with a new challenge, such as the one I mentioned about CMDB earlier in the chapter, was to get the team in a room and start brainstorming the different ideas regarding how to solve it. With a large team bringing with it different mindsets, these discussions would be extremely interesting and beneficial, as we would start to understand the challenge from multiple perspectives.

In order to drive a culture of innovation in the company, Aditya encouraged me to enroll in a course, "*MIT's Approach to Design Thinking*," offered online by a private institute. The course was aimed at imparting the following lessons to the course-takers:

- Understand the design thinking process
- Identify and assess customer opportunities
- Generate and evaluate new product and service concepts
- Design services and customer experiences
- Evaluate product development economics

Using some of the techniques I learned from the course around design thinking, the team and I put in a process to drive innovation. We started conducting a recurring ideation session where the entire team participated. One of the motivations behind setting this session was to harvest any hidden

ideas that the individuals might be carrying without realizing the potential. We were hoping that these sessions would also be helpful in triggering interesting team discussions that would inculcate a practice of listening to ideas from others and building on them to enhance the ideas further. Within a couple of iterations, we defined a format for this session along with means of capturing the ideas in a template (see Table 7-1); we called it PEP, which stands for Proposal - Edge - Plan.

- **Proposal** would focus on projecting or pitching an idea to the team and helping them understand the challenge it solves.
- **Edge** would focus on determining the business value of the idea as well as the novelty of the idea. It would also provide details around the advantages the idea had over any existing similar solutions.
- **Plan** would give an idea about how the solution could be turned into reality by providing details on the effort estimations and any other budget requirements.

Table 7-1. A PEP Template

Proposal	Edge	Plan
My idea is …	The business value is …..	This is how I think we can implement it…
	Idea is unique because ….	

Apart from generating great ideas, these sessions also helped as a team-building exercise, where the junior members would get a chance to learn about how to connect ideas to the business to add value. Also, the sessions helped the seasoned members of the team to come out of their narrow view of the world they had been working in for the past years and think beyond it. The sessions acted like a catalyst that woke up the hidden innovators within the individuals, and hence we started calling these sessions **ShakeUp**. The name resonated well with the motive of ideation and soon caught on across the organization. We started expanding the awareness of this by inviting members from other teams to join us in these brainstorming sessions and encouraged them to take the learnings from the sessions back to their teams.

A number of ideas that we implemented as solutions to solve business-related problems were either birthed or matured in these ShakeUp sessions. We would share the PEP of an idea we thought was interesting either with our leaders or with teams we thought it would be beneficial for. Later we moved away from PEP to a more industry-wide format for pitching ideas, **lean canvas**. We had come to know that lean canvas was the preferred choice

for accepting ideas by some innovation programs in our organization and was much more encompassing in terms of capturing details about an idea than the PEP template we formulated. Aligning with lean canvas automatically helped us in taking our ideas further if there was potential in the idea.

Lean canvas is a template that is adopted by many across the industry to drive ideation and pitch ideas at different forums. The lean canvas focuses on critical aspects of capturing and explaining an idea like the problem, the solution, and the uniqueness of the idea.

Running an Ideation Program

The innovation wave caught on across the organization and a number of teams started their own ideation sessions. Not every team was as organized and as effective as the CORE team in terms of getting value out of these sessions though. Running an ideation program is not easy. Some of the skills I had acquired around design thinking concepts would come in handy during these sessions. There are certain aspects that need to be well thought about to make an ideation program effective and to motivate the team to keep contributing and not feel disengaged from the program.

The typical challenges that one faces when running an ideation program are as follows:

- Determining an effective format for conducting these sessions.
- Running the program in a way that the majority of the team participates in the proceedings and that they are not dominated by a select few who are outspoken.
- Capturing the ideas effectively.
- Understanding the value of the ideas generated and tying them with the business.
- Most important of all, bringing an idea to closure.

It is not necessary that all or even the majority of the ideas should be implemented to determine the success of an innovation program. The most important aspect is to keep the participants motivated and not make them feel that their ideas are not going to make any difference and that they are just wasting their time by participating in these sessions. People are bound to be very passionate and emotional about the ideas that they put forth, and it becomes hard for them to fathom the fact that their idea is not being shortlisted for taking it further. In a business setup, it is critical to have an honest dialogue around an idea with the idea provider(s) and to determine

along with them if it actually adds value to the business. This would help in the decision-making process, which decides if this idea is something that needs to be pursued. Also upfront, it is important to make it clear what the objectives of these sessions are and what kind of ideas are being sought after. If you can execute these parts of the program successfully, chances are that the team members will not lose motivation and will continue to have faith in these programs. This would mean the innovation bandwagon keeps moving and the organization keeps benefitting from it.

Ideation Framework

With all the experience we gained from running the ideation program for CORE, we formulated a framework for scaling innovation in our organization captured in Figure 7-1. We defined four key steps to drive innovation and identified a set of ceremonies that would be needed to ensure the success of each step.

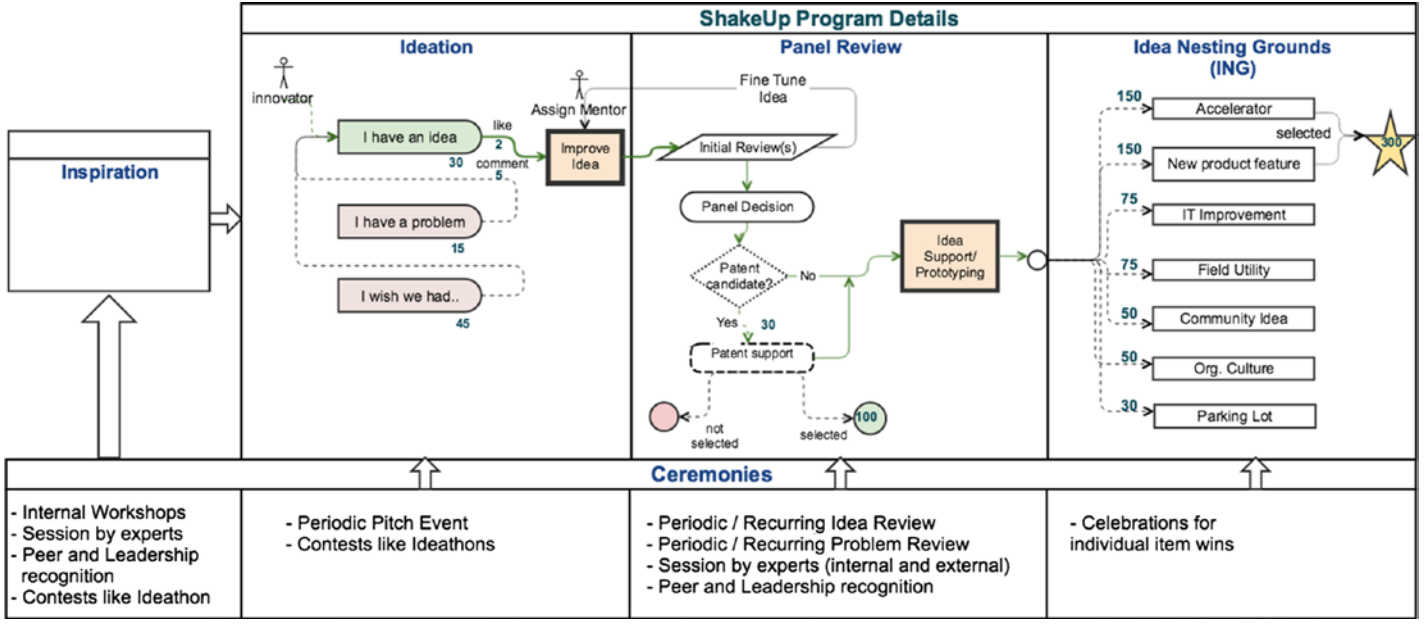


Figure 7-1. ShakeUp framework

The four steps in the ideation program are the following:

1. Inspiration
2. Ideation
3. Review
4. Landing Ground

Inspiration phase is focused on creating an atmosphere of innovation and ease for the employees to feel comfortable in bringing forth their idea in front of others. Creating this comfort zone forms the most basic step, as without achieving this, the participation in the program would be very low. The other objective of this phase is to create an awareness among the employees about the importance of innovation and why each one of them had to participate in this initiative.

