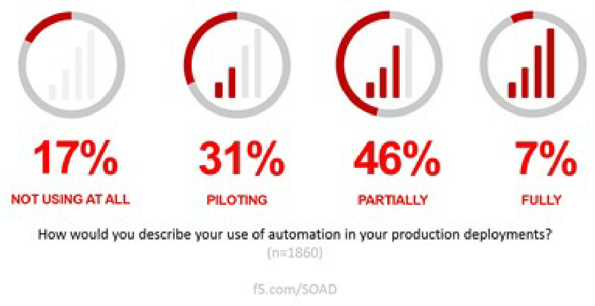
Is there a Minimal Viable Deployment for NetOps

Lori MacVittie, 2018-20-08

Is it really all or nothing for network automation?

Network automation - the practice of DevOpsing the production pipeline - is already in use by a significant percentage of organizations. While very few are fully engaged, the majority (77% according to our latest State of Application Delivery) are either piloting or partially using automation in production.



One of the concepts tightly coupled with DevOps - and thus often tied to NetOps - is the notion of a minimum viable product (MVP).

It's part of the Agile methodology, and it's used as a way to speed up development cycles and get solutions to market faster. That's something we desperately need in "the network". You might recall the Appian survey referenced in a previous blog that slapped us with research that said 72% of respondents lacked confidence in IT to scale to meet the needs of the business.

Ouch. Despite the fairly extensive use of automation in IT, developers and business stakeholders still lack confidence in our ability to get 'er done.

So adopting tools, technology, and methodologies from DevOps to speed things up (by scaling smarter) isn't all that crazy. But before we can figure out how to apply MVP to the network, we gotta understand what it is. So for those unfamiliar with DevOps, Agile, or MVP, here's a straightforward definition from the Agile Alliance:

A minimum viable product (MVP) is a concept from Lean Startup that stresses the impact of learning in new product development. Eric Ries, defined an MVP as that version of a new product which allows a team to collect the maximum amount of validated learning about customers with the least effort. This validated learning comes in the form of whether your customers will actually purchase your product.

A key premise behind the idea of MVP is that you produce an actual product (which may be no more than a landing page, or a service with an appearance of automation, but which is fully manual behind the scenes) that you can offer to customers and observe their actual behavior with the product or service. Seeing what people actually do with respect to a product is much more reliable than asking people what they would do.

A team effectively uses MVP as the core piece of a strategy of experimentation. They hypothesize that their customers have a need and that the product the team is working on satisfies that need. The team then delivers something to those customers in order to find out if in fact the customers will use the product to satisfy those needs. Based on the information gained from this experiment, the team continues, changes, or cancels work on the product.

-- Agile Alliance, "Minimal Viable Product"

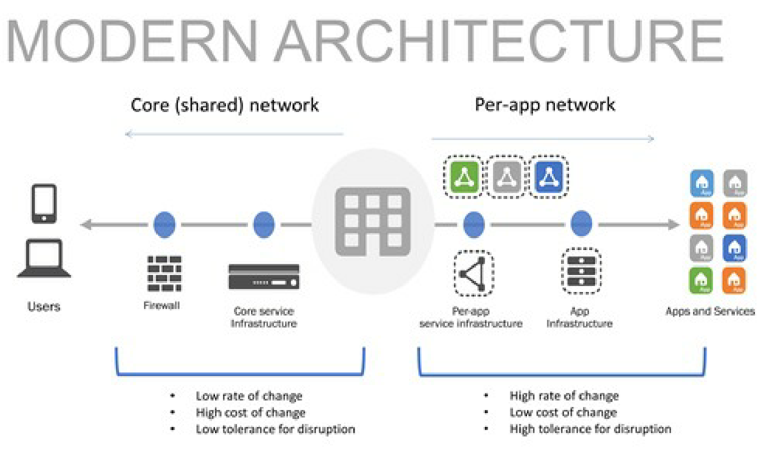
This is the point in this treatise where I note that many of the concepts associated with DevOps (and its related technologies and methodologies) do not always translate well when applied to a NetOps initiative. Experimentation is not a term engineers, architects, or executives in IT use when discussing making changes to the network.

The blast radius, you see. It's large and in charge. Most organizations do not have a high tolerance for operational risk, with good reason. Outages cost real money - and sometimes, jobs. The network is not really a good place for experimentation.

But that doesn't mean that there isn't a minimal viable deployment (MVD) for NetOps.

