News & Announcements

An interview process to learn from - ImageMagick Lessons and ATS contributions

Daniel Morilha

Published October 31, 2018

Tags

apache traffic server cdn open source platform

Technical interviews can be difficult to do well. My interview process at Netlify included some practical work which not only gave me a chance to show how I explore and learn, but also gave me the chance to share what I learned as contributions to the Apache Traffic Server open source project.

Here is what happened

I was interviewing for a position which would involve maintaining, extending and contributing to Apache Traffic Server, a high throughput open source HTTP proxy that

used to work with on a previous gig. Traffic Server is mainly written in C++ and allows customizations to be added in the form of plug-ins.

To discover what aspects of ATS Netlify might be interested in, I explored the contributions the team had previously made. After seeing some activity around the use of ImageMagick, I decided to focus my attention on ImageMagick, a well established library used for image manipulation. ImageMagick allows you to modify any images with all sorts of editing tools. You can crop it, resize it, rotate it, convert it into different formats, write text on top of it and apply different filters. To check all you can do with ImageMagick, I recommend visiting Fred's ImageMagick Scripts web site to see the software being taken to its limits.

ImageMagick can be programmatically used through its extensive APIs. A software engineer writing C++ would use the ImageMagick++ library. Using an API has the limitation of requiring code to be written every time you want to apply a different change into an image. The code written to resize is very different from the code to insert a text into the bottom right corner of a picture. Software engineers are cheap on keystrokes and love to solve problems generically. I am no different. ImageMagick also has a powerful command line tool, which allows you to convert an image based on command line arguments. To flip an image for example, you would execute a command like `$ convert in.jpg -flip out.jpg`.

The Project

After realizing how flexible ImageMagick command line tool is, my weekend plan became clear: What if I connect the powerful ImageMagick command line parser into an URL through a query parameter. This way, I would be able to write different modification scripts and plug them into their image URLs with no modifications into the plug-in itself. It turns out ImageMagick exports all necessary interfaces through the MagickWand API.

The 622 lines of code work by inspecting the `Content-Type` response header from the origin and checking if that is a recognized MIME image format. Then the `magick` query parameter gets decoded back into command-line parameters that are plugged into the MagickWand's API. The origin's response body is copied into a ImageMagick's internal buffer representation. From then on, the real ✨magic✨ starts to happen and once the ✨trick✨ is done, the plug-in ✨reveals✨ the output to the requester. All happens in memory and no external process is spawned. The modified content is also stored into

Traffic Server's cache, making subsequent requests to the same URL skip processing altogether and be served in a blink of an eye.

To make ImageMagick heavy processing demands friendly to the rest of Traffic Server duties, the heavy lifting is done on a thread coming from an internal thread pool and does not block other requests from being served on the event threads. With that said, it is still highly recommended to run the `magick` plug-in in a second tier, right behind your front line cache tier.

In order to make the `magick` plug-in more secure and prevent requests with unauthorized image modification scripts, the plug-in also accepts a `magickSig` query parameter that, when present, asymmetrically signs and verifies `magick` encoded content, only allowing pre authorized scripts to run.

The Summit

Last month I went to ATS Fall Summit hosted at LinkedIn in Sunnyvale, California for two days of great sessions from the whole community. There, I also presented the ImageMagick plug-in. The community's reception was very positive and a week later my interview project was merged into Apache Traffic Server repository and is now available to everyone: https://github.com/apache/trafficserver/pull/4376

Here is an animation composed of pictures of the presentation after running the following ImageMagick command on them: `$ convert -delay 750 \( -resize 512x384 -frame 6x6+2+2 -rotate 23 IMG-5919.JPG \) \( -resize 480x360 -frame 6x6+2+2 -rotate -17 IMG-5923.JPG \) \( -frame 6x6+2+2 IMG-5922.JPG \) -gravity center -extent 800x600 -loop 2 animated.gif`

An interview process like this is far more realistic and effective than some sort of test of what you already know. Our jobs involve learning and growing all the time. I'm delighted that the interview process went beyond getting the role and I could contribute the resulting project back to the open source community. It seems like a productive way to work.

Netlify heavily relies on Apache Traffic Server and many other great projects. If you are looking for a challenge, we are looking for engineers to join our platform team and work on a variety of initiatives.