We identified a few ceremonies such as tech talks by experts, motivational speeches by leaders, programs to recognize innovative workers at different levels, and contests such as hackathons and ideathons to drive the success of this phase.

Ideation phase was where the idea capturing would take place. We created a web portal to drive the ShakeUp program, and on that portal we created means for people to either enter a well-formulated idea, or state a problem or a wish that they hoped someone would solve for them. We virtualized the lean canvas by converting it into a webform on the portal, and the ideas would be captured in the lean canvas format.

The ceremonies associated with the ideation phase were conducting more hackathon-type events as well as having recurring pitch events in place where people could walk in and pitch their ideas to a review panel.

Review phase was where the ideas would be processed. We anticipated the fact that not every innovator would be able to compile a good lean canvas to project their idea effectively. We only mandated two areas of the lean canvas out of the total of nine areas captured in the lean canvas. A panel was formed that consisted of employees who demonstrated innovative thinking in the past to nurture the ideas. The expectation from the panel members was to review the ideas on the ShakeUp portal and work closely with the idea providers to further mature the idea. In the process, the panel members were expected to leverage a set of SMEs from the domain the idea belonged to (for example, an idea could be around improving cybersecurity standards, in which case an SME from the cybersecurity space would be engaged by the panel). Multiple rounds of discussions would be conducted around an idea. The idea and the idea provider would benefit from this practice, as different minds get in different perspectives and the idea keeps maturing during this process. Also, all involved in this process would collectively further enhance the lean canvas.

Landing Ground phase was basically where the life cycle of an idea with respect to the ShakeUp program ended. We identified multiple areas where an idea could add value and called them the "idea nesting grounds" or the "landing grounds". The decision on what landing ground an idea would end up was made during the review phase, and this would be a collective decision between the idea provider and the review panel. However, an idea provider could chose not to follow the recommendation of the panel and could still pursue a different landing ground for his or her idea if he or she wanted to. The role of the panel is not to become a checkpoint but more of an enabler. Once an idea is mapped to a landing ground, the scope of the program would end, as the idea would be handed over to the appropriate landing ground owners. From there onward, the landing ground owners would work directly with the idea providers and put up a plan around the implementation of the idea.

In order to make this program more interesting, we also introduced a bit of gamification in it. For each activity a user performs on the portal, he would be awarded points. Activities that could be performed included adding an idea, stating a problem, "*liking*" an idea, and adding a "*comment*" on an idea. Also, when the idea passes through each phase, it would be awarded certain points. This would enhance the level of engagement in the idea throughout its life cycle. The motive of embedding a point system was to be able to identify the top innovative minds in the company and reward them in different ways. For example, gamification could be used at different team levels to recognize the individuals for their involvement in driving innovation in the organization. It could also be used as a means of associating monitoring rewards to individuals who are contribute innovative ideas. Another use would be to generate healthy competition between peers to top the points table. While the use cases are plenty, it was more important to make this framework available in the solution so the different teams could use it in a way that suits them the best. We also accounted for identifying patentable ideas to filter out ideas that would add to the intellectual property of the organization and also provide the required recognition to the individuals with exceptional ideas.

The program structure was appreciated by many teams, and most of them wanted to participate in it. We started driving the overall innovation in the organization based on this framework and were successful in creating a buzz around innovation initially and in sustaining the initial hype by following the practices defined in the program. We also faced challenges in certain areas of the program such as keeping the momentum going, motivating the individuals to keep participating in the program, reviewing the ideas on time (which needed a lot of support from the panel members), and most importantly, getting the buy-in from the different landing ground owners to take an idea from ShakeUp and add it in their backlog to get the real value from the idea.

Summary

In today's world, for a business to succeed, innovation can no longer be considered optional. Innovation helps organizations stay competitive and keep delivering solutions that are relevant to changing times. While innovation has now become a table stake, it is not easy to sustain. A concerted effort needs to be put in, and it will need support at different levels in an organization. It always helps to put in a framework to drive innovation, as it will help the individuals know what to expect after participating in it. Applying practices like design thinking to real-world problems opens up doors to solving these problems efficiently by innovative means. Building teams with the right skills and mindset can be very valuable to the organization and is definitely something that should be invested in by every organization.

CHAPTER

8

Evolution of Teams

As the process and methodologies being applied at the workplace started evolving, the roles and the compositions of the teams also started changing. The operations teams were challenged with matching up with the demands of the transformation in the organization. New skills needed to be developed and new approaches had to be defined. Even though it was not an easy thing to do, many of the staff members were up for the challenge and started putting serious efforts toward staying relevant. This was an extremely positive development for both the organization and the employees. All involved were making sure they were fighting hard not only to stay relevant but to be pioneers in the process of IT support evolving in this new era of DevOps. As a result of this, along with the other business functions, the recruitment process was also evolving. The expectations around skills and aptitude from new hires were now different.

With quite a few new recruits joining us directly out of engineering school, the CORE team now had a sizeable number of members in its ranks. We had put up a customized training program to make sure that these new team members with almost no work experience were provided the right direction and skills to enable them to complement the teams they were going to be a part of. A major focus area during these training sessions was on honing the coding skills of the individuals. The training content for these trainings would be based on practical applications in the IT field to enable the individuals to be prepared for addressing the real-world problems they would soon be facing.

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_8

Easing into a New Role

I believe that when new employees join your team, regardless of how experienced they are and what their professional background is, helping them understand their new role and what is expected from them becomes critical to their successful integration in the team as well as the organization. I have seen employees who have joined teams that are functioning well as a unit still find it difficult to adjust, as they could end up feeling like outsiders trying to break into a very tight group. Having clarity on the role and the deliverables can help individuals create a place for themselves, which in turn helps them build their confidence to finding footing in a new environment.

The new recruits, especially the fresh college graduates, were partnered with an expert in their teams in the initial days to help ease them into their new roles. They would shadow their seasoned partners for a while in order to get exposure to the different technologies used by the teams as well as understand the intricacies of how the organization worked and the different processes and practices their teams were involved in. A number of training sessions were conducted by the many teams in the organization. At any given time, these teams would be working on multiple tools and technologies; training team members on this would provide them with a very good opportunity to learn and widen their knowledge base.

Of course, not every plan we made always worked to perfection. The challenges around understanding the problems on the ground and coming up with effective ways of optimizing those processes take a lot of time to master. The relevant technical skills individuals come in with and the time they invest in acquiring the skills required to make them successful in their role are important factors for the individual's as well as the team's success. To be successful in a role focused on driving automation for the organization, it definitely requires having the right aptitude and mindset. The passion of the employees and their dedication toward building the right skills and having a positive mindset will ultimately determine their efficiency.

Modern-Day Team Outlook

One of the primary challenges that the CORE team faced was being able to pick the right process to automate. This was challenging because most members in the team were fairly new to automation and identifying opportunities to automate required some practice in this space. Only a select two or three in a team of ten-plus members were adept in this art so far, and we had to marshal the rest of the resources in the right direction. This used to be a bottleneck initially, but with time the other team members were able to learn the tricks of the trade and were able to work independently. With the team having enough head count and with each member having their own strengths

and skills, we were able to take on larger initiatives that were far more varied in nature.

