Low Code Ops

And The Art Of Abstraction

Figure 1: Layers of abstraction in an ever-changing environment. Photo by PhotoHolic.

Written by

Isak Hassbring & Gustaf Lidfeldt

DD2482 Automated Testing and DevOps
KTH Royal Institute of Technology
Stockholm, Sweden
April 2021

Contents

1 Introduction **2**

2 Background **3**
2.1 Low-code and the art of abstraction 3
2.2 DevOps perspective 3

3 Practical examples **4**
3.1 Peltarion - a low-code platform perfect for ML Ops 4
3.2 Figma - a low-code tool that changed front-end development forever 5
3.3 There are many more 6

4 Reflection **7**
4.1 Defining the what and the why of Low code Ops 7

5 Conclusion **7**

1 Introduction

A lot is going on in the software engineering world. The recent years have shown tremendous improvements within the fields of development and operations - following the increasing popularity of *DevOps*. DevOps in general could be seen as a set of practices serving as abstraction layers that aim to let more people be more effective, by building interfaces and automation around difficulties - hence abstracting them away [3]. The same goes for many engineering fields, e.g. computer engineering is an abstraction on top of electrical engineering. Programming languages are abstractions (sometimes even *on* abstractions) on machine code [7]. Among programmers, a "one-liner" is often considered superior compared to multi-line programs (given equal task and performance). Why? Because humans love the art of simplicity and abstraction. It is as beautiful and satisfying as it is easy to understand [8] [14].

What is the next generation, i.e. the next layer of abstraction? It could be argued that the next generation of development practices, is perhaps currently emerging as the hugely disruptive *low-code/no-code* platforms ct. In the low-code/no-code space, the spreadsheet program Microsoft Excel could be seen as the birth of the practice around 1990, as it lets the average person compute calculations without any programming skills [21]. Now, a lot has happened since then and it in recent years low-code/no-code has emerged as an automation pool to count on. Today, platforms such as Peltarion.ai even go beyond the once-beloved one-liners and let creators (people building applications) set up, train, test, and deploy cutting-edge deep learning techniques with or without needing to write code. Bubble.io and lets you write auto-generated interactive front ends, also as a no-code practice. Microsoft is investing heavily in its low-code suite PowerApps, that provides automation of multiple business processes and user interfaces [1] [11].

As an example, we all know what the emergence of the no-code platform wordpress.org meant for web developers as it completely reshaped their industry. So, what does the broad no-code emergence mean for teams in both development and operations? Is this the beginning of the end of DevOps as we know it? We will explain why this matters for a DevOps engineer (or any engineer in general!) and discuss possible practical implications of low-code in the future.

2 Background

2.1 Low-code and the art of abstraction

Low-code and no-code development refer to the usage of a GUI (graphical user interface) for code generation instead of typing code by hand within an IDE (integrated development environment). Low-code development typically follows an agile methodology such as Rapid Application Development. Agile and DevOps are both designed to achieve faster development and deployment. [6]

While low-code platforms are not (and will perhaps never be) useful in all development scenarios, they are increasingly useful for a large and growing set of applications and practices. A UI/UX designer can for instance use *Figma* [5] to automatically generate fully functional React-JS components from their design, for prototyping and user testing. This hugely speeds up and automates the front-end application development. It lets the front-end developer take a step back and focus on API:s, integration, functionality, putting the pieces together instead of building the views. This is a more efficient use of resources and increases productivity.

Low-code platforms are especially useful in smaller, agile organizations or for smaller projects with few development resources. Low-code tools fit very well in such application development life-cycles as it lets programmers, as well as non-programmers, create custom-made apps or services for specific use-cases - faster and easier [19].

2.2 DevOps perspective

What serves as a good area to provide a low-code platform framework? Standardized development practices could be a perfect scenario. Why? Well, if something is standardized, it typically involves repetitive tasks. Hence you could build reusable, modular, object-oriented code around it. And if so - you'd be able to provide a GUI for it as well. And in the scenario of a standardized end-to-end independent task in development or operations, the entire task development could be made through a low-code GUI.

Now, what's interesting here is that standardized dev/ops practices are oftentimes *exactly* what DevOps provides development and operation teams with, and hence a subset of these practices can be provided with end-to-end GUI:s. This is in fact what several companies have realized, and a lot of big players are entering the field one way or another. *All* the DevOps cloud tools provide solutions that *do* fit into the low-code definition. Many tools still long way to go and researchers are looking into the field [20]. For us to sneak a peek into the future, we will look at the ones that are one step ahead - such as Peltarion and Figma, in the next section.

3 Practical examples

In this section, we will cover some concrete practical examples of disruptive low-code tools.

