Scaling continuous delivery: Shift-left performance to improve lead time and pipeline flow!

An excerpt from About:Performance blog

About the Author

Andreas Grabner has been helping companies improve their application performance for 15+ years. He is a regular contributor within Web Performance and DevOps communities and a prolific speaker at user groups and conferences around the world. Reach him at @grabnerandi

Best Practices

Transforming from Agile to Continuous Delivery by embracing DevOps Best Practices and a high degree of Automation is a desirable goal for many IT organizations. The biggest challenge I've seen when recently talking with companies such as Capital One, BCI, BCP, BBVA, and even our own engineering teams, is scaling that transformation. Just as Adam Auerbach (Sr. Dir Continuous Delivery and DevOps at Capital One) explained in our recent PurePerformance Podcast: *Automating the flow of code changes from Commit to Production for a few changes per day is just the first step. SCALING this from a handful to 20, 50 or more teams is the CHALLENGE.* Pushing new builds every couple of minutes will potentially uncover your pipeline's bottlenecks, particularly when you try to run thousands of unit-, integration-, functional- and potentially-performance tests per commit that all take more time than it does for the next code commit to come in. This slows down the flow and increases lead time!

In the Podcast Adam talked about the challenges of onboarding new feature teams to their Continuous Delivery Model. They came up with their own DevOps Dashboards (open sourced via Hygieia on Github). It visualizes the flow of builds through their pipeline split by team showing the flow of code from *Commit* all the way into *PERFormance Testing* and then into *PRODuction*. They monitor average execution time per phase and the wait time for a build to enter a phase. Due to the high volume of builds being pushed through the pipeline the following screenshot shows that it takes 1d 4h 37m until a code commit of a developer makes it all the way to the *PERFormance* phase. This is no longer an acceptable Lead Time nor do they consider this fast feedback to engineers:

Capital One's DevOps Dashboard showing build flows through their pipeline. Also highlighting the bottlenecks in their pipeline!

There are two approaches to mitigate that issue, especially for the *PERFormance* phase:

#1: Increase Throughput: Run Performance Tests in parallel using Docker

#2: Shift-Left Performance: Identify Performance Problems in early phases

#1: Increase Throughput

I really encourage you to listen to the PurePerformance Podcast from Adam or watch some of his presentation he gave at conferences such as StarWest or Velocity. One aspect I found interesting was that they are re-using the same Selenium scripts they use in the functional testing phase for load and performance testing. They simply spawn many Docker containers on AWS and run their Selenium tests in parallel. This is an interesting way to re-use existing test scripts for load testing instead of re-creating the same scenarios in a separate load testing tool.

Not only can you do this to run a single load test against a build. They also use the same approach to run multiple of these tests in parallel against multiple builds waiting to be load tested. With the flexibility of AWS (or other cloud providers) it is easy to spin up and down these test environments.

Definitely one way to increase throughput of the performance testing phase!

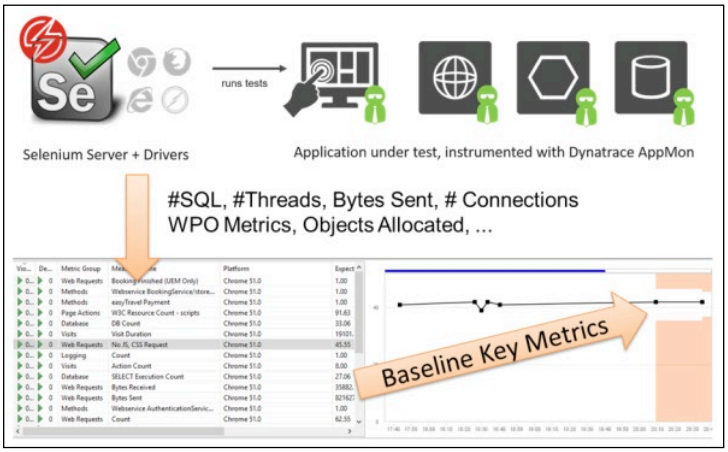
#2: Shift-Left Performance

If you have followed my earlier blog posts, my Performance Clinic or my DevOpsDays Boston 2016 session on Metrics-Driven CI/CD you will not be surprised to read that "Shift-Left Performance" is the approach I've been promoting for a while.

