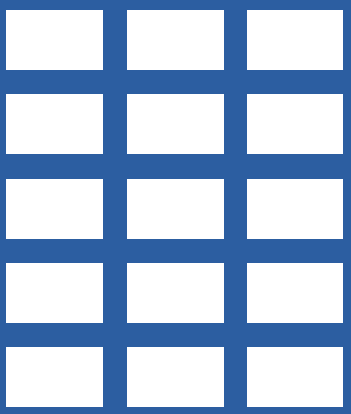
Full Stack Teams, Not Engineers



IT REVOLUTION
DEVOPS ENTERPRISE FORUM
2019

IT REVOLUTION

25 NW 23rd Pl
Suite 6314
Portland, OR 97210

Full Stack Teams, Not Engineers

Produced in the United States of America

Cover design and interior by Devon Smith

For further information about IT Revolution, these and other publications, special discounts for bulk book purchases, or for information on booking authors for an event, please visit our website at ITRevolution.com.

XebiaLabs
Enterprise DevOps

The 2019 DevOps Enterprise Forum was sponsored by XebiaLabs.

Preface

In May of this year, the fifth annual DevOps Enterprise Forum was held in Portland, Oregon. As always, industry leaders and experts came together to discuss the issues at the forefront of the DevOps Enterprise community and to put together guidance to help us overcome and move through those obstacles.

This year, the group took a deeper dive into issues we had just begun to unpack in previous years, providing step-by-step guidance on how to implement a move from project to product and how to make DevOps work in large-scale, cyber-physical systems, and even a more detailed look at conducting Dojos in any organization. We also approached cultural and process changes like breaking through old change-management processes and debunking the myth of the full-stack engineer. And of course, we dived into the continuing question around security in automated pipelines.

As always, this year's topics strive to address the issues, concerns, and obstacles that are the most relevant to modern IT organizations across all industries. Afterall, every organization is a digital organization.

This year's Forum papers (along with our archive of papers from years past) are an essential asset for any organization's library, fostering the continual learning that is essential to the success of a DevOps transformation and winning in the marketplace.

A special thanks goes to Jeff Gallimore, our co-host and partner and co-founder at Excella, for helping create a structure for the two days and the weeks that followed to help everyone stay focused and productive. Additional thanks goes to this year's Forum sponsor, XebiaLabs. And most importantly a huge thank you to this year's Forum participants, who contribute their valuable time and expertise and always go above and beyond to put together these resources for the entire community to share and learn from.

Please read, share, and learn, and you will help guide yourself and your organization to success!

—Gene Kim
June 2019
Portland, Oregon

Collaborators

Jason Cox
Director, Platform & SRE, Disney, @jasonacox

Christian Posta
Field CTO, Solo.io, @christianposta

Cornelia Davis
VP of Technology, Pivotal, @cdavisafc

Dominica DeGrandis
Director, Digital Transformation, Tasktop, @dominicad

Jim Stoneham
SVP Growth, New Relic, @jimstoneham

Thomas A. Limoncelli
SRE Manager, Stack Overflow, Inc., @yesthattom

Introduction

Randy is a full-stack engineer. He knows the company's software better than anyone else. He has experience with all aspects of the company's production services: literally "from chips to CSS." He writes server code and front-end code, makes user-interface decisions, administers the database, configures the CI/CD system, and handles outages. For all this he is paid handsomely.

There are two things that Randy doesn't know. First, he's about to burn out. Nobody can sustain his level of stress for long. His health is starting to suffer, and he can't figure out why working more hours doesn't enable him to stay ahead of all the projects that need to be completed. He hasn't had a vacation in years.

The second thing Randy doesn't know is that he's becoming a bottleneck who is hurting the company. He has so much responsibility that nearly every project depends on him; therefore, nearly every project stalls while waiting for him. The teams' frustration is palpable.

Randy's manager knows this. The company has tried to hire an additional full-stack engineer for two years but have failed. Full-stack engineers are as rare as hen's teeth. In the rare cases where they do exist, it is an unhealthy situation for themselves and the company. Finding Randy was a lucky accident. Now, he's a single point of failure. The company is one bicycle accident or winning lottery ticket away from disaster.

So what is a company to do?

In this paper, we'll discuss the myth of the full-stack developer and describe a more sustainable paradigm called the *full-stack team*, also known as a balanced team or a cross-functional team.

A full-stack team is a balanced team. It has many people, each with a core set of skills plus different specialties. As a whole, the

team's knowledge covers the entire stack. Team members are always seeking to educate and empower their coworkers to help them improve their skills. This creates a division of labor and a healthy system of specialization, and also encourages learning and growth for the entire team.

First, let's understand what a full-stack developer is, as well as the history and mythology behind the role.

Part 1: The Origin of the Full-Stack Developer

The Rise of the Full-Stack Developer

So, where did the term "full-stack developer" or "full-stack engineer" originate? Organizations are increasingly turning to technology as a catalyst for new business opportunities as well as a means to realize efficiency and speed through automation. The need to go faster and stay competitive is at an all-time high. New features, products, and services need to ship fast, with greater agility and higher quality than ever before. With that goal in mind, some organizations are naturally looking to simplify their operations, avoid slow transactional loops, and streamline complex and difficult-to-operate traditional technologies. They are seeking ways to compress their delivery cycle, eliminate latency created by siloed teams, and concentrate knowledge and expertise into smaller and smaller groups to expedite action.

As a result, companies have begun hiring developers and engineers who possess a greater set of skills spanning multiple disciplines in order to reduce the coordination necessary between specialized teams and ultimately deliver products faster. If an organization could hire one person, like Randy in our example above, who is an expert in front-end developer technologies, back-end middleware and databases, networking and storage configurations, and security concerns, why would they want to hire multiple developers with limited skills in these areas?

Some organizations believe that hiring full-stack developers or engineers is a way to implement DevOps principles and practices in the organization. While the goals of DevOps practices and full-stack developers can be very similar, there are distinct differences. DevOps is about enabling different teams to communicate and work better together, not eliminating the need for cross-team communication by merging teams

or making one person responsible for everything. DevOps practices help organizations speed up development, innovation, and delivery of high-quality products and solutions. DevOps is not an organizational structure; rather, it defines a way to organize independent teams (cross-functional or "full-stack"), a culture (humane and outcome-based), a set of Lean principles (fast feedback, including feedback from production), and a set of practices (highly automated, continuous delivery).[1] It appears that these DevOps efforts to organize cross-functional independent teams have resulted in some organizations thinking that individuals on those teams, similar to Randy in the example above, must singularly embody expertise in every area as a full-stack developer.