For example, instead of just trying to automate testing reliability on a datacenter's storage replication, we planned on how we could automate a full DR situation. Automating this had a much larger scope than automating the replication processes and would cover many other major steps in the process. For delivering solutions like this, we had to work more closely and for extended periods with members from other teams. CORE was now expanding its areas of influence across different functions true to its name, Cloud Operation and Reliability Engineering.

Success with Reliability Engineering for us meant that we were creating an atmosphere where there was trust in the organization's ability to deliver solutions that were both efficient and resilient.

The CORE members were gaining rich understanding of the IT world by interacting with the experts from the other teams, and in return these domain experts were learning new tricks of the trade in terms of automating their day-to-day work and were also now starting to think of optimizing processes on a larger scale. The interactions we had with these team members were mutually beneficial, as there was something to learn for both sides. The concept of optimizing work through automation, especially managing infrastructure through code, was alien for a lot of them, and these interactions helped them get a deeper understanding in these areas.

Collaboration within teams improved when they sat together across the table, brainstormed the next challenge at hand, and worked toward the common goal of delivering the best solution. Having all the stakeholders involved throughout the solution development process was very important to achieve the desired value out of any solution being implemented. This practice not only helped in building the right solution but also in a way ensured that the solution would have a good adoption rate, as no one would feel left out or surprised. A few such exercises meant that the mindset of multiple teams was now getting positively influenced. Individuals as well as the teams they were part of now wanted to be the ones leading the innovation wave in the organization. This change in the work culture was helping CORE be more effective, as this was making our job much easier to execute. We no longer were facing the high level of resistance from other teams for helping automate pieces of their work. In fact, teams were now lining up solutions they wanted to partner with CORE to automate.

With many teams focusing on optimizing their work through automation, the transformation toward a modern-day DevOps-driven organization was well underway. The system admins who previously knew only of a single means of performing their respective duties, which was by logging into a server and

making a change, now started evolving their approach. New techniques were being explored and creative solutions were being built.

Automation Center of Excellence (COE)

There were a host of technologies and tools that were at our disposal both from the product catalogue of our company and from the open source market. We did our due diligence in understanding what our requirements were and how we wanted to operate, and based on this we adopted the right set of tools to support us. The direction provided to us from our leaders was to not hesitate in taking risk and to be undeterred by failures. The term that they often used with us was to be ready to "fail fast." This meant that if you found something interesting enough to pursue, go ahead and satisfy your curiosity but make sure that you do not invest in it to an extent where you burn down too much time, energy, and resources. This helped in satisfying the innovation zeal among the teams.

The CORE team operated as a COE for automation, helping drive the automation efforts in all the other teams. Multiple teams would reach out to us to brainstorm on optimizing processes managed by their teams. As a COE for automation, CORE was forefronting some key initiatives in the organization. These activities included the following:

- Training and upskilling other teams
- Advanced automation with infrastructure as code (IAC)
- Leading the transformation process
- Creating reference patterns for breaking down processes before automating them

The remaining sections of the chapter are aimed at explaining these points with some more details and a few examples.

Training Teams on Automation

Multiple teams placed requests to the CORE team to provide training on implementing automation. We did our bit by conducting internal trainings for teams on relevant technologies such as scripting, automation/orchestration tools, Cloud, and so on. Any time we adopted a new technology to help with automation, we would make it a point to conduct internal cross-training sessions for other teams so they could benefit from those technologies as well. Every individual had his/her own learning curve. The most challenging trainings for the teams were those around a scripting or a programming language.

Even though most of our colleagues in the organization were from a computer science background, they had been in the IT world working as admins for a long time and either never had the chance to write any code or had almost fully lost touch with writing code.

IAC: Upping the Automation Ante

Having achieved success in most of the automation initiatives we had undertaken, it was time to raise our game up a notch. We always had the support and backing of most of the leaders, and now with the employees in the field on our side as well, things were looking propitious, which gave us the confidence to aim even higher. We no longer felt the need to have processes implemented manually by admins.

The number one reason why our IT environment was in a not-so-optimal state was because of humans. Humans are bound to be error prone as they get distracted easily, can get physically and mentally tired, and tend to be careless.

By automating solutions, we put in a lot of effort to clean up the environment. Performing the same task multiple times on multiple systems opened doors for poor-quality implementation. Performing these tasks the same manual way all over again did not make sense. Our drive to ensure that the quality of the environment remained top class pushed us to get everything managed and administered with the help of code. We had identified a few use cases for ourselves that would drive the IAC model for us. These activities were the ones that were most time consuming, as they comprised repeated tasks. Having the ability to perform these with the help of code and in most cases implemented as self-heal solutions strengthened our move toward automation and minimized human dependency, which in turn eliminated human errors.

We put up a configuration management framework that involved installing an agent on all the servers to help us with automating the config updates of the systems. Any changes required on the servers could be delivered by means of writing code and executing the code with the help of these agents sitting on the servers. By doing this, we hoped that all configuration changes on servers would be performed from a centrally managed system, which would enable us to standardize the environment and also assist with rollback and troubleshooting. We ensured that enough awareness of the tools used was created and also offered multiple trainings to different teams so that they could start leveraging these new services available to aid them.

Taking the Lead in the Transformation Process

Although not all members of the different teams were able to keep up with the pace at which new technologies were being adopted and the processes were evolving, there would be at least one or two members in each team who stood out in terms of evolving with the changing times. They became flag bearers for their respective teams when it came to transforming manual processes into automated solutions. These employees had some particular traits that set them apart from the rest. They believed that automation "actually" yielded results, could be trusted, and was their friend. These employees were not afraid to bet on what seemed like the unknown if it provided hope for something big. Perhaps, it was "unknown" for others but an informed and intelligent decision for them. The most important of the traits they possessed was the ability to understand the big picture. This helped them easily comprehend details of processes like the touchpoints, integrations, impact, and the optimization opportunities. Once these attributes were identified, the process of automating a particular solution became simple. This would open up doors for them to start working on converting a process to an executable code.

These individuals became major influencers in the organization, as they were trusted members of their teams and were able to get their peers excited about the promise of automation and code. Employees were going online and getting self-trained on the required technologies. The education team in our organization, which was responsible for getting employees trained in the right tools and technologies, was now receiving requests for providing the training for all these new technologies that our teams were adopting. Everyone wanted to deliver some automation and get noticed, which was a win-win situation for all.

These were the times when we first saw signs of transformation taking effect. The system admins who were earlier used to logging into a system console and updating the configuration changes were now taking an evolved approach. They were now looking at the automation frameworks in place to drive these kinds of changes in the environment. They would partner with experts from the CORE team to help author the appropriate scripts to perform those changes using a common configuration management system. While not everything worked as expected the first time these scripts were executed, the teams would perform incremental updates to the scripts at times while testing them on the lab machines to ensure the quality of the solutions. Once tested thoroughly, it hardly took us any time to execute those scripts on the production servers regardless of the number of servers this code needed to be executed on.

Breaking Down a Process for Automation

Success with one such solution would do wonders for the confidence of the system admins and also would boost their confidence in automation and code. The leaders of the teams would also make sure that they encouraged and recognized the members in their teams who made an effort to challenge the status quo and drive transformation in their teams. Slowly but surely, the transformation was gaining momentum and was sweeping everyone in its way. The admins were no longer looking at traditional ways of working.

