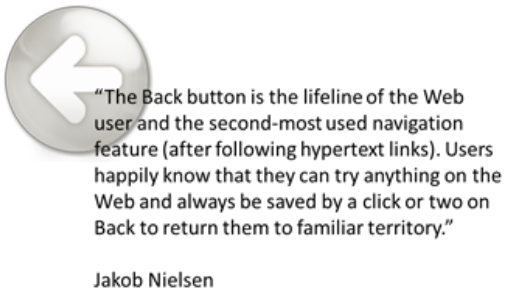
DevOps: Where's Your Back Button?

Lori MacVittie, 2013-15-07

#devops There is no back button on deployment. But there can be.



It was nearly 10pm on a Saturday night. A handful of architects and developers sat around a cube, munching on cold pizza and chatting while waiting for the phone call that would kick off testing. The phone call came, but not to begin testing. It was the dreaded call of failure. Something had gone wrong during the push into production and we, at least could go home while operations took on the dirty task of backing out the changes.

At least we had a plan. We knew failure was an option going into the production environment, and there were scripts and procedures in place to make the task if not pleasant then at least a bit less onerous.

I was reminded of the need for rollback plans (and that very long Saturday night) a few weeks ago when a friend accidentally made public a direct message on Twitter, and then had to explain that tweet when it started showing up on Facebook (amongst other places, I'm sure). You see, it's interesting that we have a wealth of tools to automate propagation of tweets and status updates and social data across multiple networks, but there's almost never a "take back button". Oh, you can certainly delete it in the system that triggered the process, but that process doesn't continue. The Tweet may be gone but there's still the Facebook status and a LinkedIn status and <insert social network here> update.

There's no back button on social network integration. Nor on continuous delivery processes. But there should be.

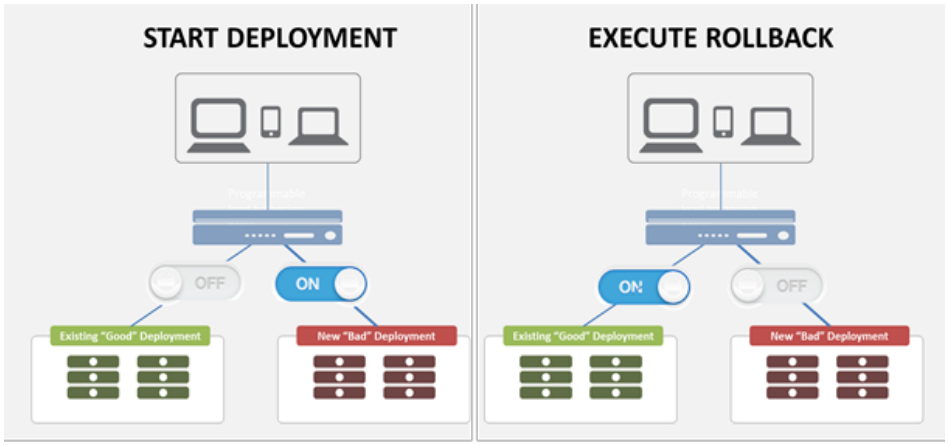
Luckily there are ways to assist in ensuring that continuous delivery doesn't devolve into continuous down time.

A Deployment "Back" Button

Most of the time when we talk about continuous delivery or continuous deployment we talk about the forward motion. How do we coordinate across network, devops, and developers to ensure releases can be delivered at the touch of a button (the enter button, to kick off a process, of course)?

But we don't make much conversation around rolling back, on ensuing zero down-time redeployment and rollback of releases that for whatever reason just didn't cut it.

There are a variety of options, and most of them begin and end with version control and configuration management solutions. The problem with these options is they aren't going to achieve zero (or at least near-zero) down-time rollbacks. They can't, because it takes time to walk back a process, step by step, and restore configurations. In the mean time, other parts of the network and application infrastructure relying on connections or next hops may be stymied by a misconfiguration. Which means down-time.



Using a programmable load balancing proxy to enable a side-by-side deployment. If the need to rollback arises, a simple "switch" in the proxy will revert to the "good" deployment, allowing near zero-downtime rollback.

Depending on which application is being rolled back, that could be a Very Bad Thing(TM).

It's not that you shouldn't use version control and configuration management systems. In fact, those should be tools you're using today (if you aren't, make it a point to start shopping around). But in order to realize a more zero down-time rollback process you may need a little architectural help. Help from a toggle switch sitting in the network that allows you to nearly instantaneously reroute from the "bad" deployment to the "good" deployment, and thus restore normalcy.

This is the type of scenario a load balancing proxy is uniquely suited to enabling. A load balancing proxy views each deployment as a "pool" of resources and can be directed use pool A or B to select the appropriate service. This means by configuring two pools - one for each deployment - you can effectively switch between them. You can either take a configuration based approach, i.e. assign only one of the pools at any given time to the associated VIP (Virtual IP - virtual server, in other parlance) in the configuration, or use data path programmability to codify the choice (thus leaving the configuration always intact). A very basic "select $resource from POOL_A" (in the appropriate language, of course) will force the load balancing proxy to always choose its resource from POOL_A. Change POOL_A to POOL_B and voila! Deployment is switched.

By also ensuring it has a programmable management plane, i.e. you can control and configure via an API, the process of executing a rollback can be built into the deployment process. In effect, you **can** build the "back" button for deployments.

"Built to fail" is a commonly heard mantra today in architectural circles thanks to cloud and virtualization. It should be as common in the realm of DevOps.

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

©2016 F5 Networks, Inc. All rights reserved. F5, F5 Networks, and the F5 logo are trademarks of F5 Networks, Inc. in the U.S. and in certain other countries. Other F5 trademarks are identified at f5.com. Any other products, services, or company names referenced herein may be trademarks of their respective owners with no endorsement or affiliation, express or implied, claimed by F5. CS04-00015 0113