As we'll see in this paper, the desire to move toward full-stack developers has a detrimental, cyclic effect on organizations' recruiting efforts as well as developers' abilities to advance their careers. Full-stack developers are routinely heralded as "unicorns" or "rock stars" and are paid commensurate with that illusion. This may cause technologists to have unrealistic expectations for their own abilities and can dampen the drive for others who don't confidently believe they meet the requirements of a full-stack developer.

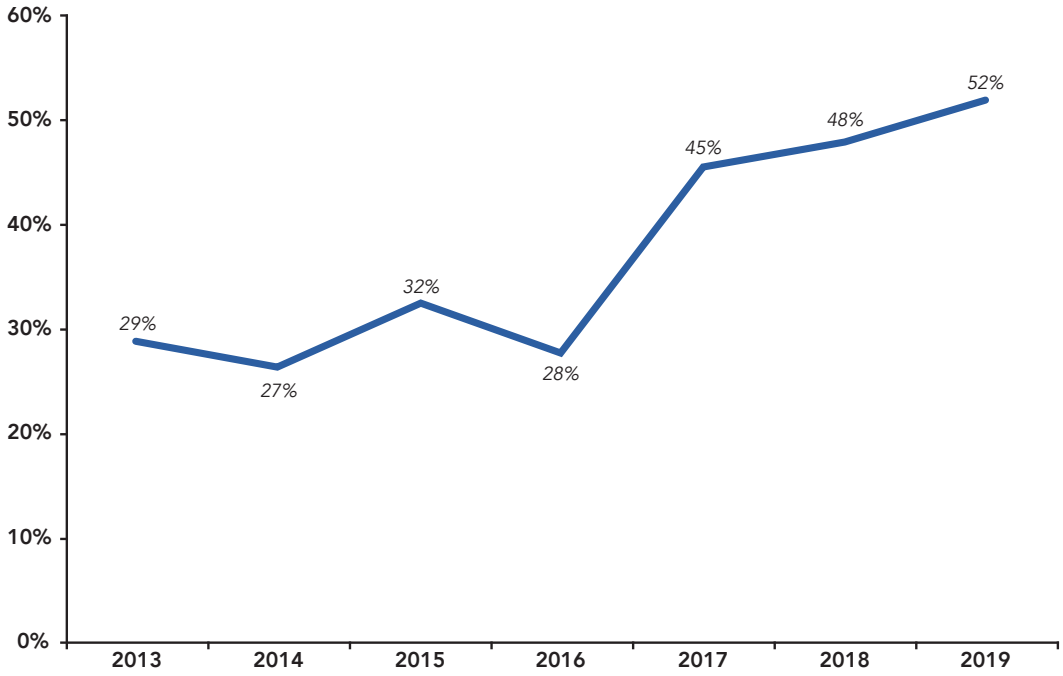


Figure 1: Full-Stack Developer over Time

Stack Overflow annually surveys the developers who use their site, and then they publish the results. Based on data from 2013 to 2019, there was a rise in developers identifying themselves as "full-stack developers."[2]

What Is a Full-Stack Developer?

The full-stack developer, also referred to as a full-stack engineer, has a broad range of expertise. Those skills can differ based on the context, but the intent is that the engineer has a working expertise in every layer that is required to build, ship, and run the subject products. As illustrated in the example above with Randy, a full-stack developer at a company may be responsible for every layer of the technology stack that powers the business, product, or service. The developer would have understanding of not only the end-user experience, but the API services that support the product, the development pipeline, quality engineering, security architecture, and infrastructure required to build and run the application. As mentioned in Randy's story, the full-stack engineer writes the code, runs the code, manages the data, and is the designated lead for handling outages.

Many organizations are recruiting for full-stack developers. Below is a job description template from SkillGigs.com that shows the typical set of skills and responsibilities expected of full-stack developers:[3]

- Design overall architecture of the web application.
- Maintain quality and ensure responsiveness of applications.
- Collaborate with the rest of the engineering team to design and launch new features.
- Maintain code integrity and organization.
- Experience working with graphic designers and converting designs to visual elements.
- Understand and implement security and data protection.
- Be highly experienced with back-end programming languages (e.g., PHP, Python, Ruby, Java, .NET, JavaScript).
- Proficient experience using front-end frameworks (e.g., AngularJS, KnockoutJS, BackboneJS, ReactJS, DurandalJS).
- Development experience for both mobile and desktop.
- Understanding of server-side languages (e.g., Jade, EJS, Jinja).
- Experience with cloud message APIs and usage of push notifications.
- Knowledge of code versioning tools (e.g., Git, Mercurial or SVN).

- Make sure to mention any additional frameworks, libraries, or other technology relevant to your project/company.

Here's another template for a full-stack developer position from BetterTeam.com:[4]

- Responsibilities:
  - Develop front-end website architecture.
  - Design user interactions on web pages.
  - Develop back-end website applications.
  - Create servers and databases for functionality.
  - Ensure cross-platform optimization for mobile phones.
  - Ensure responsiveness of applications.
  - Work alongside graphic designers for web design features.
  - See a project through from conception to finished product.
  - Design and develop APIs.
  - Meet both technical and consumer needs.
  - Stay abreast of developments in web applications and programming languages.
- Requirements:
  - Degree in Computer Science.
  - Strong organizational and project management skills.
  - Proficiency with fundamental front-end languages such as HTML, CSS, and JavaScript.
  - Familiarity with JavaScript frameworks such as Angular JS, React, and Amber.
  - Proficiency with server-side languages such as Python, Ruby, Java, PHP, and .Net.
  - Familiarity with database technology such as MySQL, Oracle, and MongoDB.
  - Excellent verbal communication skills.
  - Good problem-solving skills.
  - Attention to detail.

What Is the Full Stack?

A full-stack developer is one who is an expert in every part of the technology stack that is involved in the implementation and operations of some piece of software. As you can see from the template job descriptions above, this often means code that forms both the user interface (often referred to as the front end), a set of services or APIs (often referred to as the back end), data storage systems (such as databases and/or lower-level storage devices), and integration tooling (such as message busses).