For instance, the process for deploying a security patch on a set of servers had evolved to a new process. Most of this process was automated and it made the task of patching servers extremely easy to execute. The process consisted of the following steps:

1. Running a script first to discover the right group of servers where the patch was applicable
2. Deploying the patch on a set of test servers
3. Validating these test servers after patching had completed
4. Running validation scripts on the production servers to make sure they were running fine before the patch was deployed
5. Proceeding with patching the production servers
6. Running the validation scripts again on the patched servers to make sure there were no regressions introduced in the environment by installing the patch
7. Running a security scan to make sure the vulnerability had been remediated with the patch installation
8. Incorporating notifications in the process to keep the right parties informed of the changes and the impact

We were no longer operating with an average system admin team working only toward keeping the ship afloat. We had now an evolved team working with us manning the IT fortress, making sure that they were thinking beyond the present by planning for introducing long-term resiliency. An outage on a service that was caused by an "out-of-memory" scenario no longer meant that the resolution for it was to assign more memory to the server. These employees were now working as true engineers who were hell-bent on understanding the root cause for the memory peaks in the servers and delivering the "correct" solution, keeping multiple perspectives in mind. The right steps to troubleshoot and resolve the issues were all being scripted and tested thoroughly.

The support from all the other functional teams in the organization had a major role to play in the success of this transformation. Teams such as application support, network operations, and IT service management were all aligned in supporting each other and driving the IAC concept. This was proof of a well-known theory that *if all forces are accelerating in the same direction, then the velocity is highest*.

Summary

It becomes important for an organization that all employees understand the direction where the organization is headed and align themselves in this process. The roles and responsibilities of all the teams need to be active participants in the transformation process to make it successful. There is a natural evolution that takes place for all involved in the journey, and the ones who survive are the ones who take up the challenge of transformation and look at it as their ticket to taking their careers further.

IAC is an important means of achieving operational efficiency for infrastructure teams. There are quite a few tools, both open source tools and solutions provided by tool providers, that can be used as the foundation for driving IAC in a team. But the most important factor is the adoption of this approach by the employees. IAC can seem highly complex for sys admins who have not been used to writing scripts for the majority of their career. Having these admins understand the need for adopting this approach and then providing them the right training as well as the environment to ramp up their skills can become the biggest hurdle. Easing the adoption of IAC on them by exposing the admins to simple automations before embarking toward adopting it is a good way to ensure that everyone is onboard and ready to contribute.

CHAPTER

9

Accelerating Towards DevOps

In the previous chapter, I talked about how the teams in the organization were moving toward adopting the IAC model. Once we started thinking in terms of driving all infrastructure changes through code, we came to the realization that the use of code is not limited to managing infrastructure-related operations only. Because of the inherent capability code provides with respect to scaling, standardization, integration, and automation, we were more and more attracted to embedding code-driven implementations in as many areas of our work as possible. This gave birth to the concept of "Everything as Code."

Everything as Code

Ben had joined our team as a Cloud admin and had spent the last three years of his professional career managing and administering Cloud platform. Most of this work of his involved him operating on the Cloud using the user interface consoles provided by the Cloud vendors. He seemed to be in a confused state of mind when we talked about locking the access to these user interfaces and operating on Cloud purely using CLIs or by means of writing code to leverage the rich APIs available for those platforms. He was even more surprised when he saw the size of the team that was currently in place to manage the huge scale of Cloud services at our organization.

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_9

He mentioned to us that at his previous employer, the size of the Cloud environment was close to 30% of what he saw at our organization. And then he added something very interesting. He said that the size of the team managing that environment was close to three times that of the team managing Cloud at our organization. This was music to our ears. We always took pride in the efforts that we had put in place to optimize our Cloud environment, and what Ben had told us validated our belief that we had achieved an extremely evolved state in the work we were doing. It took Ben some time to realize why we insisted that he sharpen his coding skills and he was extremely swift in aligning with the practices we followed to go about our work.

Scaling on Cloud can only be achieved by proficient developers. If a system admin can write code, then there is no better fit for the role of a DevOps engineer.

Another story I want to mention is that of Mary from the change management team. She was responsible for making sure that the practices defined for change management were adhered to by all teams. Her team mostly consisted of employees with little to no coding skills but who were very well versed with the requirements for an enterprise to meet the compliance standards. By nature, these types of job functions in the organization had no integration points with any development teams. But these were different times. Mary's team and the CORE team had partnered on a number of initiatives and her team had developed a good sense of what can be achieved with coding.

Most of the manual work was now in the process of getting automated. A number of solutions that we had built for the change management team required integrations with multiple systems. Data exchanges between these systems was a common requirement. Initially, this data was collected and aggregated manually by Mary and her team by gathering inputs from multiple systems. Although this was an age-old practice and had been working for a long time, it was neither a scalable nor a reliable approach. Especially because these solutions were for addressing the requirements for compliance, there was absolutely no scope for any errors.

Joe and Mary countered this challenge by coauthoring simple scripts that pulled data from these different systems, translating them into ingestible formats and orchestrating workflows to make the solutions as human intervention free as possible. Different data formats were relied on but the most popular were json and csv (comma-separated values). This approach had empowered the nontechnical teams to start contributing to the development work that was in full swing with respect to optimizing the IT operations. A teammate of mine, Henry, had termed this approach as the beginning of **Everything as Code**.

Henry's thought process behind adopting this term was that he believed the most evolved state of DevOps is when all communication between the different stakeholders happens in the form of code. The exchanges would evolve from raw textual artifacts that contained requirements and plans

toward configuration and data files of a mutually agreed-upon format that would be ideal to fit into any service orchestration platforms that were in place. By doing this, the value coming out of these interactions between teams would be enhanced multifold and would automatically help align the work to the DevOps way of execution. We ourselves were nowhere close to achieving this but it did help us in defining for ourselves a goal to work toward.

Culture at the Center of Transformation

The fact that employees like Mary and Ben were starting to transform into believers in the DevOps philosophy and were driving it within their respective teams was a big achievement for our organization. Various teams and individuals displaying these traits augured well for the future, as the seeds that were planted a couple of years ago were now bearing fruit. The thought process of the people was changing. DevOps demands a change or evolution in the culture of the organization, and the change in culture becomes the most important parameter to gauge the success and value driven out of DevOps.

The way one needs to think about the culture of DevOps is to separate the technology piece of the transformation from the process or execution part. Teams within an organization are expected to become proficient in practicing the following:

- **Working toward Common Goals**
  - Understand the different value streams and align all work toward the organizational strategy
  - Make business value-driven improvements
- **Agility in Execution**
  - Understand the pulse of the requirements during execution
  - Create customer feedback loops
  - Constantly steer the solution toward changing customer requirements
- **Innovative and Creative Thinking**
  - Ability to step aside from the process and think out of the box
  - Not get influenced by set practices while designing solutions
  - Apply design thinking practices to create efficient solutions

- **Displaying Courage and Risk-Taking Ability**
  - Ability to try new things, risking failures
  - Reward risk taking
  - Fail fast and move on
- **Decentralizing the Command Center**
  - Democratize decision-making without inducing delays in the process
  - Leave experts to take calls in their areas of expertise
- **High Trust**
  - Always believe your colleagues to have positive intent
  - Don't shy away from sharing any breakthroughs you have achieved
- **Open Organization**
  - Eliminate "me" and adopt "us" in discussions
  - Candid and straight talk to eliminate assumptions and misinterpretations
  - Adopt other problems as your own
- **Identifying the Right Problem to Solve**
  - Deep understanding of where the real problem is in a process
  - Understand the value driven from solving the problem and determining the ROI
  - Apply lean practice before automating or solving a problem to reduce the scale
  - Quantify success by determining measuring parameters upfront
- **Continuous Learning and Knowledge Sharing**
  - Keep abreast with the latest trends and technologies
  - Create training platforms that are able to deliver new trainings in a swift and effective manner
  - Create platforms where teams can cross-train and create a chain effect in terms of spreading the knowledge

- **Frequent and Incremental Deployments**
  - Modular and loosely coupled designs
  - Compact and shortened release cycles
  - Automated integration and deployment

Technical Practices to Drive IT Home

Once there is clarity on how the culture fits in the successful adoption of the DevOps framework, the focus needs to shift toward applying some of those practices to determining how it affects the technical practices. Often, cultural practices are not quantifiable and the means of overcoming this is to understand how the culture is driving the execution on the ground level. Effective execution is mostly a result of practices put in place and the technology involved.

