Cloud Native Development Part 1 (general information, IaaS, SaaS, AWS SDK)

Another approach to creating applications is to build them on another company's platform. This is not about deploying a ready-made application on the provider's hosting, but about building from scratch, using the partner company's specific technologies from the very beginning and adapting the application architecture in accordance with the recommendations. In this chapter, we use AWS as an example, but other companies also use similar mechanisms.

As an example, we will take the authentication and authorization subsystem and consider how it is implemented in a regular application and in an AWS application. For a regular application, there are many ways to identify a user and determine their authority. For example, checking their login and password against database records, and performing all the checks yourself. In the case of Cloud native Development, you should rely on the recommended approach. For AWS, you need to use IAM. The entity is attached, for example, to the EC2 instance where the application is deployed. Alternatively, you can use AWS STS AssumeRole, which issues short-term credentials. The advantage of this approach is that the application "grows" into the partner's infrastructure using its resources in the best possible way. The disadvantages stem from the same thing. Imagine if, over time, the terms of cooperation with the partner became unacceptable. For example, the price has increased, scaling difficulties have arisen, or it is impossible to use the infrastructure due to government restrictions. How can you extract the application from there and launch it on another infrastructure? In turbulent times, this risk cannot be underestimated. A sudden change in the terms of using the infrastructure can bury your business. In practice, a business rarely starts without any software. After all, a business solves a real-world problem: baking bread, transporting people, and manufacturing metal structures. Almost certainly, there is software for closing business processes. In my opinion, the most popular software is MS Excel and similar spreadsheets, and not any fancy programs. Therefore, in most cases, you will have to improve the existing software and not create a new one from scratch (although anything can happen in life). You will have to choose from the following options:

* Self-written software. It may have been in the company for decades. Large companies have dozens of software titles. Their integration with each other adds complexity.
* SaaS (Software-as-a-Service). In this case, you buy the right to work with software that is deployed and configured on the partner company's service. SaaS includes MS Office 365, Gmail, and AWS RDS. About the latter, let's say that AWS provides, say, MS SQL Server, where you can deploy databases, and you don't need to think about a license and hardware.
* IaaS (Infrastructure-as-a-Service). In this case, you get hardware (and an add-on above it) at your disposal. AWS EC2 is a virtual machine with an operating system installed. It does not have any additional programs. Their installation and maintenance are yours.

Each of the above options has its own advantages and disadvantages.

The option with self-written software requires significant capital investments. The development team will work on the project for a considerable amount of time, and the result is difficult to predict. In the future, the software will require ongoing support, including editing for changes in business processes, error correction, and other necessary updates. On the other hand, this approach provides flexibility and independence from vendors. No matter what happens to them, you can always move to another. In addition, historically, software was created within organizations for their own needs. So its share in companies is still significant.

The SaaS option allows you to start working today. There is no need to spend money on maintaining a team of developers. In contrast, the product has weak customization. Some vendors offer a separate logical layer, where the client company can make edits, adjusting it to its business processes. The capabilities are specific to a particular situation. Often, with an abundance of client companies, the SaaS owner company does not respond to the need to fix errors and the need for additional functions. Out of desperation, the client company tries to grow some software "on the side" to close the hole in functionality. This refers us to the option with your own software, described above. Of course, the SaaS vendor has complete control over its product. It can suddenly raise prices, remove functionality, or set other restrictions. Imagine that your company decided to expand into a new geographic region, and the vendor prohibits the use of vital software there. What will you do? The IaaS option gives more flexibility. In essence, you run your software on someone else's computer. Since Linux / Windows installed on you’re and someone else’s computers are approximately the same, if the vendor company starts to misbehave, you can always pack your things and leave. This scheme has found its place. For example, for two decades now, companies have been offering website hosting services, file storage, or entire virtual machines. This is where cloud computing began. And the zoo of different solutions today is an extension of this process.

To sum up the above, in most cases, you will be dealing with existing software, around which the usage practice has developed. In addition to the purely technical aspect, it is also essential to consider how people use the software. They create ready-made recipes, even bypassing known errors and shortcomings. When planning an evolution or even a revolution in software, you will have to take into account the established practice.

Transferring software to the cloud is not an end in itself. After all, we are not at a hackathon, and we do not need to impress colleagues. Company owners strive to reduce overhead costs and increase business reliability, rather than seeking recognition for mastering a technological innovation. On the contrary, implementing any innovation entails overhead costs and the risk of disrupting the established process. The more revolutionary your proposal, the more resistance you will encounter. In short, before adopting the cloud and moving all the company's software, consider the reasons behind this decision, the specialists required, how you will address potential issues, and who will cover the costs.

Most often, infrastructure updates occur gradually. First of all, those components are selected that are not closely related to others. For example, we have prepared a simple program "s3-read-file".

It simulates a scenario where a file storage was decided to be moved to the cloud, and an AWS S3 Bucket was used for this. To access the storage, the program uses the AWS SDK (Software Development Kit). This package is available for different programming languages, including JavaScript.

The program "s3-read-file" is essentially straightforward. It creates an instance of S3Client and passes it the availability zone, Access Key ID, and Access Key, which are specified in the S3Bucket properties. Then the program ensures reading of data from the cloud storage and, ultimately, receives the contents of the file.

Here you can see how all the complexity of working with the cloud is packed into the SDK. The software developer focuses on a specific task.