With the increasing use of microservice architectures, the full-stack developer also needs to be well versed in a myriad of tools that support their use, such as circuit breakers, service discovery and configuration services, and distributed tracing. Of course, they'll also need to understand a whole host of tools to support the software development processes, such as source code control systems and CI/CD systems, and be experts in the practices involved in building, releasing, and running applications in production.

But even these job description templates make several references to things that can be added or removed to make the description relevant to your project or company. The skills needed by the full-stack developer are highly dependent on what technology the product requires as well as what processes are followed in bringing the product from idea into production. What do technologist and companies mean when they say "full stack"?

Figure 2 suggests one definition of "full stack" and illustrates that the expertise needed for a full-stack developer position spans a broad spectrum. In the center is the architectural stack, starting with infrastructure at the base and adding on middleware and numerous layers of the software itself. To the left of this stack is another stack of tools and practices that the full-stack developer might need to have expertise in to implement the software delivery life cycle (SDLC). And to the right are the operational practices needed to keep the software running well in production.

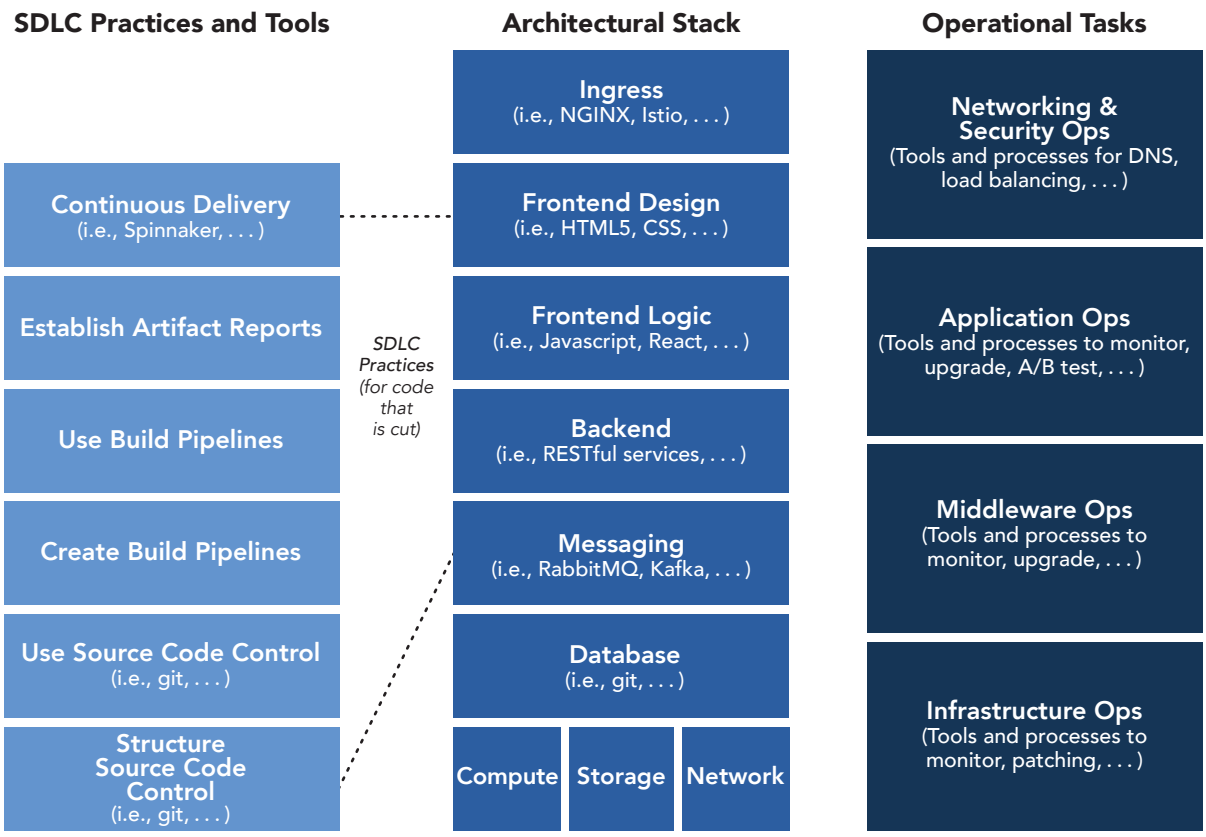


Figure 2: Example Full Stack Engineer Developing New Software

Is the full-stack developer really expected to do all these things? We will probe this question more fully in later sections of this paper; however, at the moment we want to turn back to this question: What is "the full stack?" Depending on the environment that the software is being developed and running in, some of the above elements may change.

For example, with the rising popularity of the cloud and cloud platforms, such as AWS, GCP, and Azure, some of the operational requirements may be handled by the platform, eliminating or reducing the level of expertise required by the full-stack developer. For instance, if Amazon RDS (Relational Database Service) were used for the database, the full-stack developer would not be responsible for upgrading their MySQL instances. But even so, some level of database expertise is likely still needed from the full-stack developer.

On the SDLC side, it is not unusual for organizations to have a group that provides CI and CD tools as a service to their developers. In this case, the full-stack developer

would not be responsible for making decisions around some of the specific CI and CD practices. However, they would still be responsible for following the established practices and they would still need to have some level of expertise in using those tools.

Figure 3 shows an alternative stack reflecting these types of scenarios.