3.1 Peltarion - a low-code platform perfect for ML Ops

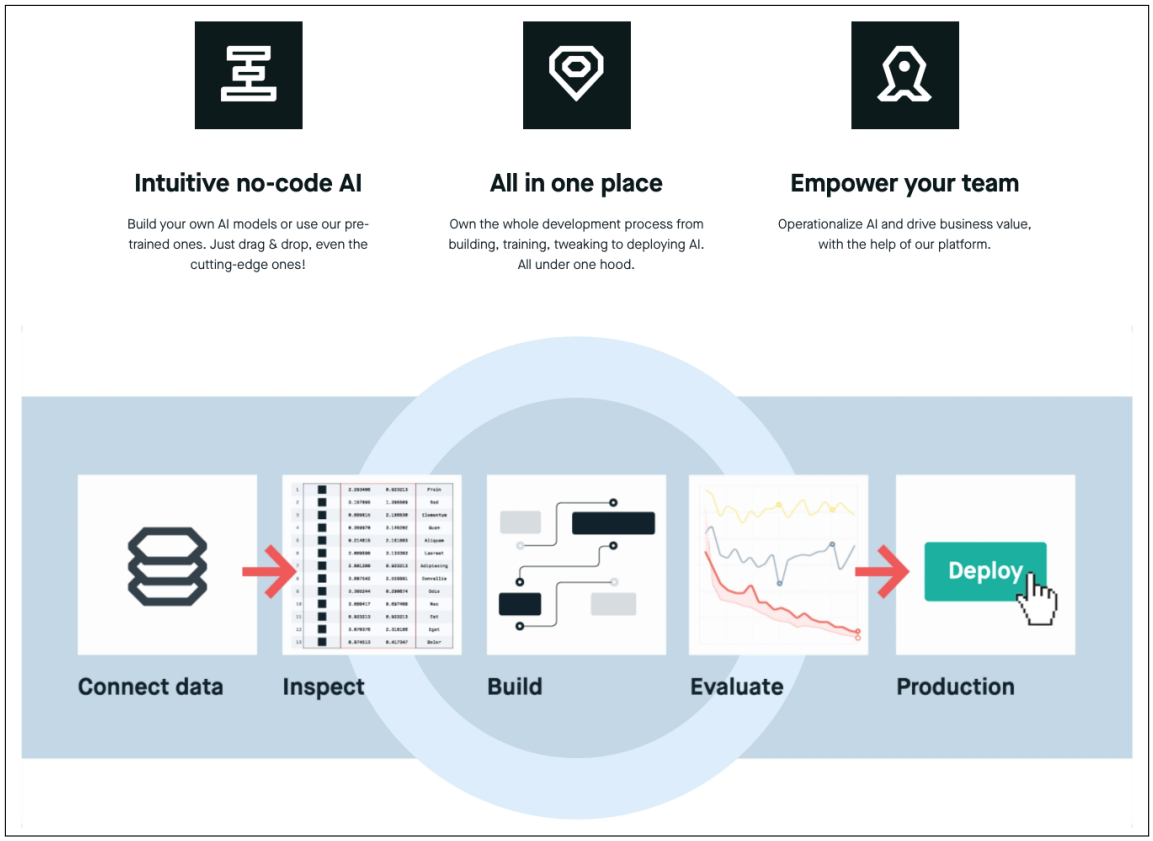

Figure 2: ML Ops enabled by Peltarion. Photo by Peltarion.

Peltarion [13] is a Wallenberg and EQT backed Swedish company that provides an end-to-end machine learning (ML) platform that makes the fuzz unfuzzy when it comes to ML usage and ML Ops. Deeply technical companies such as Tesla, Spotify, King, etc are using the Peltarion platform. Peltarion offers a no-code platform where you can use their cutting-edge pre-trained AI-models or build your own ones with a drag and drop interface. The entire development process from building and training to deployment is held in the cloud on Peltarion's platform. A tutorial is available here.

