Introduction to Part 2

Here comes your boss again. “Make sure you define a Service Level Objective and include metrics in your logging. By the way, we use Log4J for our logging. We also have standardized discovery and registration services. Don’t forget to hedge your cloud requests. Artillery is our tool of choice for load testing. Our deployment pipeline is triggered from your commits so you don’t need to worry about that but you do need to provide unit tests for the build server.” At least this time you understand the words if not necessarily all of the context. Part 1 helped you the last time you were confused, Part 2 will help you even more.

Part 2 deals with two fundamental topics – microservice architectures and the deployment pipeline. Microservice architectures have a symbiotic relationship with containers. That is, packing a microservice architecture in a container simplifies the deployment of the microservice and a microservices size suggests a limited use of resources which fits into the constraints on containers. Another relationship we will explore is that between microservices and team size.

The second portion of our discussion of microservices focuses on the technical aspects of microservices. We discuss microservices from the perspective of their quality attributes – performance, reusability, modifiability, and availability. The relationship between a microservice and its cloud platform plays a role in understanding these quality attributes.

Since a microservice communicates only via messages, the protocols used to pass message passing are important. We discuss REST and gRPC since those are very commonly used in the internet environment.

The final portion of our microservice discussion deals with the common services used by microservices. These are common to many microservices within an organization and are factored out into libraries or other services. Here is where you will read about discovery and registration.

The second topic in Part 2 discusses the deployment pipeline. Modern practices use continuous delivery if not continuous deployment. The pipeline is dependent on using different environments for different purposes and we identify environmental management as a first class concept in a deployment pipeline. TTL for DNS servers is one mechanism used in switching environments. The deployment pipeline is heavily dependent on tools and we discuss different classes of tools used in a pipeline.

We also discuss different strategies for deployment and partial deployments. Feature toggles are used to control the deployment activities and these rely on the distributed coordination mechanisms that we discussed in Part 1.