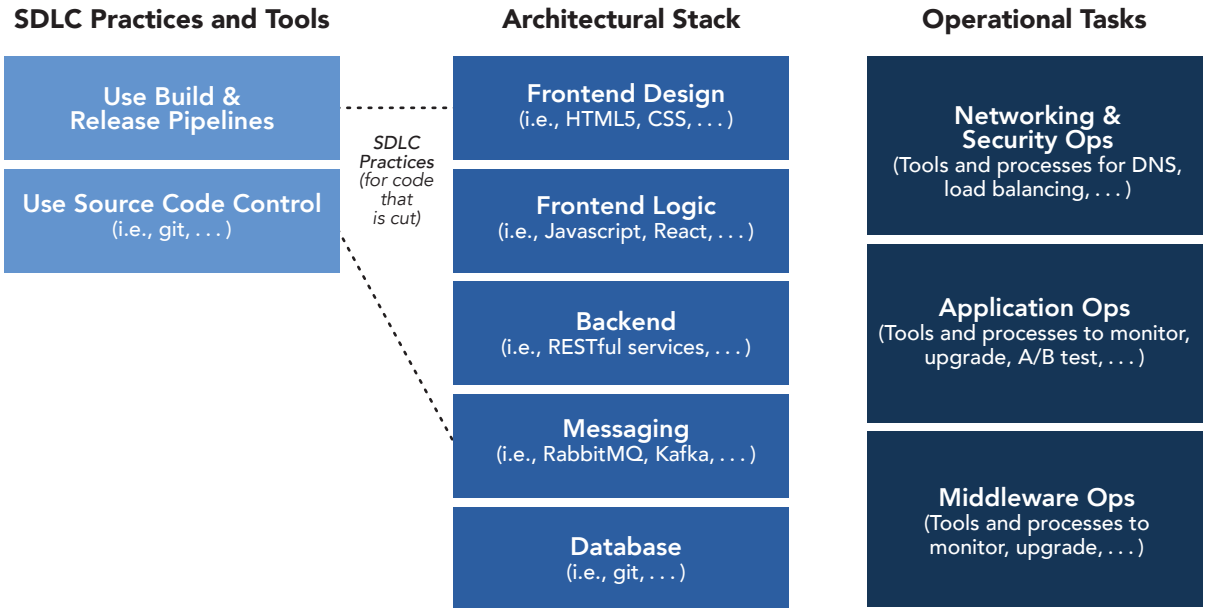


Figure 3: Example of Reduced Stack Developing New Software Where a Platform Team Provides CI and CD Tools as a Service

Lighter shading indicates reduced expertise needed.

Clearly, different definitions of "the stack" will result in different requirements on the full-stack developer. Further, these examples hint at the crux of the matter: Who does what? There are many things that go into developing and running software in production. How those responsibilities are distributed is at the core of our discussion. One of the main levers we have in forming efficient DevOps practices is how we define the stack.

Part 2: The Problems with the Full-Stack Developer

The Myth of the Full-Stack Developer

While the full-stack developer is on the rise, is it even realistic to find such engineers? Let's take a minute to walk through what it means to be a "full-stack" developer or engineer.

As suggested above, one important step is to define your "full stack." Are you talking about a range that goes from chips to CSS?* Or something smaller? Some teams include the network in their full stack, while other teams leverage platform providers and therefore don't require that expertise.

The job description template that we introduced earlier implicitly defines the stack through the list of technologies the developer must be proficient in with the expectation (and suggestion) that there will need to be extensions for the specific company platforms, frameworks, and unique requirements. Recruiting for these roles can be extremely difficult. Finding great talent is already a difficult proposition. By adding the full-stack requirements, the pool of candidates shrinks rapidly. We are observing this in the industry. As an example, one of the authors of this paper had full-stack roles posted for two years for their growth team and never filled any of those roles with a person with all the skills. They did eventually hire people who, over time, grew their skills to be able to work across the stack, but the full-stack criteria was problematic when used to screen candidates.

As we discussed above, the definition of the "stack" can vary greatly. Consequently, the definition of the full-stack developer role varies greatly, both by job posters as well as job seekers. This has resulted in confusing threads on job boards with companies and candidates trying to grapple with disconnected perceptions. In addition, bootcamps are forming to help individuals be credentialed as full-stack developers, but it's unclear if these bootcamps are helping close the gap.

Anecdotally, there seems to be a recruiting problem driven by a shortage of full-stack talent. However, can we find data to support that claim? For analysis, let's

---

* "Chips to CSS" refers to the layers of technology. At the lowest layer are the computer chips themselves. On top of that is the operating system, the software frameworks, the application itself, and the CSS code (part of HTML) which defines the look and feel of the application.

use the SkillGigs.com job description from the previous section and compare that against a dataset representing the job market. We will use the 2018 Stack Overflow Developer Survey,[5] which is a robust global survey of software developers, to see how the full-stack developer description matches the reality of talent in the real world.

In the survey, approximately 48% of the respondents identified themselves as full-stack developers. Based on the survey data, we should be able to narrow down the population (N) to those who actually qualify for the basic full-stack developer job description discussed previously. Table 1 illustrates the steps.

	Notes	Developers
Funding of milestones predefined at project scoping. New discretionary budget means the creation of a new project.		N = 98,850 developers
Filter by location: United States.	Narrower location, like city or state, was not available.	N = 20,300
Filter for candidates who describe themselves as full-stack engineers.		N = 10,400
Design overall architecture of the web application.	"Architecture" isn't directly measured in the survey, so we're skipping it for now.	N = 10,400
Maintain quality and ensure responsiveness of applications.	Again, this isn't measured directly so skipping for now.	N = 10,400
Collaborate with the rest of the engineering team to design and launch new features.	This is something you'll have to search for manually in a candidate's resume. Skipping for now.	N = 10,400

Maintain code integrity and organization.	"Use version control system regularly" was used as our proxy for this. (Using things like Git and Subversion.) It's not perfect, but it's a place to start.	N = 8,432
Experience working with graphic designers and converting designs to visual elements.	There is no available data about working with designers.	N = 8,432
Understanding and implementation of security and data protection.	There is no available data about security and data protection.	N = 8,432
Highly experienced with back-end programming languages (e.g., PHP, Python, Ruby, Java, .NET, JavaScript, etc.).	We note that a language like Go could (and some would argue should) be included, but a recruiter who is strictly sticking only to languages listed here would not include that. That is, even though there is an "etc." at the end of that list, a non-technical recruiter wouldn't know to include Go. We also filtered on any exposure to the languages versus "highly experienced."	N = 44

Table 1: Full Stack Job Postings vs. Available Engineers

This table shows the population of survey participates as filtered by an example full stack job posting requirements. The left column shows the requirement and the right column shows the remaining number in the population. Starting with 98,850 participates, the pool of candidates that meet the full stack requirement reduces to 44.

Let's stop here. There are still five more qualifications listed in the job description, including frameworks, development on mobile and desktop, additional languages, and cloud APIs, and we skipped five of the requirements above because we didn't have data. We only filtered based on four of the fourteen requirements listed (though we added a geographic requirement).