The technical practices that are highly conducive to successful adoption of DevOps can be listed as follows.

Security Embedded Upfront Reduces Risk

The significance of a highly mature threat management system in an enterprise can never be overstated. For too long, security has been treated as an afterthought, and this not only introduces undue risk to the organization but also acts as dead weight to the entire system. If security practices are not thought about upfront in a system, then people find their own ways of trying to either apply security at different stages in a totally broken fashion or totally ignore their responsibilities toward securing the environment. Either of these cases is undesirable and will have significant consequences to the well-being of a company. Thus, it is very important that security features among the top three items to consider when designing or introducing a new system.

With more and more companies now opting for a mixture of IaaS (Infrastructure as a Service), PaaS, and SaaS offerings, the reliability of the Cloud providers in embedding security practices is on the rise. But this by no means relieves an organization of its own responsibilities toward securing itself. The Cloud providers will only provide means of securing the accounts, but the security design has to be prepared as per the requirement of the organization. An even bigger challenge that follows is the execution of the security plan. The cybersecurity team plays a critical role in defining the standards for an organization and has to lead the way in ensuring that the processes and checkpoints in place are extremely efficient. Proactive scanning, continuous audits, and a well-defined remediation path for all threats need to be in place for a successful security strategy. By having all these checks in place, the rolling out of DevOps practices in an organization becomes that much easier and well integrated.

Effective Cloud Adoption

Cloud has not only deeply penetrated all industries but it is leading the way in terms of enabling a highly efficient IT organization. The inherent traits of Cloud like API support, integrated environment, and agile nature are conducive to DevOps adoption and transformation. Actions like infrastructure provisioning and purging, which were earlier thought to be complex tasks and acted as bottlenecks, are extremely easy to execute on Cloud, and this by itself has increased the turnaround times of IT teams multiple times. Adding integration with systems that enable continuous deployment and automated testing makes Cloud highly desirable for an organization that wants to adopt the DevOps framework.

Microservices, a Perfect Fit for DevOps

Similar to the Cloud, microservices is another technology that was created to promote the DevOps way of work. Small features developed on different technologies and deployed in a continuous fashion on container platforms have simplified application deployment by leaps and bounds. The portable nature of containers and the ease of scaling the services on a container platform are directly targeted at the key teachings of what a DevOps environment should be like.

Slightly evolved but not as mature as the container technology, Function as a Service (FaaS) is also quickly catching up. FaaS provides us with most of the benefits of what a container-based microservice provides and also adds to its advantages by removing the maintenance of infrastructure. The Cloud providers take the full responsibility of providing the required resources to execute this function, so as a result the end user gets unlimited resources that are also cost efficient, as you are only charged for the duration of the function.

API-Driven Solutions for Easy Integrations

A big part of DevOps is about how one is able to optimize its delivery chain. This requires a lot of integrations between the different components found in a typical DevOps chain. From planning all the way to development, security scanning, testing, integrations, and deployments, there are multiple tools and systems involved in the process, and each of them needs to be able to talk with at least one more tool in the process. Adopting solutions that expose their services with a comprehensive set of APIs becomes key in automating the delivery chain.

Begin and End with Testing and Test Everything Else in Between

Every step in the DevOps process needs to be validated and tested thoroughly. Within a process, error handling, failover, and rollback are all dependent on how extensively you have embedded the various testing and validation practices. The following are the tests that need to be considered in a mature DevOps environment:

- Code quality test
- Code security test
- Unit testing at functionality level
- Integration, build, and penetration tests
- Performance, regression, and scalability or stress tests

A good DevOps process will ensure that most of these testing requirements are met. Having these tests ensures that the solutions delivered are robust enough and can be relied upon.

While automating solutions, it is also important that the failure path is well defined, as this keeps the chain functioning smoothly.

Adopting the Burst and Hops Model for Agility and Quality

Agility plays a key role in defining the value you are able to generate from your DevOps processes. The burst and hops model that was introduced in Chapter 5 is a good framework to follow when you are looking to enable DevOps. Burst and hops allows you to quickly create a workable solution that will help give clarity toward the value being realized with the bursts with your solution. With time and a few hops, you will then be able to improve the overall quality of the solution. In case you realize that the bursts are not giving you much return, then you can look at modifying the solution or building a new model.

Adopting the burst and hops model allows you to fail fast, which is a critical attribute of successful DevOps implementation.

Completing the Circle with Automation

We have talked about how the beginning of transformation at my organization was jump-started with effective automation. Automation forms the backbone for the operational efficiency that can be achieved by adopting DevOps. There are various stages of automation an organization evolves toward during its journey in automation, and the effectiveness of the automation differs at different stages. The evolution of the automation journey for our organization is captured in the Figure 9-1.

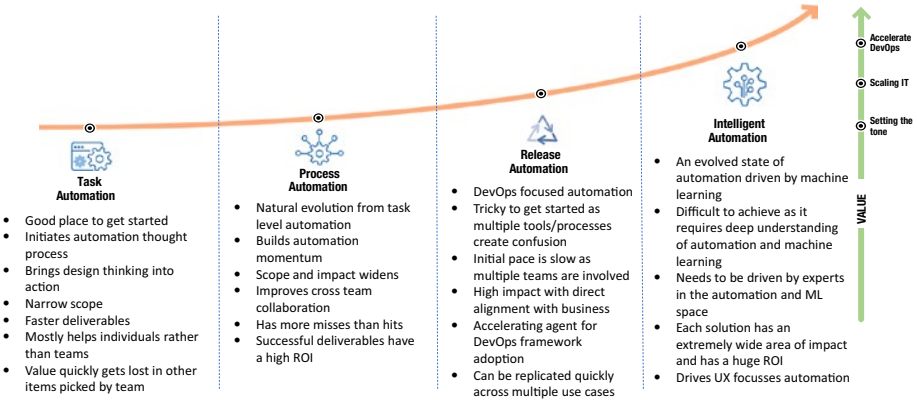


Figure 9-1. Evolution of automation

These different phases of automation can be explained as follows:

- **Task/Micro-Level Automation**

  This was the earliest stage in our journey, in which we would identify individual tasks to automate. These tasks would often be the slowest or the weakest piece in a bigger process. By automating at the task level, we would make the overall process a bit faster than it would take to complete normally. While this kind of automation has a good value, it often misses the larger picture and can be considered as a bandage rather than a remedy.

- **Service/Process Automation**

  This evolution from a task-level automation to a service-level automation is a very natural process. In this automation, instead of automating one piece of the process, the process itself comes under that radar of automation. For

this kind of automation, it is important that the overall effectiveness of the process is well understood, and we need to be open toward modifying the original process flow itself. Automating a process provides us a chance to have another look at the process and make it leaner or more effective from the previous lessons. Once the process flow is determined, then all the tasks within the process are fair game for automation. Automation at a service or a process level yields much more return than automating at a task level.

- **Release Automation**

  When we talk about DevOps, then release automation cannot be far behind. In fact, there are some schools of thought that define release automation as 75% of what DevOps means. While I am not of the opinion that if one has a good CI/CD (continuous integration/ continuous deployment) process in place then there's not much left to do in terms of DevOps, I do believe that release automation is one of the most critical and high-value items that needs to be ticked before one can claim victory over DevOps.