Defining a MVD for NetOps

The production pipeline today is comprised of both shared resources like switches, routers, DNS, and multi-cloud routing (GSLB) as well as per-app application services such as load balancers, WAF, and application access control.

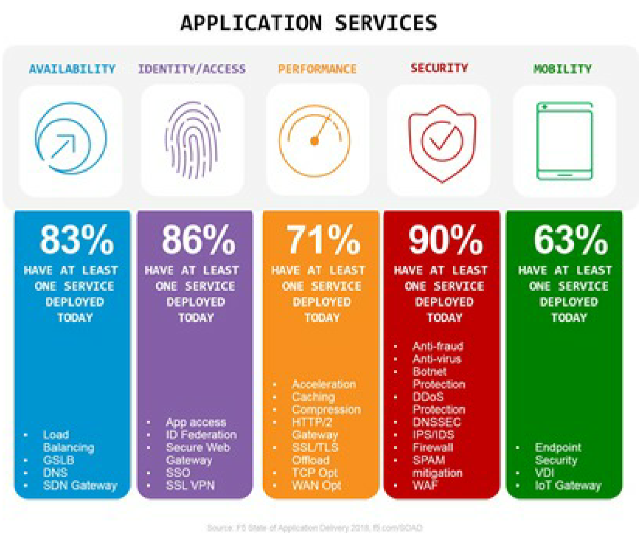


Interestingly enough, if we look at the rate of change associated with shared resources, we'll find they're fairly nominal. That is, they have a low rate of change. That's good, because they have a low tolerance for disruption, as well. Jump over to per-app resources and you'll find a higher rate of change with a greater tolerance for disruption.

That's one of the benefits of a per-app architecture, after all - isolation of the data path that protects other apps from disruption when something goes wrong.

Along that data path are the average sixteen different application services our research tells us organizations use to deliver and secure their applications. Some of those - like a network firewall and DNS - are shared resources. Others are not, or at least they don't need to be. They might today be deployed on a shared platform, but they could be architected into their own data path if you had a good reason to do so.

Which, of course, is what I'm going to give you.



The good reason is that you can effectively develop a MVD for an application if you adopt a per-app architecture for those application services that are tightly coupled to the app in the first place.

As our definition of an MVP tells us, the "product" (in our case, an app deployment) does not need to be fully automated. If we operate on the premise that the riskiest, least-tolerant resources must continue to be configured (and verified) manually, we still gain ground. Firewalls and core services like DNS have a very low rate of change, so we can assume that manual methods are not going to significantly impact the deployment timeline. That's even more true if we automate the bulk of the per-app application services because then we're freeing up time for operators and engineers to make the manual changes if need be.

Assuming that the ratio of core (shared) services to per-app application services is about one to three*, that means our average organization has at least four shared resources to manage manually and twelve per-app resources to automate.

Looking at the extensive list of application services (we're currently tracking thirty distinct services) we'll note that some of them are necessary to deliver or secure an app (DDoS, WAF, load balancing for scale, app access) while others are more, shall we say, enhancements. That'd be application services like performance-improving acceleration options or productivity enhancing single-sign on (SSO).

So if we were to consider the implementation of a MVD, we might take an Agile approach and focus on those application services that are critical to day one delivery and security. It doesn't mean we're ignoring the enhancements, it just means we're going to initially focus on the critical ones and get them automated first. We can still manually manage those services that improve productivity or performance, but for a MVD we want to zero in on profit-impacting services.

An MVD Approach to the Agile Network

Approaching automation with the intent to define and deliver an MVD means we're moving faster (we're more Agile) and gives us the opportunity to iterate on the automation to improve and expand it with each sprint (measured in weeks, not quarters) until we've got a solid, sustainable product (automated deployment).

Adopting an MVD-based strategy to automation requires commitment to not only an architecture but an approach and attitude that focuses on applications. That's because this kind of an approach requires an understanding of the application and its needs from both an operational perspective and a business point of view. The MVD for one app may not match the MVD for another. That's one of the reasons the per-app architecture is such a critical component when transitioning from a fixed, manual network to an agile (automated) pipeline.

So it turns out there is a minimal viable deployment for NetOps. Which means you can take an Agile approach to network automation – one that will be infinitely faster (and safer) if you transition to a per-app architecture as part of your agile network initiatives.

Automate (almost) all the network things.

*that's totally a SWAG based on the list, my experience, and my (strong) opinion. Your mileage and definition may vary.

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

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