Forty-four. All of the companies in North America are competing for forty-four full-stack engineers represented by the 2018 Stack Overflow Developer Survey. That is just 0.04% of the surveyed population that matches the desired profile for a full-stack developer. Imagine recruiting against this profile and the tradeoffs necessary to actually fill roles! This helps us understand why organizations recruiting for full-stack developers have such a difficult time filling those roles and retaining that talent. The challenge becomes even more difficult if your goal is building a gender diverse team, since only 7.1% of respondents to this survey identify as non-male.

Organizations trying to improve their diversity and inclusion should take note: advertising for a full-stack developer may discourage women from applying. Studies show that men apply for a job if they feel they meet 60% of the job description.[6] Women, on the other hand, will not apply for a job unless they feel they meet all of the job requirements. Since the typical full-stack developer job description represents the pinnacle of engineering skills, it is very likely that organizations are unknowingly discouraging women from applying. If you are having difficulty attracting women to apply for your engineering organization, it could be that you are using terms or job descriptions that imply exaggerated skill levels. You may also want to eliminate exaggeration terms like "full stack," "rock star," or "ninja."

Why the Full Stack Engineer is Problematic

Previously, we have defined the role of the full-stack developer and looked at how difficult it is to hire for this role. In this section, we will look at the role itself and ask if the individuals in these roles are in a sustainable position or at risk of burnout. We don't let our servers get to 100% capacity utilization. Why do we allow our people?

Can you possibly be good at everything? In the digital era, the sheer quantity of tools, frameworks, programming languages, and methods/models overwhelms the brain. Technologies come and go, and learning a new technology often involves intense

effort. Learning a new environment and domain knowledge takes time. When asking engineers how long it takes for a new hire to get up to speed, a common response is six to twelve months.

It takes years of hard work to become a master of anything, and DevOps skills are no different. Kubernetes, for example, is famous for its complexity and relative ease for junior developers to botch up in unpredictable ways. If you are drinking through the firehose while learning new domain knowledge, you might also be deep into learning a new language like Clojure, utilizing new cloud APIs, applying the Big-O notation to new algorithms, and mastering machine learning and serverless functions. And that just scratches the surface of technology that continues to accelerate and expand. How do you intend to keep up with the growing stack? Mastering one skill by default means you will intentionally ignore other skills to focus on the one you want to master. The brain can only hold so much. Our human working memory is vulnerable to overload, which occurs as we study increasingly complex subjects and perform increasingly complex tasks.

Cognitive overload can result in lower quality, heroics, burnout and health problems.

One prime complaint from engineers is the consistent interruptions that prevent individuals from completing work during the day. If you can't do your most important work during business hours, when do you get your work done? The pressure to work during the wee hours of the night on top of one's regular day job is strong. Working long hours may come with trendy bragging rights, implying strength and power, but as *The Wall Street Journal* health writer Melinda Beck says, "Genuine 'short sleepers' only need four hours of sleep per night, but they comprise only 1% to 3% of the total population."[7] So for the 97% to 99% of us who aren't short sleepers, working the wee hours brings sleep deprivation and mistakes—both are contributors to burnout.

Burnout is more than feeling blue. It is a chronic state of being out of sync at work, and it can be a significant problem in your life, including the following impacts:

- **Loss of energy:** The burned-out individual feels overwhelmed, stressed, and exhausted.

- **Loss of enthusiasm:** Passion for the job has faded; the job "rubs you the wrong way;" it feels like a burden or a chore.
- **Loss of confidence:** Without energy and active involvement in one's work, it is hard to keep motivated.

Professor of Psychology Dr. Christina Maslach explains that burnout has many consequences for individuals, including physical illness, feelings of hopelessness, irritability, impatience, and poor relationships.[8] In severe cases, burnout can negatively impact executive functioning, attention, and memory. The Maslach Burnout Inventory (MBI) is considered the "gold standard" for measuring burnout. An internal study of four hundred Infosec engineers shared anonymously with us indicated that 47% are at significant risk of burnout primarily due to high levels of cynicism, a key dimension of the study because it captures the difficulty in dealing with other people and activities in the work world.

Risk of burnout from information overload is real and leads to serious health issues:

> **Too much information puts our brain's health in danger, resulting in an information overload. Over time, exposure to multiple sources of data leads to the overstimulation of the brain. Neurons get overloaded with data, numbers, deadlines, targets to be met, and projects to be completed, and all this unnecessary information can ultimately destroy them. Consequently, a stressed and overloaded brain is at high risk of dementia and other neurodegenerative disorders (Parkinson's and Alzheimer's diseases).[9]**

Expecting individuals, including ourselves, to be skilled in the "full stack" adds to the ongoing cognitive load that must be carried to get the job done. This can lead to burnout and even, in some cases, more serious health issues. But we need to be able to support the entire stack to help our businesses stay relevant and keep moving forward. What can we do? In the next section, we will look at some options to help support the entire stack while providing a healthy and humane work environment for our teams and ourselves.

Part 3: What's a Real Path Forward?

As discussed above, the full-stack developer or engineer is extremely difficult to find, develop, and sustain. In cases where individuals are able to fulfill these roles, they place themselves at risk of cognitive overload and the related health risks. Additionally, the acceleration and evolution of the technology stack means that mastery is a moving target, demanding continual learning investment in multiple domains. However, our businesses need full-stack functionality to stay relevant, efficient, and nimble in the market. How do we meet that demand without overloading individuals?

The Full-Stack Team

A full-stack team has the combined skills across its members to effectively design, build, deploy, and operate software throughout all development cycles of their deliverables. Moving to full-stack teams helps us deliver the full-stack advantage to our organizations without the challenges of recruiting, developing, and sustaining full-stack developers or engineers. Here are some examples of full-stack teams:

- **Rotational Engineer:** This team delivers an internal customer health dashboard that is used by all customer-facing teams. It has two back-end engineers, a front-end engineer, a data scientist, a data analyst, a designer, and a product manager. Each team member is essential, but rotation between roles and teaching within the team is strongly encouraged. Each specialist role also is active in communities of practice across teams to share and build deeper expertise.
- **Matrix Example:** A full-stack team can be assembled by embedding functional experts into product teams. The team members have two reporting relationships, the primary one being their product team leader and their secondary one being the functional area manager in charge of ensuring functional skills and career development. These embedded teams are responsible for learning from the rest of the team, taking ownership of the overall delivery and operational reliability of the product, but also ensuring the rest of their team has a grasp of their functional area as well.