  At a high level, release automation has a common flow for most of the technologies and stacks, but during implementation of release automation you quickly realize that there can be so many variants of this automation, which depend on the technologies and platforms in use.

- **Smart/Intelligent Automation**

  After conquering the earlier stages of automation, we had earned our right to start evaluating the next level, which is intelligent automation. Intelligent automation is basically using the advantages of concepts like machine learning and artificial intelligence to increase the benefits of automation. For instance, while it is good to have an autoheal automation that falls in the service-level automation, it still is a reactive approach. With intelligent automation, we try to achieve a state where we are able to predict and prevent an error in the system. This not only eliminates the need to automate but more importantly ensures that there are no service interruptions or outages.

Getting started with intelligent automation is often quite complex and it needs a good amount of investment along with a visionary thought process to be able to look into the future and tie it with the events in the past to make the present better.

These stages of automation can be leveraged by an organization to understand the current state of automation it is operating in and what more remains to be accomplished.

DevOps Is for Everyone

An important thing to remember when we talk about DevOps is that there is no team or individual who should be left out or not touched by DevOps. Participation is needed from every team, and every team will be affected by DevOps in some way or another. For instance, if the engineering team is creating a CI/CD pipeline to automate the deployments more frequently, the teams that are automatically affected are the change management teams, incident management teams, testing teams, and support teams like the service desk team.

In the software deployment process, when you consider changing any of the steps, whether designing, planning, testing or deployment, there is both a trickle-down and a bubble-up effect to the steps that lead up to and follow the deployment, and hence the impact is felt by most of the teams in the organization. There are also instances when as a result of moving toward DevOps, certain tasks of some teams might get eliminated, and this often causes a panic in those teams. What is needed by the teams in situations like these is to look at this as an opportunity to broaden their scope and pick up those fringe or complex items that they never had the bandwidth to take up before. When teams are successfully able to transform toward working in this mode, then the negativity quickly turns into a positive feeling and starts benefitting not only the individual team members but the organization as a whole.

To illustrate the preceding statement with an example, when the CORE team optimized the process of provisioning and asset management for our organization, they initially eliminated the work that the platform operations team used to perform. There were at least five operations admins who were kept busy by the influx of requests by application teams for new infrastructure. These admins were provisioning the servers manually and spent most of their time on these requests. Once this work was automated and transitioned to the applications teams themselves as a self-serve solution, the operations team was able to look at more complex tasks like Cloud resource configuration and security improvement–related tasks. This gave them a chance to upskill themselves, with the chance to become an expert in an in-demand technology.

If the admins were stuck doing the provisioning, then they would have been deprived of a great opportunity to build on their current skill set. Having worked on such advanced tasks on Cloud, quite a few of these admins have since been able to further their careers in Cloud as architects both within our organization and externally. Every so often, I do get an opportunity to interact with these individuals and they seem to be doing great in their respective roles and have now become evangelists and champions within their teams and organizations of modern practices of DevOps, whether automation, Cloud usage, or something else.

Summary

The cycle of DevOps begins and ends with automation. While there is a lot more to DevOps than automation, like culture and mindset change of the organization, teams, and individuals, each of the steps in the journey needs to be supported by automation. An extremely important part of adopting DevOps is to understand what the journey for culture transformation will look like for the specific organization as this journey tends to be different for different organizations. It depends on the nature of business, how old an organization is, the health of the organization, what the team structures are, and what the mix is with respect to individual skill sets as well as mindsets. Another factor that can contribute to this amalgam of things to consider for DevOps is the region or regions from which the organization operates.

Once the cultural aspects of transformation are identified, tools and technologies need to be considered. Here again, the organization needs to identify its priorities and determine its current state with respect to adopting new technologies. This will help in putting in some timelines and setting the pace for adopting practices that take the organization closer toward the final state of DevOps (which by the way is a myth, as there is no end state when we talk about achieving the DevOps).

CHAPTER

10

Conclusion: The New Era

For an organization, achieving an evolved state of efficiency in any of the initiatives or practices it wants to adopt involves planning, perseverance, focus, and a good amount of time. Akin to this, improving the efficiency of IT operations by means of adopting DevOps is a long process that starts with putting up a plan and needs to be followed up with sustained efforts from all levels within the organization. The plan that worked for us involved focus on five key areas as described in Figure 10-1.

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3_10

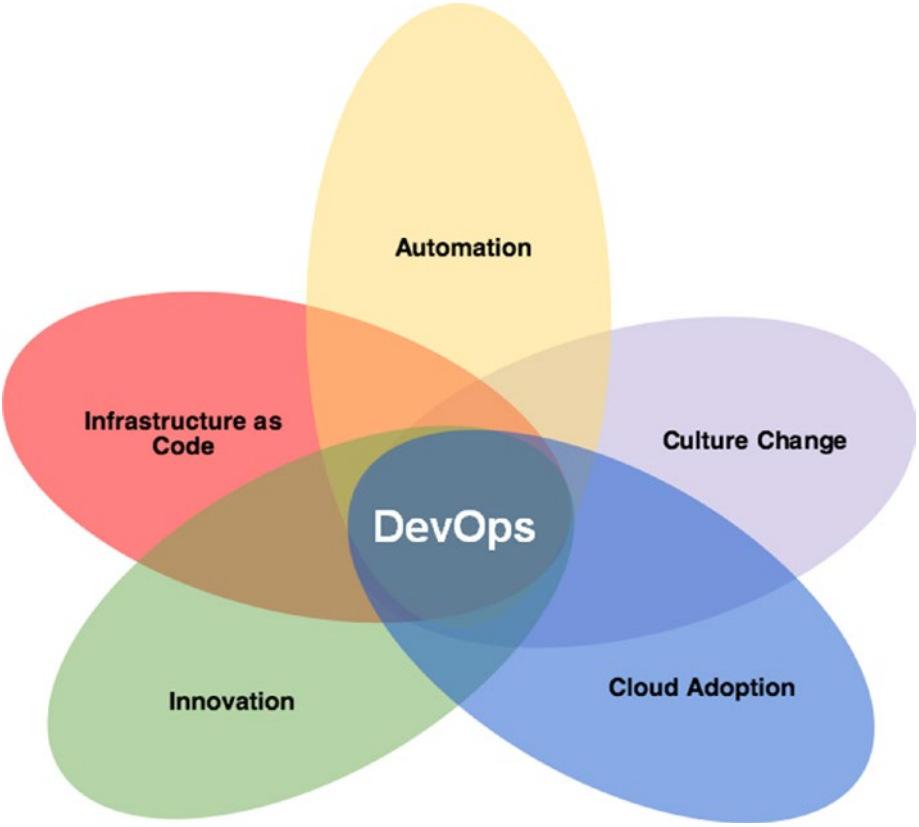


Figure 10-1. Key areas of focus to enable DevOps

The order in which these areas is adopted is not of much significance, as each area in itself needs continuous focus for a long period of time and will at some point be implemented in parallel to at least one or more of the other remaining areas. Also, each of the areas has an overlap in some way with almost all others and the integration of all of these creates a perfect ecosystem for driving DevOps.

In one of the earlier chapters, I talked about how the framework of DevOps enables as well as promotes iterative business process improvements. Well, in my experience the iterative approach applies equally well at the different stages during the adoption of the DevOps framework itself.

Back to the Beginning

While we were successful in making a lot of progress in terms of embedding the DevOps culture and framework in the organization, it would have been naïve for us to think that we were anywhere close to the finish line. All we had managed to do was to build a strategy for bringing in DevOps practices, to lay down a good foundation for enabling DevOps by identifying and implementing the tools and technology, and to create some reference patterns for the teams to follow when they are executing their work. We realized that the burst and hops model applies to different areas of work, and we had now perhaps reached the end of the first burst with regard to adoption of DevOps at our organization. A number of hazards had been identified along the way, and we would need to address them one at a time in the multiple hops that the teams would now be working on.

