Innovation, not Novelty: Three models for Microservices Architecture Design

Beaton, Leif

NGINX, Inc Leif.Beaton@nginx.com

Abstract: Innovation, not Novelty: Three models for Microservices Architecture Design

Microservices are key to the digital transformation journey that many – if not most – organisations are going through today. The move from large, cumbersome monolithic application infrastructures to agile, complex microservices approaches can be challenging for many at a technical, let alone cultural, level and there are a large number of different approaches to help accelerate and ease this change.

One of the most significant factors when moving to microservices is network design and implementation. Communication between services happens over the network, rather than in-memory, as is the case in traditional monolithic designs. Consequently, in this talk, Leif Beaton, Senior Technical Solutions Architect, will outline the three models that constitute NGINX's reference architecture for microservices, each representing an increasing level of sophistication and power.

Leif will also refer to the circuit breaker pattern, which is well-known by many solutions architects; rather than trying to avoid failure from the get-go, it embodies the notion that failure will occur because of the multiplicity of moving parts within any application environment. The circuit breaker principle uses a device to monitor the overall 'health' of the application infrastructure and takes action to mitigate or prevent errors, rather than relying on users or systems staff to spot issues.

The three models that Leif will discuss within the reference architecture are not necessarily mutually exclusive and represent an evolving 'maturity model' of adopting microservices during a digital transformation journey. The first, the Proxy Model, uses NGINX software as an ingress controller, handling inbound requests to microservices, as well as dynamically handling DNS routing as new services are created. NGINX's service registry also handles interservice communication within the cluster in a similar fashion to how Docker Cloud operates.

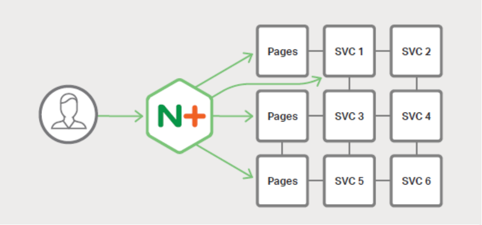

Fig. 1. The Proxy Model for Microservices handling

Secondly, the Router Mesh Model represents a mid-point of complexity and functionality. It uses two NGINX instances; the first to handle overall traffic routing (reverse proxy) and places another in the middle of each microservice cluster. This deployment manages the connections amongst the different microservices, balancing the load between them. This includes the use of the circuit breaker principle to monitor the health of instances, for example, throttling or splitting traffic as needed. This model allows SSL/TLS termination at the cluster, but not service, level.

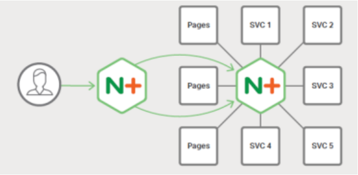

Fig. 2. The Router Mesh Model

Finally, the Fabric Model is one of the most sophisticated and complex models for microservices deployment, also representing some of the greatest scope for growth and flexibility across the application environment. It places NGINX Plus in each container that has a microservice instance within it, handling application traffic and becoming the forward and reverse proxy for traffic. As in the Router Mesh

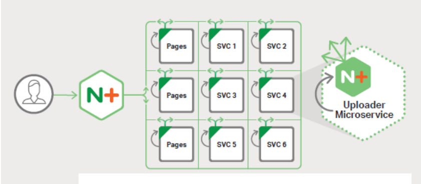

Fig. 3. The Fabric Model

model, NGINX handles the load balancing, service discovery and ‘health checking’, but also allows the deployment of SSL/TLS security at the microservice level without compromising on performance. Furthermore, these secure connections can be established between microservices in a stable and persistent fashion, allowing organisations like government bodies to create ‘mini VPNs’ either between services themselves or services and external points.

Whilst there are as many approaches to microservices as there are applications, Leif believes that these three represent a solid foundation for flexibility and growth within an organisation engaged in digital transformation. Architects and engineers within industry can often be guilty of ‘magpie syndrome’, making it crucial not to embrace technology for its own sake – for example, Istio was designed for a very specific purpose, but has almost been adopted as an industry standard. To combat this, Leif will also discuss examples of successful microservices implementations throughout the talk, from CapitalOne – which sees 12 billion API calls each day – to Canal+, which can now handle 50,000 requests per second, but has also managed to reduce its AWS footprint.