3.2 Figma - a low-code tool that changed front-end development forever

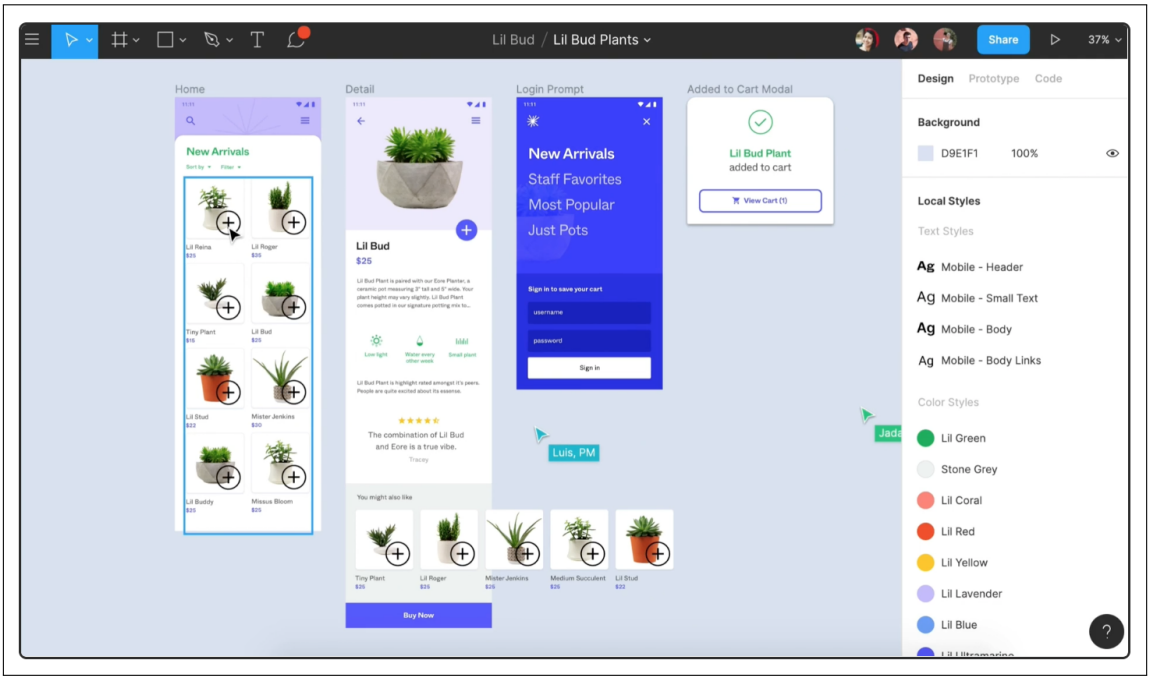

Figure 3: Demonstration of Figma's low-code platform for front-end. Photo by Figma.

Figma is a popular low-code tool used within front-end development and design teams to rapidly speed up development phases including GUI parts. Figma lets a designer or front-end engineer design an element or a component just as you would do it in a traditional design tool - but instead of generating an image, your result is fully reusable JavaScript components (e.g. React-components), which can be used directly in prototyping. The use of Figma speeds up the actual code generation and enables people who typically are not considered programmers, to participate in front-end development or code generation. As compared to traditional hand-coding, this is a big change [5].

3.3 There are many more

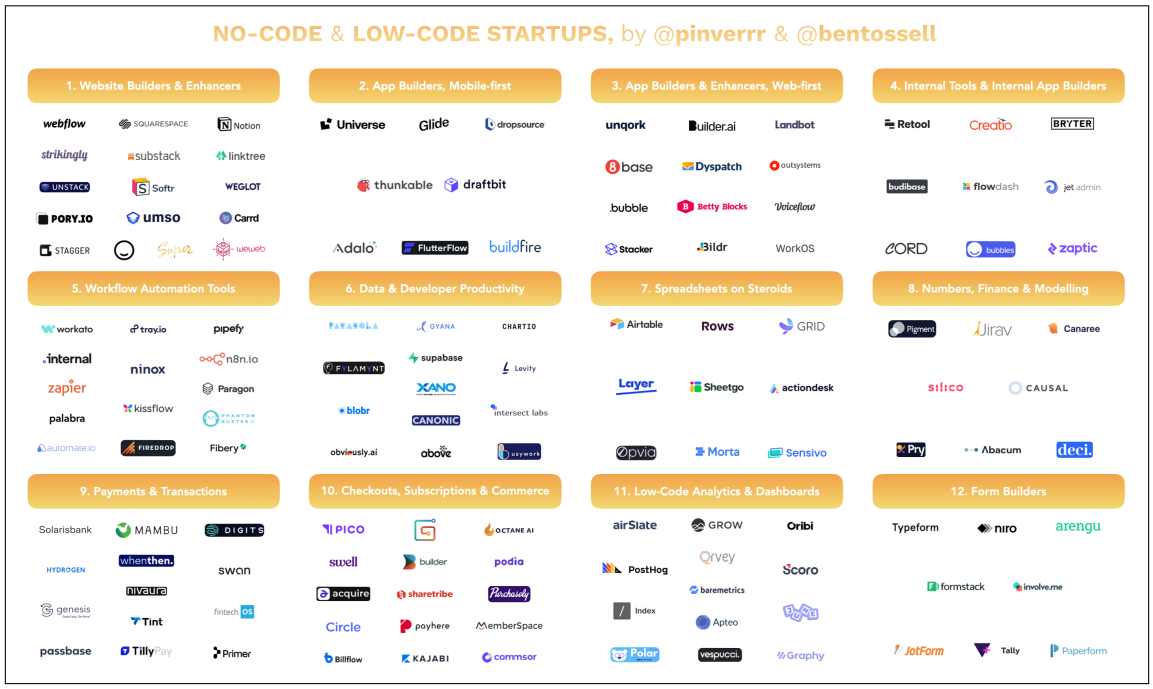

Figure 4: Examples of no-code and low-code platforms. Illustration by Pietro Invernizzi.

