Macro problems with microservices – part II

19 October 2017 | **Software Engineering** | **Igor Spasic**
Reading time: 3 minutes

Microservice architecture is more and more part of modern software solutions. As we have seen in the first part of the blog series, the development of microservices requires additional attention and planning, that might not be anticipated from the beginning. We are continuing now with more challenges – this time regarding the infrastructure of microservice architecture.

Infrastructure

Logging is no more an activity that happens in the console. It is now a distributed task that often requires more infrastructure components. Depending on the load, you might end with having log emitter, buffer, datastore and logs visualizer. Each microservice may format logs differently due to different technologies and you want to collect and process it all. You also want to have the clear separation of logs per microservice and also per single instance (when microservice is scaled). Don't forget that amount of logging may be huge even with a small number of users, especially when debugging. I've seen file system throwing I/O exceptions on AWS just because it could not flush all the logs to the files (of course, the instances were provisioned for development purposes: small and not production-ready).

Start monitoring from day one. Scratch that, monitor from day zero! The only constant in microservice architecture is that each component may fail anytime. Coming up with good detection of the failures may be challenging; nevertheless, do it right away. You must have the feedback on how your set of distributed components are working. Add health checks for services: check not only if a component is up (i.e. listening on some port), but if it is also working as it should. Collect data across the infrastructure, add alerts. Monitor not only your software components but infrastructure components as well. Try to detect slow calls, IO spikes, network lags. Monitoring is your eyes of the system, open them. The worst thing that may happen is the situation when production stops working and you have been given a SSH to a node with literally thousand containers and the task to figure what's wrong. Yeah, goodbye my night sleep.

Automation is the king and it rules the microservice project. Due to a number of software and infrastructure components, it is impossible to live without any sort of automation. Be prepared to end up with a load of scripts. Note that different teams may need different automation or scripts – they simply use parts of the infrastructure differently and may want more fine-grained access. Furthermore: automation is more than just scripting. Having working CI/CD infrastructure is also part of the developing pipeline you must have in your

project; just now it has to work for all components.

Micro take-a-way

The above set of points is something that is considered as good software engineering practice. But the thing is: with monolith application, you don’t have to implement them all. You may introduce some later, or in a smaller scale, or even completely skip some. You may get away with that. With microservices, you can’t allow missing any of above points.

There is no easy way around. With microservices, you don’t need just great development – you will demand high engineering. And you and your team must be up to it!