The goal is to find performance related issues earlier in the pipeline and only allow "good builds" to make it all the way to the later pipeline stages. How? By simply capturing architectural, performance and scalability metrics while you execute your unit-, webapi- and functional tests in your earlier pipeline stages. These are tests that execute very fast but which are mainly used for functional verification only. I suggest that by also capture important architectural metrics (# of Objects on a Page, Total Bytes Transferred, # of SQL Queries Executed, # of REST Calls ...) and baseline these metrics across tests and builds you automatically converted your functional tests into performance tests.

The concept is not new. There are also enough open source tools out there that can capture some of these metrics. Yet other tools to store these metrics and another set to identify regressions based on baselining these metrics over time. With Dynatrace AppMon & UEM we provide native support for capturing, baselining and analyzing these metrics for your Unit, HTTP-based Web/REST-API or UI-Driven Browser Tests. No need to stitch together tools!

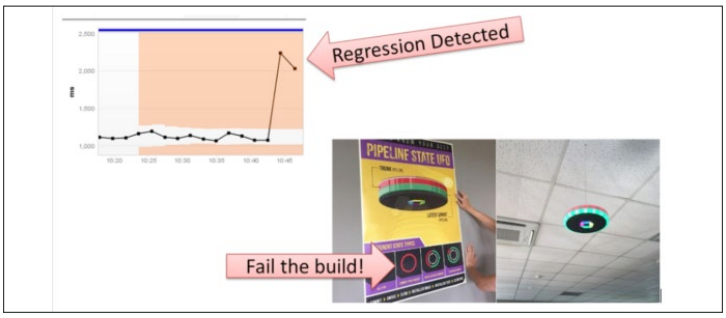
The following illustration shows one use case where we analyze Selenium Tests executed via Sauce Labs. The application under test is instrumented with Dynatrace AppMon & UEM. With a simple Selenium script change we can pass the test name to Dynatrace. Dynatrace then captures Browser Metrics (# of Images, Page Size, DOM Nodes, ...) as well as Server-Side Metrics (# of SQL, # of Exceptions, ...) through its PurePath technology. All metrics are automatically baselined!



Dynatrace automatically captures and baselines all key web performance and server-side metrics across all builds and tests.

After a couple of builds Dynatrace has established a baseline and alerts on regressions per metric for each test case. This allows you to identify architectural regressions minutes after a developer made a code or configuration change that causes an increase in Page Size, # of Images, # of SQL Statements ... of the feature he/she is working on. This also works well for Unit and Integration Tests that test against mocked services as we mainly look at "Count" metrics such as "How often did we call a certain Interface, API ..."!

Through the Dynatrace Test Automation REST API and the plugins we provide for most common Build Servers (Jenkins, Bamboo, Go, Team City, TFS, Councourse ...) you can stop the build in your pipeline much earlier than before. In our Dynatrace Engineering Labs we even hook this up with what we call the "Pipeline State UFO" which represents how far a build made it in the current sprint and master branch:



Fail faster by identifying regressions on key architectural, performance and scalability metrics.

Fail faster by identifying regressions on key architectural, performance and scalability metrics.

Stopping bad code changes early will remove bottlenecks you currently have or will have in your pipeline if you keep pushing more code changes from dev into performance and prod. I strongly believe that this is a key approach to scaling continuous delivery in larger organizations.

The following screenshot shows the potential improvement of a pipeline. By stopping bad builds earlier you not only take pressure off the later phases. You also improve Lead Time from Commit to Production.

This illustrates how Shift-Left Performance could look like in your pipeline and how it optimizes your throughput and lead time.

Start scaling your Pipeline

If you are moving towards Continuous Delivery make sure you think about how you can scale this once you have to onboard more teams. Shift-Left Performance is an easy to implement concept. Try it yourself by following these steps:

1) Get your own Dynatrace AppMon & UEM Personal License
2) Follow the Dynatrace Selenium or the Sauce Labs Tutorial on Github
3) Watch my DevOpsDays Boston presentation on Metrics Driven CI/CD
4) Provide us with feedback through answers.dynatrace.com or @grabnerandi

My colleague Sonja Chevre and I are also did a Live Performance Clinic on UEM-based Test Automation with Dynatrace AppMon & UEM 6.5. We used the Selenium and Sauce Labs examples that you can get on GitHub. Register here to see us LIVE (including Live Q&A) also watch the recorded session in case you miss it on our YouTube Channel.

Don't allow your pipelines to slow you down!

Dynatrace Digital Performance Platform — it's digital business... transformed.

Successfully improve your user experiences, launch new initiatives with confidence, reduce operational complexity and go to market faster than your competition. With the world's most complete, powerful and flexible digital performance platform for today's digital enterprises, Dynatrace has you covered.