Serverless = Operations on Autopilot

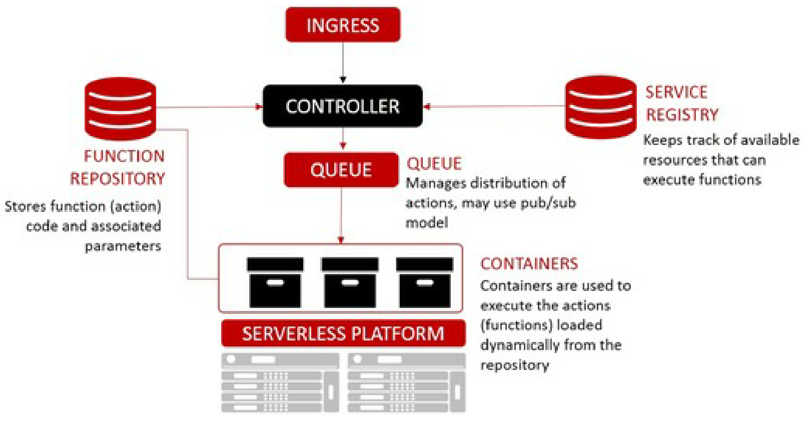
Lori MacVittie, 2017-31-07

You're going to need a platform for automation and orchestration. Serverless might be just what you were looking for.

Serverless (a.k.a. Function as a Service) is still very much in its infancy. And any technology that's just finding its legs tends to need to answer the question "but what is it good for?" That's particularly true in an age when there's some new development or technology clamoring for our attention on a nearly daily basis.

The typical use cases for serverless revolve around its ability to harness the power of cloud for specific types of workloads, like image processing or data mining. But there are orgs using serverless today for more, shall we say, unsexy purposes, like automating operations.

In fact, I might argue (and I think I will because, me) that serverless is perhaps one of the most efficient means of automating operations today. Here's four reasons why I'm going to suggest you seriously consider serverless as a platform for automating operations.


INGRESS
CONTROLLER
FUNCTION REPOSITORY
Stores function (action) code and associated parameters
SERVICE REGISTRY
Keeps track of available resources that can execute functions
QUEUE
QUEUE
Manages distribution of actions, may use pub/sub model
CONTAINERS
Containers are used to execute the actions (functions) loaded dynamically from the repository
SERVERLESS PLATFORM

GENERALIZED SERVERLESS ARCHITECTURE

1. It's event-driven

In an event-driven system, actions cause things to happen. Events (sometimes called *actions*) can be triggered in any number of ways, but most likely would be invoked via a REST API call either through a command line (think 'curl') or from a simple web interface. This includes tasks like provisioning a service, or updating a configuration, or disabling a firewall rule. Basically, the deployment of applications into production – and subsequently its requisite network and app services – is decomposed into a series of "actions". Many actions in the pipeline are triggered by the completion of a previous action, which maps nicely to "action sequence" concepts in serverless environments like OpenWhisk. Deployment orchestration is, after all, the automation of a process that comprises many smaller, individual automated tasks. Each one might be properly represented by an API call into a serverless environment, with an overarching API call initiating the entire sequence.

2. It promotes infrastructure as code and re-use

Templates and scripts fit the notion of infrastructure as code, but a serverless-based system would existentially meet the criterion. Doing so further promotes re-use of deployment artifacts and constructs by encapsulating them as code injected into the process. That means predictable, repeatable, and consistent processes that can be measured and optimized over time. A "one task, one function" approach further encourages composability and enables a more fluid creation of the process that can be easily driven by injecting different functions based on a dynamically generated pipeline.

3. It's "serverless"

Now, we all know serverless isn’t literally without servers. But the same concept – that the underlying hardware (and software) infrastructure is “invisible” to those using it – that makes it appealing to developers should make it appealing to infrastructure and network operations, as well. Remember, once you start automating and orchestrating the deployment pipeline, you’re going to need software and systems to manage it. That means servers and software that must be licensed, maintained, managed, scaled, secured – well, you get the picture. You already know how taxing it is to manage consumer-facing apps, imagine the same for operations-facing apps. A serverless infrastructure, once established, provides a consistent platform upon which you can build any number of workflows without worrying about the underlying infrastructure. Which means the same benefits that attract developers to serverless can be realized by operations as well.

4. It’s poly-everything

You might think what the heck does that have to do with it? Well, lemme tell you. No data center runs on a heterogeneous infrastructure, and even within the subset of a single vendor, organizations run multiple models and versions of hardware and software alike. One of the challenges organizations face is managing the variety of devices and versions across the environment. Most serverless environments already support a wide variety of languages and toolsets (one action can be written in Python, another in node.js) as well as “image-based” actions, which are containers that run, well, whatever. In a world where you’re trying to orchestrate a process comprised of tasks automating such a diverse set of systems, the ability to use the tool that works best (and maybe the only one) is a boon to operations.

The thing is that unlike most applications which are constantly being accessed and used by consumer and corporate users alike, operational tasks are executed infrequently and sometimes (though we don’t like to admit it) without much warning as a response to security or availability incidents in the broader environment. That means an event-driven system like serverless seems like a good fit. It provides an “always on” platform that can execute a wide variety of tasks across the entire operational spectrum. One minute it might execute a security-related task – update a firewall rule. The next, it might be firing off an action to inject a new app service – say a WAF – into the data path of an application as a response to a zero-day exploit in need of immediate redress. The same platform can provide the mechanism to execute just about every operational task required, in an automated and extensible way.

Events might even be triggered by ticket creation or a bot command from a ChatOps-enabling system like Slack, and automatically pull the required information from the ticket or command to use as the appropriate actions are invoked to complete the task.

Serverless is focused on developers, but the underlying principles and mechanisms make it an attractive alternative to building your own operational orchestration systems or relying on a loosely coupled set of systems that bring with it their own integration and operational challenges.

If you’re just starting out on your operational automation journey, you might want to take a long hard look at serverless platforms for the enterprise to see how well it might suit your needs.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | wwww.f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com