Why Should DevOps Practitioners Care about Per-App Architectures?

Lori MacVittie, 2018-27-08

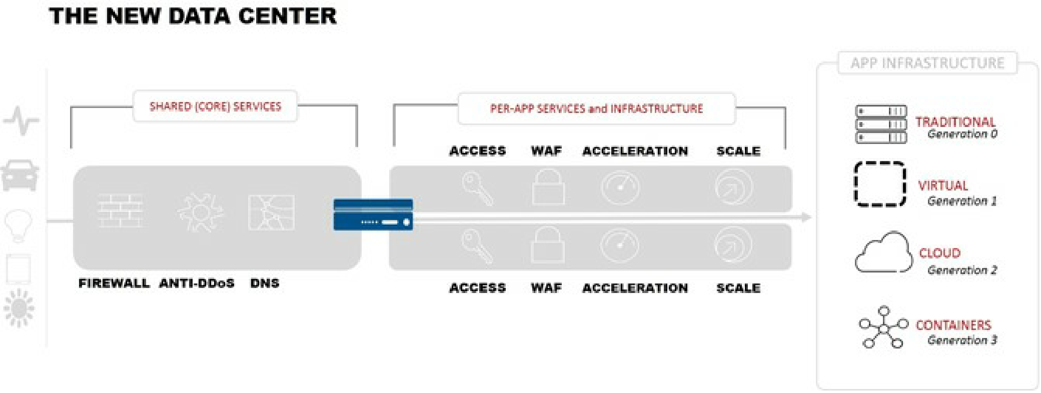
Implied in that question is the conclusion that you should.

For decades now we have been deploying the application services that comprise much of the production pipeline on shared platforms. Application services like load balancing, application performance services, WAF, and federated identity are provided by NetOps on shared infrastructure often in the form of an Application Delivery Controller (ADC).

But time – and demand – along with emerging application architectures like microservices - are forcing changes in that pipeline. These changes are good, both for NetOps and DevOps alike, because they are moving to a model that aligns more closely with deployment schedules and operational practices like infrastructure as code.

There are some application services that remain firmly entrenched on shared infrastructure. DNS. Defense against volumetric DDoS attacks. Network firewalls. These services are by nature ***corporate*** services. That is, they are designed to protect, defend, and serve the business. If these services fail? No productivity or profit across the business.

But other application services are highly application affine; their configurations, APIs, and services are often configured specifically to support a single application. Not a single architecture, but a single application.



Responsibility for these services are increasingly shifting left toward application operations (DevOps) and engaging the developers that deliver applications. And while a shared model can (and often does) still work to provide the delivery and security application services necessary, a per-app model for many of them is desirable for several reasons.

First, a per-app model neatly resolves deployment schedule conflicts. Enterprise organizations are often constrained by shared infrastructure conflicts with respect to configuration and deployment of the application services it supports. Shared infrastructure means shared resources, which invariably raises the potential for a clash of configurations. By eliminating that possibility, resistance to more frequent or off-schedule deployments can be mitigated. After all, if an application services is dedicated to my application, and running on its own platform with its own resources, the conflict is neatly resolved. My app, my risk.

Second, a per-app model supports emerging automation efforts that include putting into practice methodologies like infrastructure as code. By dedicating resources and platform to a single application, the configuration includes only that application. Immutability can be enforced, and the risk of entropy avoided. In fact, I'd argue (and recently did) that you can't properly do immutable infrastructure ***without*** a per-app architecture. This approach enables inclusion in a continuous deployment pipeline that extends beyond application infrastructure and into "the network." Providing this capability means shifting the responsibility of configuration and ongoing management from NetOps to DevOps. The good news is that with that greater responsibility comes greater control, as DevOps becomes the "owner" of the application service and is empowered to upgrade, patch, and re-deploy on its own terms and schedules.

Lastly, a per-app model is immediately more portable. Without being tethered to a shared platform and relying almost exclusively on deployment artifacts (configurations and templates), the bulk of the production pipeline becomes able to migrate from cloud to cloud, on-premises to off, with far less friction and effort. That freedom offers organizations the ability to choose the best cloud and location for the application in question without being constrained by costs associated with migration.

DevOps can benefit greatly from encouraging the adoption of a per-app architecture for application services, both on-premises and in the public cloud. With NetOps under pressure to provide greater control and visibility to other operational domains, now would be a good time to grab some pizza and beer and have a conversation with your NetOps counterparts about moving to support a next-generation, application-centric per-app architecture.