Here are some recommended practices to help build full-stack teams:

- **T-Shaped Team Members:** Everyone's an expert in one or a few skills (the vertical bar of the T) but there's a basic understanding and empathy for many or all other skills needed to build and run the product (the horizontal bar). Consider rotating team members through different specializations to help create more T-shaped team members. Rotate/reshuffle team members and/or teach others on the team to ensure that experts can take a vacation and reduce risk in the event that someone leaves the company.
- **Everyone Teaches:** By teaching, you better understand the subject yourself while also building empathy for others. One VP of Engineering we talked to said the measure of a Senior Engineer is how much they teach, not how much they code.
- **Growth Mindset:** Hire for growth mindsets as you staff your teams, so those members are open to growth. It's okay to be curious and take on additional roles to move forward using rotation.[10]
- **Communities of Practice:** Build communities of practices so your team members can develop deeper skills and learn best practices from others.
- **No New Silos:** Don't create new functional silos required to ship value. The intent is to create high-performing teams of specialists with high degrees of communication, empathy, and knowledge-sharing. Creating silos can severely inhibit speed, communication, empathy, and knowledge sharing.

Partition "The Stack"

In an earlier, section we suggested that one of the challenges with the term full-stack developer was that it was not clear what constituted the full stack, and therefore it was difficult to even know the precise set of skills that were needed from the full-stack developer. Of course, in the meantime we have first debunked the myth of the full-stack developer and suggested a solution—the full-stack team (FST). But even with the full-stack team, the question remains: What exactly is the collective team responsible for? Is it, for example, everything shown in Figure 2, everything shown in Figure 3, or something else altogether?

What we introduce here is the idea that through careful partitioning of the set of all things required to develop and run some digital offering, the overall development and operations thereof can be optimized. And we already see ample evidence that this works from the last decade in IT, in particular with the astounding uptake of services offered from the likes of Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure, and others. Figures 4a, 4b, and 4c illustrate the cloud platform evolution seen over the last several decades. When a new application was to be brought online, the developers' organization was responsible for everything from racking and stacking hardware all the way up through the application code (Figure 4a). Then, with the advent of services like AWS, GCP, and Azure, the organization could obtain virtualized computing, storage, and network from one of those infrastructure as a service (IaaS) providers (Figure 4b). Through this, the "full stack" effectively shrunk so that the team bringing the application online was responsible for and needed expertise in far less tech.

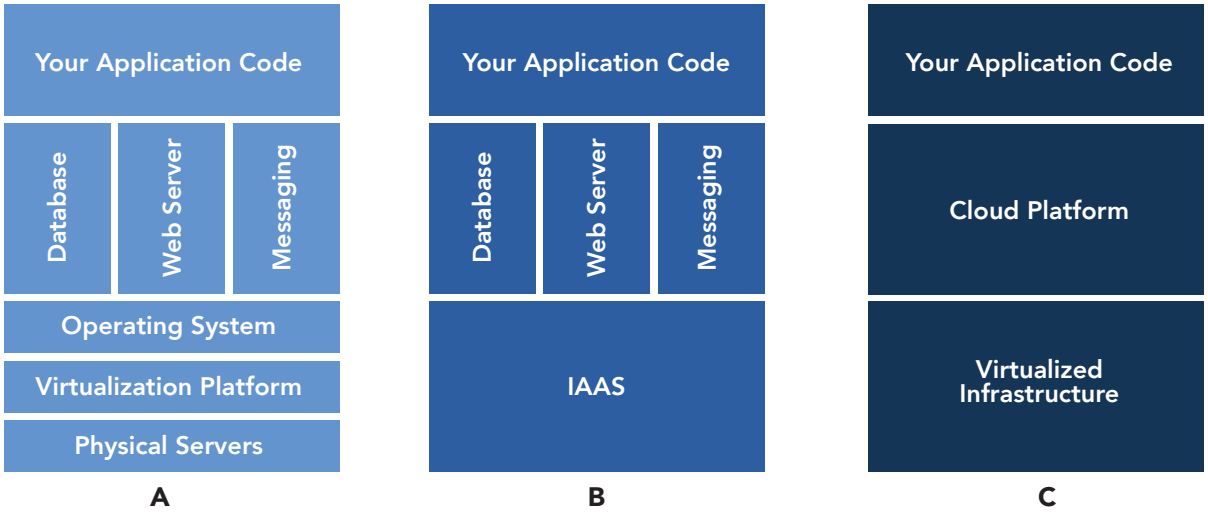


Figure 4: The Cloud Platform Evolution

In the last several years, even higher-level abstractions have emerged, called the "cloud platform." The cloud platform can take different forms, but it often includes many middleware and other services, such as workload scheduling and runtimes, databases, messaging services, routing, and more. These capabilities are offered, usually as a service, at higher level abstractions than IaaS, effectively further shrinking the set of things a full-stack team must have expertise in (Figure 4c). Some of the forms that these cloud platform services provide are:

- **Container services:** allows the full-stack team to simply provide container images such as docker files in which they have encapsulated their application implementation as well as runtime dependencies, and the platform takes care of provisioning all supporting compute, storage, and networking needs. Kubernetes-based examples of this type of platform are Google Kubernetes Engine (GKE), Amazon Elastic Container Service for Kubernetes (EKS), Azure Kubernetes Service (AKS), and Pivotal Container Service (PKS).
- **Application services:** allows the full-stack team to simply provide the application implementation while the platform provides the runtime for the application, routing services, and more. Examples of these platforms include Cloud Foundry, Heroku, Google App Engine, AWS Elastic Beanstalk, and Azure App Service.
- **Function services:** allows the full-stack team to simply provide implementations of low-level functions and a specification of how events tie those functions together to define the application behavior. Examples of these platforms include AWS Lambda, Google Cloud Functions, and Azure Functions.

These different abstractions effectively define the set of skills that the full-stack team is responsible for, and as Figure 5 suggests, may increase or decrease the size of that set. In short, careful selection of the platform used by the full-stack team to develop and bring their applications to the consumer play a central role in what the full-stack team is responsible for and must possess skills in.