The release automation piece is a good example to illustrate this. We had developed a framework for executing CI/CD. This process would address execution of the individual tasks like checking the code into a repository, triggering automated as well as manual code reviews, followed by testing at different stages, initiating security scans and conditional deployment to the various environments. Along with defining the process, we were also able to identify and provide the toolset and hosting platforms needed to enable this framework. But after all this, there was a realization that the toughest part had just begun. This toughest part was embedding the new process in the daily routine of the different teams that were stakeholders in the process. In all honesty, it was not easy for the CORE team itself to stick to these practices at all times; it took us quite an effort to make this happen, and we are still not fully there. I believe the first few hops for us in the future will be focused on how we can onboard the other teams to align completely with the framework we have put in place.

Reiterating the Basics

One of the ways to increase the adoption of the new solutions is to block all the workarounds. In certain cases, this can be achieved by blocking access to the various systems that often act as a gateway. By doing this, we can navigate the admins onto the path we would like them to adopt without making it appear that it is being forced onto them. Doing this is very tricky, though. An approach like this could be perceived by people as extremely restrictive, which takes us away from the core principle of DevOps, which is agility.

The trick lies is having people realize that this is the right thing to do. The only way to achieve success in this regard is to make the teams understand that the "DevOps" way is the easiest and most effective way to complete the task. They need to realize that in the long run these solutions will provide better integration and quality in the services being delivered, and that will help

the teams to work in a more efficient manner. In order to achieve this, we established partnerships with various teams for creating awareness around the new DevOps practices they were expected to adopt. We would socialize the framework that we had put in place by inviting the teams over to join us for tech sessions. In these sessions, we would talk about different process improvements and process changes that were being built to drive DevOps. We would also perform demos for the solutions that were already in place. This would help paint the overall picture for these teams and increase their understanding of DevOps. The closer these discussions were to the area of focus of these teams, the better they would grasp the essence of DevOps. It was very clear that DevOps was going to mean different things for different teams, and we would have to help the teams decode DevOps for their areas.

Moving from Point A to Point B

There were two areas where we felt we needed to put in extra effort to increase the effectiveness of DevOps and to speed up its adoption. These areas were

1. Understanding the current state of each of the teams, which would become the starting point for them
2. Putting in a plan to help the team achieve the state that they needed to evolve towards to operate in an advanced state of DevOps

Every team had to honestly assess and accept where it stood currently in terms of the overall efficiency of services and solutions it was churning out. The next step would be to determine a goal they should be working toward to drive maximum efficiency in the way it operated. This would help in determining what path to adopt and what the journey would be like. Every team would then have to take its own odyssey toward reaching the evolved state it desired to achieve.

By obtaining a good understanding of these two aspects for a team, we believed we would be able to identify the scope of work that was required to complete the journey from the current state to the desired state. As an initial step in this direction, we started engaging the leaders within the teams. This first step itself would at times be very challenging, as the leaders of these teams felt that all this talk about transformation was slowing them down from delivering on their core responsibilities. We would tackle this by tying the practices of DevOps to the job function of their teams and show them opportunities to improve the overall quality of their work.

During our discussions with these leaders and their teams, the following are the areas we would usually focus on the following:

- Driving efficiency in the teams by adopting automation
- Enhancing governance for the solutions by designing processes that had minimal chance of deviation
- Motivating the team members by having them realize there is opportunity for them to learn and innovate
- Having the leaders realize that being at the forefront of these initiatives in their teams gives them a chance to create individual and team recognition and also invaluable experience that will serve them well in the future

In most cases, these discussions proved to be quite fruitful and helped us get out the door in terms of creating a customized plan for the respective teams.

The most important thing for achieving success in the adoption of DevOps is to ensure that people believe it is going to help them.

If we were able to convince the teams that adopting DevOps was not going to slow them down but instead allow them to work in a more agile fashion and ensure that the services and solutions they were building would be much more secure and resilient, then the remainder of the journey would be a lot easier. It also helped the teams that when we engaged them in conversations around DevOps, we focused on their area of focus and did not speak a generic language. This helped eliminate ambiguity for them. They were in a much better position to determine how to apply the practices of DevOps in their day-to-day work. By understanding how their work will change, they were able to act independently in terms of getting together as a team and chalking down the path ahead. They were also able to overlay the transformation toward DevOps with their teams' respective roadmaps to make sure there is no friction being introduced and their deliverables are not being missed.

Creating the Right Training Plans

Another realization that dawned on us was that we needed to create the right skills within the teams to succeed in this journey. We engaged the education team in the organization, which was responsible for creating the learning platforms. We worked with the different teams to create a comprehensive training plan that would not only enable these teams to get trained in the required technical skills but also skill them in the required culture change within their teams.

A focused training plan is always helpful to speed the adoption of any new large programs like Agile or DevOps.

The training methodology we adopted was completely revamped. We did not want the training to be anything like the usual trainer-trainee–based training. We felt that kind of training could at times make individuals perceive it as simply another formality to be taken care of rather than as an opportunity to learn something. We started converting the delivery means of these trainings into workshops, which were more of a team activity. The teams would be given specific tasks as part of these trainings, for which they would need to collaborate and depend on one another's ability to make everyone successful. The trainings were less generic in nature and more focused on the type of work and operations that went on in our organization.

Identify Opportunities to Improve Efficiency

When we get to the implementation part, DevOps works very differently for different teams in an organization. To drive DevOps effectively, it is important that you apply the right flavor of DevOps within each team. Multiple factors need to be considered, such as the type of work the team is responsible for, the organization's priorities, and most importantly, the team dynamics.

By adopting all of these practices, the teams were in a much better position to understand DevOps and the change that the organization wanted to bring in. Once the teams understood how DevOps was going to change their world, they started focusing on identifying opportunities within their areas to implement the DevOps practices. Everything from infrastructure management to compliance practices to security implementation was fair game to the change. *There is no one brush to paint with in terms of applying the DevOps practices across teams.* The teams were now realizing this and getting comfortable with it.

Age-old legacy practices were challenged if they were reducing the team's agility or its ability to deliver. Practices like change management, deployment process, asset discovery, and many more were constantly being discussed. New ways to improve these practices were being designed. Shortlisted processes were dissected, details were being captured in flow diagrams, and touchpoints were identified. Owners for these touchpoints would be approached and brought into the discussions so that everyone understood what the big picture was and worked toward a common goal of improving efficiency by optimizing their individual areas.

Team members would come up with plans for their teams of different time durations. Some of these plans would be short term, ranging somewhere between two and four weeks. Most of these would be focused on bringing about a change in some of the individual tasks or steps within a larger process.

Other plans were relatively long term, like a quarter or two. Any plans beyond three quarters usually were broken down into smaller time periods, as we believed that driving a program for more than that period went against the practices of Agile, which we had now started relying on heavily. These long-term plans were more focused on revamping a complete processes and were targeted at bringing about a larger change in the way their team functioned. These plans almost always included details around collaboration between different teams, as this was critical to the success of these plans.

The goal of putting these plans in place was to identify the current state of each of these processes, which were slowing down the organization's ability to keep moving ahead at the required pace, to determine the ideal end state, and to chalk out what the journey would look like. Depending on the term of these plans, the end state would at times be targeted as bringing about small changes to certain processes or could be aimed at revamping a complete process itself. In some cases, it could also mean getting rid of a process completely, as it was acting as dead weight and was no longer needed or was redundant. Once we understood what the end state should look like, we would then apply the burst and hops approach to complete the journey from the current state to the end state.