There are numerous low-code tools to check out, and we can't cover them all within this essay. For the curious reader, please check out the image above or some of our favorites: Substack, FlutterFlow, Workato and Chartio, or get to know the space better via e.g. Sahay et. al [17] or Invernizzi [9].

4 Reflection

4.1 Defining the what and the why of Low code Ops

We can define Low-code Ops as the practice of including *both* code-centered development tools and processes as well as low-code tools, in the development and operations life cycle. The difference between DevOps and Low-code Ops is that DevOps streamlines everything almost like an hour-glass, through the use of *code*. Simply put, everything DevOps is strongly related to code. What if there is no code involved in parts of the workflow? Suddenly it would be a struggle to put git be the core of the work processes. This is a struggle for DevOps today that must be addressed to keep up with the pace of low code, as more and more people will be able to participate in what is typically seen as development processes. Furthermore, *modeling* has been found lowering the learning curve of DevOps processes for for non-technical users enabled by low-code platforms. Colantoni et al. propose a conceptual framework for modeling and combining DevOps processes and platforms, as a future solution [4].

5 Conclusion

Is this the end of DevOps? No, it is not. Low-code is still dependent on traditional development and operations teams enabling the underlying infrastructure [22]. But, a growing set of what is typically considered DevOps practices will probably not require the DevOps engineer expertise in the future, due to low-code platforms providing abstraction. And for the DevOps practice itself, it *could be* a good idea to try to emerge from the dependency on services such as git being the core foundation of DevOps, and find ways to properly collaborate with low-code development practices. Nevertheless, the need for traditional coding languages and DevOps is still and will still be, critical.

References

[1] Bisson, S. 2020. *Microsoft's low-code tools: Now everyone can be a developer.* Link (accessed 30 April 2021).

[2] Bubble. 2021. *The Bubble Platform.* Link (accessed 30 April 2021).

[3] Cap-Gemini. 2019. *How to source your DevOps teams.* Link (accessed 30 April 2021).

[4] Colantoni et. al. 2020. *Towards Modeling DevOps Processes and Platforms.* Link (accessed 30 April 2021).

[5] Figma. 2021. *The Figma Platform.* Link (accessed 30 April 2021).

[6] Girard, LW. 2019. *DevOps vs Agile: Does it have to be one or the other?.* Link (accessed 30 April 2021).

[7] Heering, J., Mernik, M. 2002. *Minitrack Introduction: Annual Hawaii International Conference on System Sciences.* Link (accessed 30 April 2021).

[8] HP Inc. 2020. *How low-code/no-code platforms may reinvent DevOps.* Link (accessed 30 April 2021).

[9] Invernizzi, P. 2021. *Decoding the no-code / low-code startup universe and its players.* Link (accessed 30 April 2021).

[10] Jawale, C. 2020. *How low-code can fit into the DevOps culture.* Link (accessed 30 April 2021).

[11] Lindström, K. 2021. *Microsoft continues to invest in low code tools (Swedish).* Link (accessed 30 April 2021).

[12] Makadia, M. 2021. *Is Low-Code No-Code the Future of Software Development?.* Link (accessed 30 April 2021).

[13] Peltarion AB. 2021. *The Peltarion Platform.* Link (accessed 30 April 2021).

[14] Ruth, JP. 2019. *How Low-Code/No-Code Fit into a DevOps Strategy.* Link (accessed 30 April 2021).

[15] Rymer J., Richardson, C. 2015 *Low-Code Platforms Deliver Customer-Facing Apps Fast, But Will They Scale Up?.* Link (accessed 30 April 2021).

[16] Reselman, B. 2018. *Why the promise of low-code software platforms is deceiving.* Link (accessed 30 April 2021).

[17] Sahay et. al. 2020. *Supporting the understanding and comparison of low-code development platforms.* Link (accessed 30 April 2021).

[18] Skuid, Inc. 2021. *Skuid | Build awesome apps.* Link (accessed 30 April 2021).

[19] Steed, V. 2021. *Low Code and No Code: Developer Tools Expand Into Finance* Link (accessed 30 April 2021).

[20] Tisi et. al. 2019. *Training the Next Generation of Experts in Scalable Low-Code Engineering Platforms.* Link (accessed 30 April 2021).

[21] Ugur. 2020. *The First No-Code Platform: Spreadsheet Software.* Link (accessed 30 April 2021).

[22] Vikebø, Sydvold. 2019. *An Inquiry into Low-Code Solutions in Institutions for Higher Education.* Link (accessed 30 April 2021).