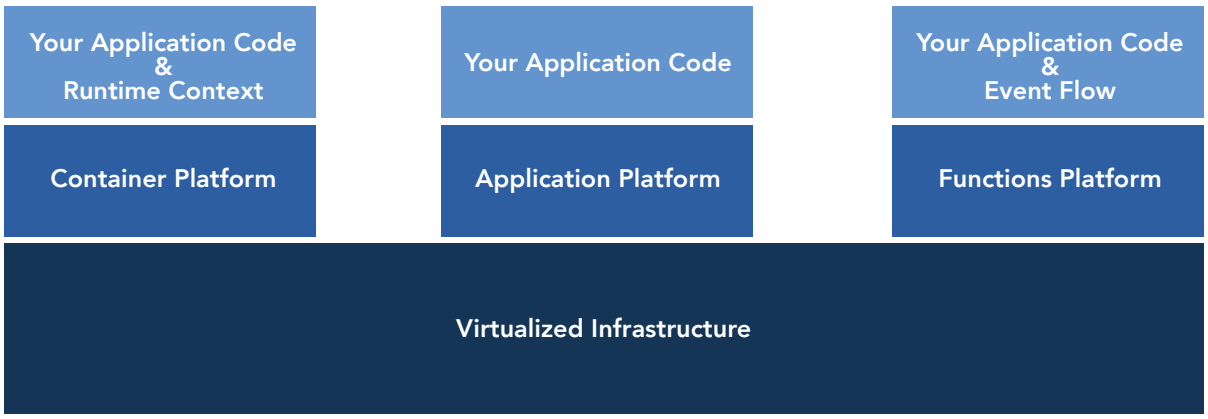


Figure 5: Example of how Abstraction of Infrastructure Reduces the Complexity of the Stack that the Team Must Own

There is, of course, a team that is responsible for providing the platform to the full-stack team. They provide this platform as a product, are themselves responsible for the development and operations of this software product, and as a team have a specific set of skills required of them; we call this team the *platform team* and the team that consumes the platform the *application team*.

Considering this, we offer one more interesting observation: what we have described here is a transfer of accountability for certain parts of the stack to different teams. This transfer of accountability is effective only when there is a well-established protocol for the application team to utilize the services offered in the platform by the platform team. We would warn that other attempts to transfer accountability, such as allowing developers to write code while making quality assurance (QA) responsible for its quality, are not advised. One of the most important elements of the full-stack team is that all members are incentivized to the same outcome: provide value to the consumer through their software.

Epilogue

It was five o'clock. Jennifer and her team were leaving for the day. Jennifer paused to reflect on what a good week it had been. A major new feature launched and everyone was in a celebratory mood.

How different, Jennifer thought to herself, *than a year ago*.

A year ago, the company lost a full-stack developer named Randy. Randy had burned out, became toxic, and ended up leaving the company. It was a struggle to replace him, but realizing that full-stack developers are next to impossible to find, Jennifer took a different path. She did not want to recreate the situation they had with Randy, where each launch's success or failure depended so heavily on him.

Instead, she decided to build a full-stack team. Their hiring strategy changed to focus on hiring smart people that liked to share and learn. Slowly and methodically they built a team of T-shaped engineers; each was a shallow expert in everything and a deep expert in one or more focus areas. Each member took responsibility to educate other team members and help them "level up."

The result is a team that is cross-functional and always sharing and learning. People do "fire drills" and "game days" to gain experience and cross-train. They work

collaboratively. They swarm on problems. No one person is a bottleneck. Projects get done faster, on time, and stakeholders are impressed with the results.

While business results are important, Jennifer is most happy about the work-life balance she enjoys. She works no more than forty hours each week, looks forward to work each day, and takes vacations without guilt. She loves working here and can't imagine leaving.

Most importantly, her team has become the exemplar. Engineers on other teams want to join her team; product areas want to work with her team. Other teams look to Jennifer's team for advice. Success breeds imitation and Jennifer's team encourages this by hosting bi-weekly lunch-time talks.

As she closed her office door, she thought to herself, *Why would we work any other way?*

FAQs

How do I get T-shaped individuals?

A T-shaped individual is an engineer who has at least a basic understanding of all the functional areas (development, systems, quality, product, systems, security, etc.) and also has deep expertise in one of those areas, which they can contribute to the overall team's full-stack capability. The way this is developed can take many forms. In all cases, a "growth mindset" versus a "fixed mindset" is essential to building the cross-functional horizontal bar. Organizations can help develop these areas by encouraging continuous learning, allowing individuals to move between teams and roles, and promoting a collaborative open environment. Building communities of technology or guilds of expertise can be an organizational design to help foster this growth.[11]

Carol Dweck writes,

> **When entire companies embrace a growth mindset, their employees report feeling far more empowered and committed; they also receive far greater organizational support for collaboration and innovation. In contrast, people at primarily fixed-mindset companies report more of only one thing: cheating and deception among employees, presumably to gain an advantage in the talent race.[12]**

I'm in a "full-stack engineer" role. What do I do?

There is nothing wrong with having expertise in multiple areas. However, there are dangers to watch out for as we have identified above:

- **You are a SPOF (single point of failure):** In some cases, the full-stack engineer is the only engineer supporting the platform. This puts the team and organization at risk of having a single point of failure. More importantly, it drastically impacts your ability to enjoy time off or time to invest in yourself. On a career level, this often results in a dead end—you are too critical in your role to promote.

- **You have a hero mentality:** Be strategic about how you position yourself. It is tempting to want to be a hero, but as mentioned above, that comes with human consequences to health, life balance, and longevity.
- **The quality of your work decreases:** We see continuous improvement in all functional areas. The ability to concurrently stay relevant at an expertise level in all of these areas can be onerous. Instead, focusing on an area to learn, perfect, and deliver can add a higher degree of quality to the overall team. It also presents a powerful opportunity to share that digested learning with the rest of the team to augment their horizontal aptitude.

How can you get out of this situation?

- Focus on teaching. Don't fix problems; instead, teach other people to fix the problem. It will take longer to fix problems, but be comfortable with that. This is an investment in the long-term health of you and the company.
- Develop a work style where you document as you go. Don't document something after the project is over; you'll be working on another project by then. Don't try to document everything. Assume the audience is someone you trust to take over this function, but who can call you if they have questions.
- When fixing a problem, encourage other people to "take the keyboard" and walk them through fixing problems instead of fixing them yourself.
- Enumerate the multiple jobs you do and work with management to delegate each job to someone else with you as the advisor. Every new hire should be someone that can take over part of your job so that you can specialize.
- Plan to take a three-month vacation. The planning will force your organization to develop coverage and train up other people in the company.