Conflict for Credit

With many teams now motivated to start operating in a DevOps fashion and starting to drive it within their teams, the ownership for driving DevOps as a framework in the organization became fuzzy. The CORE team was less of a business process/function owner and more of an enablement team to make other teams more efficient. As other teams started understanding DevOps more, they built capabilities within their teams for automation and other pieces in the DevOps puzzle to get more control over its adoption. This was also at times driven with the aim of being able to showcase the team's achievements in making strides in the right direction. There were both advantages and disadvantages to this approach.

Those teams that involved the CORE team and the other teams who were stakeholders in the processes they were revamping at just the right time added great value to the overall adoption of DevOps. This was because the plans and roadmaps would be shared and the required adjustments were made to remove any conflicts or duplication of work. This meant the team driving the change was now acting as a turbo to the DevOps engine and helping to accelerate the process.

There were some teams who still worked behind closed doors, not willing to open up their plans or share the roadmaps initially. There was a feeling that either the other teams were not skilled enough to add any value to their

team's plans or that the credit received for delivering the change would get diluted. Helping teams that fell in this category was a little more challenging, as they would be moving in a particular direction that was at times not aligned with the plans and direction where others were heading. It would take a lot more effort to align their individual plans with the organization's plans. Not many teams fell in this category though, and more often than not, with time they would realize the benefits of sharing the roadmaps as early as possible and reach out to other teams for help at the earliest.

The trick for the management to make the transformation as effective as possible was to ensure that the right teams and the right individuals were getting the required feedback as well as credit for the work they were putting in. One step in the right direction on this cause would mean that the motivation level increased by multiple times, while any mistake in this would put us behind considerably, as a believer would now move into a doubter category. This was a game that would need to be balanced constantly with little to no chance for error.

Future Is Now

Even though we are occupied with implementing and streamlining the items identified within different teams for enabling DevOps, we are simultaneously creating a path for ourselves that will set us on course to achieve further value out of the current DevOps movement in the future. We are pursuing adoption of concepts like **DevSecOps**, **Immutable Infrastructure**, **Symptom- and Cause-Based Monitoring**, and **Service Registry Code**.

We believe that these are the next-generation concepts of DevOps and that they will help us in the following manner:

- Embed security during every stage of building a solution and providing services rather than leaving it as an item we will circle back to at a later point once the functionality is implemented or the environment has stabilized..
- Move away from the grind of the change deployment processes to drive agility in operations.
- Ensure operations teams are only pulled into issues that are genuine and need immediate attention by categorizing problems based on impact to end-user services. We intend to achieve this by implementing intelligent solutions and integrations that are able to correlate multiple events captured by the monitoring systems.

- Use business as a primary driver while building solutions, embed the principles of User Experience (UX) to better understand the customer requirements and always take a holistic approach by understanding the end to end process and not solve the problem from only a particular perspective. The best practices of **UX** along with **Design Thinking** concepts are the areas we are looking towards to guide us on this path.

Introducing DevSecOps

It's often said that the only thing that is constant is change. While we were blowing full steam toward adopting the DevOps preachings, a new pattern was emerging in the industry. A number of prominent organizations were getting hammered by attacks on their Cloud environments. There were cases where either certain human actions were causing the environments to crash or the lack of focus on security was leaving certain hidden doors open for hackers to gate-crash the Cloud party in these organizations. Although security forms an important piece in the overall preaching of DevOps, the lack of calling out security explicitly at times during the execution was at times not bringing enough focus on security, and often it would be left for "later."

Across the industry, DevOps was being viewed only as a process improvement for the delivery pipeline of products and improving the operational efficiency of the companies by introducing automation. Great strides in a short period of time were being made in those areas at the cost of large gaps left in the security space. This was creating opportunities for the hackers to exploit, and they were making merry by introducing all kinds of ransomware, Trojans, and other kinds of malware into the systems. In order to bring the focus on security, a new term started emerging, called DevSecOps. This brought attention back to security at a time when everyone was solely focused on speed. Organizations started assessing their current security profiles against industry standards like the NIST security frameworks, COBIT, or the ISO 27000. Embedding the practices defined by these standards started becoming table stakes for all organizations, and any new or existing process was evaluated against these standards to determine its existence in the organization.

We started pivoting our DevOps implementation in this direction as well and had to make the necessary adjustments to incorporate within our plans what was needed to make this new requirement successful. Although DevSecOps does not take away anything from what DevOps preaches, it adds an additional layer of security to be incorporated in the overall DevOps practices. This new layer though was not something that was easy to incorporate, as implementing security almost always slows down everything else. This is the new task at hand for our organization and I am looking forward to the challenges it brings with it. After all, if there isn't anything new coming in, then there is not much

to be excited about. While the other teams are catching up on their learning on DevOps, CORE with a few other frontliners are now trying the pave the way for bringing in DevSecOps. I assume that the cycle of burst and hops is going to play out all over again as expected and we are looking forward to the ride ahead. I am hoping at some point soon we will be able to break away from preachings of burst and hops too, and we would like to launch ourselves away in a tangent like we have always done and discover an even more evolved way of achieving big things in a small period of time.

Summary

In today's world, the role of a CIO team has evolved from a mere support function into a part of the organization that can act as the pacesetter for the rest of the business functions. The technology and frameworks available today have the ability to catapult organizations into achieving greatness in their areas. During the journey of authoring this book, my focus all along has been on giving a first-hand account of how an organization, no matter how big or small it is, can drive DevOps to achieve this greatness. Although there is no single or guaranteed way, there are certainly some must-dos that need to be executed effectively.

These are as follows:

- **Top-down approach**: Drive DevOps as a value stream from the top supported by the executive leadership team. Investment in this area is perhaps the biggest favor the leaders can do for the organization.
- **Starting small**: Create a small focused team whose members are willing to get their hands dirty and empower this team. It is important that this team starts off by solving small problems and thus is able to generate confidence in other teams to adopt their practices.
- **Automation**: At every step of implementing DevOps practices, automation is going to surface. Ensure that you have a strong automation framework in place that is able to handle all the various types of automation requirements that will surface during this journey. Zeroing in on the tools to help with automation can prove to be a big asset in one's drive to achieve success in embracing DevOps.

- **Culture of innovation**: This can prove to be the trickiest to implement of all. There is no one way to do this. What is needed for this to be successful is persistent support and motivation at different levels in the organization. Once there is innovation happening in an organization, there is no limit to what it can achieve. Of everything else I have mentioned in this section, innovation holds the biggest value for the money.
- **Empowering teams and embracing failure**: There is absolutely no way that everything that the teams try comes out successfully. Failure should not be considered as the end to what could have been but a step that takes you closer towards achieving success in the future. The biggest impediment to innovation can be the fear of failure. The teams need to be given the confidence that they are okay to try out things they feel will drive business value and it is alright if they fail in this process.
- **Filling the skill gap**: The phrase "horses for courses" cannot apply better anywhere else. It is extremely important that the right people are picked for the right job in the process. Upskilling the teams in terms of their shortcomings in soft skills or technical skills plays a very important role as well.
- **Adopting Cloud**: There is no two ways about this. Cloud has become the starting point of adopting DevOps. Most organizations of the future are going to work at least in a hybrid Cloud environment if not in a complete Cloud environment. The agility and feasibility of DevOps that Cloud provides can in no way be ignored.
- **Keeping one eye on the future**: Adopting DevOps is a like shooting a moving target. There is no one moment in this journey where you can feel settled with the requirements. There is always the need to constantly evaluate what has changed and what is new out there.

I

Index

S. M. Farooqui, *Enterprise DevOps Framework*,
https://doi.org/10.1007/978-1-4842-3612-3

D

E

F

G, H

I, J, K

L

M

N, O

P, Q

R

S

T, U

V, W, X, Y, Z