RunMyProcess.

a Fujitsu company

On-demand Micro-Services – A Perspective

Posted on 26th November 2015

The concept of micro-services is one of the hottest topics in the IT industry today, promising to accelerate change and transform our ability to scale as we head towards a new hyper-connected world.But what are micro-services,how do they help and how do we maximize the benefits they bring while minimizing unnecessary technology distractions?

The concept of Micro-Services is one of the hottest topics in the IT industry today. Rather than creating heavy-weight & monolithic services, it suggests a more light-weight approach by splitting services into independently managed micro-services. As an example just think of an order process that can be divided into order placement, order confirmation, payment and shipment (simplistically). If you put all this into a single service you'd run into a couple of issues like complex maintenance and coarse scalability. You also lose agility because you always have to assemble the complete service even though you may just want to update the shipping component for instance. Vice versa a micro-service can be updated & scaled independently and the call contract between different micro-services is enforced via a clearly defined interface. As such micro-services also emphasize the sense of ownership because rather than being owned by the abstract being called the "software department" a micro-service is owned by the responsible (sub-) team.

These days micro-services are often piggybacking on container technologies like Docker. Docker is an evolution from old school VMs. You can run multiple resource isolated containers on one or more VMs (or bare metal machines) and container management makes it fairly easy to start/stop/migrate them – all at much higher speed and without caring too much about the specifics of the underlying infrastructure. However, while this approach has obvious advantages over VMs, containers still follow old thinking:

Who wants to manage a horde of long-running containers?

Who wants to manage the runtime engines within the container?

Nobody. These tasks are not the core competency of a business. A business wants to create value via software but it doesn't want to operate it.

Now if we go one step further, there are indeed technologies that can tackle some of these issues. One interesting service in this area is AWS Lambda, which allows you to upload your micro-service code ("Lambda functions") and execute it on demand. AWS then manages all the underlying containers and runtime engines and you are only billed according to the accumulated execution time and deployment size.

Sounds cool, right? But ... wait a second here. Is this so-called "serverless" approach really "news"? Fujitsu RunMyProcess has been around for years now. RunMyProcess allows you to define process flows. A step within such a flow can be associated with a script – using either Javascript or Freemarker – which is in essence nothing else but a -let's call it- nano service. Multiple process steps can then be orchestrated into a higher-level micro-service. Such a flow can also be exposed via a synchronous API, essentially creating a backend-as-a-service. Persisting data in a NoSQL database is also available.

RunMyProcess continues further, going beyond a simple code hosting approach. For example in Lambda you can create versions at a single function level only whereas in RunMyProcess you are able to handle the complete life-cycle management of an orchestrated service. Rather than just relying on bare-metal code, you also have a visual representation of your flow which can be immensely helpful when communicating with non-technical stakeholders. Furthermore you also gain development speed by taking advantage of the code-less aspects of the platform such as the support for both mobile- as well as web- front-ends, the integration with remote web-services (in the cloud and/or on premise) and the availability of process analytics.

Welcome to our world of micro-services!