How to become futureproof as a SaaS company

18 October 2022, 14:47

Nick Boucart

As a Sirris expert, I regularly get questions about technical issues from SaaS entrepreneurs. A daily reality for a lot of start-ups and scale-ups, which is why the answers and solutions I offer can be useful for other companies as well. Below are some recent questions I have been asked and how I addressed them.

“I am not sure we are doing the right thing”

One of the founders of a promising SaaS scale-up says: 'I have noticed that we are not moving so fast lately, customers regularly find new bugs and honestly, I sometimes wonder if we are using the right technology for our platform'.

I get these kinds of questions quite often. Building a SaaS is not easy, and the tension between the business (sales, marketing, customers) and the technical team in charge of building everything, may be quite heated at times. The first ones want loads of functionality, here and now, while developers are often very deep in their code.

Two worlds colliding

When I get these questions, I usually suggest sitting down with the team for half a day. I will then ask open-ended questions about how the team works, what technology they use and most importantly, how they use it. How are requirements defined? How is planning done? Are unit tests or other (automated) tests developed? Etc.

I almost always manage to come up with 4-5 concrete recommendations after these meetings to bring the business and technical team closer together, to make business and development start trusting each other more. At the end of the day, in my opinion, the better the business and technical team understand each other, the better they know each other's sensitivities and metier, the better the chances of success.

"How can we get our software scaled up?"

"Our customers are running many more simulations than they used to, and our server is struggling," said the head of R&D at a fast-growing engineering firm, which has turned some of the know-how in their field into a SaaS product.

"Our engineers conduct research in our field, developing and refining our algorithms and calculation methods all the time. These Python scripts serve as the basis for our technical partner, who integrates them into a webapp/SaaS platform that we have been successfully marketing for some time. We are now finding that our current architecture and infrastructure are getting harder to run our simulations on time, and we are looking for ways to make this process more solid. Could the cloud help us there, and if so, how do we do that?"

It's certainly not the first time I've been asked that question. Technical teams do turn to the cloud more often as a potential solution to scale their SaaS platform more efficiently. But honestly, the barrier to start with the cloud is quite high. Just take a quick look at the AWS or Azure product pages, and you are inundated with buzzwords, lists of services, inspirational cases that a mere mortal can rarely identify with, and pricing pages that require higher studies in mathematics to understand. Getting your way around that and getting a feel for what a combination of the various compute and storage options might look like for your own use case often requires overcoming a serious learning curve.

What I usually do in those cases is use example architectures to outline various possibilities of how such a problem could be tackled, and then also consider the various implications of those choices. Together with the team, I go over the pros and cons of certain technical choices, and together we think about operational aspects such as monitoring, (auto-)scaling and security. During such a process, the technical team often turns out not to be very familiar with trends such as 'serverless' or containers, and therefore cannot assess what this might mean for them.

In the case above, we used a proof-of-concept to see how AWS Lambda, AWS S3 and AWS Step Functions could make the compute engine a lot more scalable and transparent. By working through a practical proof-of-concept, the team also gained insight into aspects such as IAM, infrastructure-as-code, AWS Cloudwatch ... these are all things that will come up sooner or later, but which do present a steeper learning curve for getting started with the cloud.

Organising for change

Whenever I talk to a development team at a start-up or scale-up, I always try to find what they do to organise themselves for change. Because if you have to name one recurring aspect when it comes to software, it's change: every contact with customers, prospects or users gives new insights and ideas for improving the product, new team members join and sometimes someone leaves, new cloud/open source building blocks potentially offer new opportunities. So, the question is not so much ‘what will change again tomorrow? ‘, but should rather be ‘how can we organise our software development so that we can best cope with tomorrow's changes?’

For instance, I always ask to explain the software architecture in broad terms. My rule of thumb for being 'change-proof' is: 'the simplest solution that could possibly work'. Not that it is necessarily wrong, but a team of three building microservices does not make things easier, especially when in practice, for almost every change, every microservice also has to be changed.

Another tool to better cope with change are automated tests. This primarily concerns unit tests, possibly some functional and/or end-to-end tests. I rate automated tests highly for several reasons:

- They provide a safety net for developers: solid unit test coverage gives developers quick feedback on whether what worked yesterday still works today.
- Automated tests are also a form of documentation: they show how code works and can be used, both now and in the future.

For me, automated testing helps both with keeping the existing code 'maintainable' and making the team 'change-proof'.

Recognisable? Always willing to exchange views!

Also check out our Cybersecurity Masterclass - SaaS, Digital Platforms, Software!

Authors

Nick Boucart