I have a small company/organization and cannot afford to hire a big team, so I must hire an engineer who can do it all.

Many of us have worked in startups and small organizations that require all members of the team to wear many hats. This is a perfect example where a single person must do it all and become an expert in the entire stack. However, as has been addressed above, there are still risks and limitations to continuing to operate in this model, including the creation of single points of failure, burnout, and eventual reduction in

quality. What happens when that single full-stack engineer needs time off for a life event, vacation, or for health reasons? How do you mitigate the burnout of working twenty-four seven? How do you ensure all the areas can scale at the quality that is needed for your business to thrive?

Practically and rightly so, established full-stack engineers, or engineers placed in roles where they must be an expert in the full stack, can demand significant compensation. For the same costs, teams and organizations may be able to hire multiple T-shaped engineers who have a growth mindset and are willing to work together to build a full-stack team.

References

"10 Symptoms of Information Overload and How It Affects Your Brain & Body," *Learning Mind*, accessed June 21, 2019, https://www.learning-mind.com/information-overload-symptoms/.

"The Science," *Mindset Works*, accessed June 21, 2019, https://www.mindsetworks.com/science/.

Beck, Melinda. "The Sleepless Elite," *The Wall Street Journal*, updated April 5, 2011, https://www.wsj.com/articles/SB10001424052748703712504576242701752957910.

Cox, Jason, Chivas Nambiar, and Paula Thrasher. *Continual Learning: Building a DevOps-Inspired Career Track*. Portland, OR: IT Revolution, 2017. https://itrevolution.com/book/continual-learning/.

Dweck, Carol. "What Having a 'Growth Mindset' Actually Means." *Harvard Business Review*, January 13, 2016. https://hbr.org/2016/01/what-having-a-growth-mindset-actually-means.

Evko, Tim. "There's no such thing as a full stack developer." *Dev*, February 12, 2018. https://dev.to/tevko/theres-no-such-thing-as-a-full-stack-developer-1m3i.

Fienen, Michael. "The Myth of the Full Stack Developer." *The Aquent Blog*, November 12, 2015. https://aquent.com/blog/full-stack-developer-myth.

"Full Stack Developer." SkillGigs.com, accessed June 21, 2019. https://www.skillgigs.com/full-stack-developer/?nabe=4589400386961408:0,5551131242266624:1.

"Full Stack Developer Job Description." Betterteam.com, accessed June 21, 2019. https://www.betterteam.com/full-stack-developer-job-description.

Gerasimov, Alexey. "Are Full Stack Teams an IT Utopia?" *The Doppler*, August 8, 2018. https://www.cloudtp.com/doppler/full-stack-teams/.

"Is it possible to become a full stack developer in 1 year?" Quora. Last modified June 18, 2018. https://www.quora.com/Is-it-possible-to-become-a-full-stackz-developer-in-1-year.

Katrompas, Alexander. "The hard truth about the full stack developer. Myths and lies." *Medium*, July 9, 2018. https://medium.com/@alexkatrompas/the-hard-truth-about-the-full-stack-developer-myths-and-lies-945ffadeeb8c.

Liu, Trista. "6 Essential Tips on How to Become a Full Stack Developer." *Hackernoon*, December 24, 2017. https://hackernoon.com/6-essential-tips-on-how-to-become-a-full-stack-developer-1d10965aaead.

Maslach, Christina, Jackson, Susan E., and Leiter, Michael P. *Maslach Burnout Inventory*. 3rd ed. Palo Alto, CA: Consulting Psychologist Press, 1996.

Maslach, Christina and Michael P. Leiter, "Understanding the Burnout Experience: Recent Research and its Implications for Psychiatry," *World Psychiatry* 15, no. 2 (2016): 103–111, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4911781/.

Schwartz, Mark, Jason Cox, Jonathon Snyder, Mark Rendell, Chivas Nambiar, Mustafa Kapadia, and Alyson Hoffman. *Thinking Environments: Evaluating Organizational Models for DevOps to Accelerate Business and Empower Workers*. Portland, OR: IT Revolution, 2016. https://itrevolution.com/book/thinking-environments/.

Shafer, Andrew. "There Is No Talent Shortage." SlideShare. Posted by Andrew Shafer, October 23, 2013. https://www.slideshare.net/littleidea/there-is-no-talent-shortage-velocity-2013.

Sharma, Gaurav. "Which Letter-Shaped Employee You Are?" LinkedIn.com. May 7, 2018. https://www.linkedin.com/pulse/which-letter-shaped-employee-you-gaurav-sharma/.

Shora, Andy. "The Myth of the Full-Stack Developer." Andyshora.com (blog), accessed June 21, 2019. https://www.andyshora.com/full-stack-developers.html.

Stack Overflow. *Stack Overflow Annual Developer Survey—2019*. Accessed June 21, 2019. https://insights.stackoverflow.com/survey.

Notes

1. Mark Schwartz, et al., *Thinking Environments*.
2. Stack Overflow, *Stack Overflow Annual Developer Survey—2019*.
3. "Full Stack Developer."
4. "Full Stack Developer Job Description."
5. Stack Overflow, *Stack Overflow Annual Developer Survey—2019*.
6. This stat is based on Hewlett Packard's internal study of their own employee records. Found in Tara Sophia Mohr's "Why Women Don't Apply for Jobs Unless They're 100% Qualified," *Harvard Business Review*, August 25, 2014, https://hbr.org/2014/08/why-women-dont-apply-for-jobs-unless-theyre-100-qualified.
7. Beck, "The Sleepless Elite."
8. Maslach and Leiter, "Understanding the Burnout Experience."
9. "10 Symptoms of Information Overload," Learning Mind.
10. "The Science," Mindset Works.
11. A good reference document or organization design to promote thinking environments is *Thinking Environments* by Mark Schwartz, et al.
12. Dweck, "What Having a 'Growth Mindset' Actually Means."

A Special Thank You to Our Sponsor

Our mission for the Forum is to bring together technology leaders across many industries and facilitate a dialogue that solves problems and overcomes obstacles in the DevOps movement. For three days at this private event, we gather 50 of the best thinkers and doers in the DevOps space to tackle the community's toughest challenges. We ask these thought leaders to collaborate and generate a piece of guidance with their best solutions to the challenges.

We would like to thank all of our attendees and our friends at XebiaLabs for helping to make this year's Forum a huge success.