You should be able to install my software in less than one hour – or why DevOps is important

Martin Fenner, Gobbledygook

April 14, 2013

Cameron Neylon yesterday wrote a great blog post about appropriate business models for shared scholarly communications infrastructure. This is an area I have also been thinking about a lot recently, and in this post I want to add a technical perspective (and an announcement) to the discussion.

DevOps is an important trend that brings software development and administration of IT infrastructure closer together. Agile software development, server virtualization, cloud infrastructure and software automation tools such as Chef, Puppet or CFEngine are an important pars of DevOps, but it is really the collaborative aspect of IT administrators working much closer with software developers what defines DevOps. The end result is often faster and more stable software releases, and that is what is users and customers care about.

This makes DevOps particularly relevant for all areas where innovation is important, and that of course includes tools and services for Open Science. We not only need infrastructure that facilitates software development (with services like Github, among many others), but we also have to streamline IT administration. The question is not whether you do your development in Java, Python, Ruby, PHP or Javascript, but how well you integrate your software development and IT administration. The shift towards web-based tools has centralized software installation and updates, but these web-based services are becoming increasingly complex and difficult to set up and administer. Running an institutional respository, research information system or a journal is a complex task. The software may be freely available as open source (e.g. Dspace, VIVO or Open Journal Systems), but the resources required to run such a service still make this a big investment.

Two solutions to this dilemma are to pay either a vendor for installation and maintenance, or to use the software as a service (SaaS) that is hosted somewhere else. Why these two options are popular, they may not always be the best choices because they mean that you are locked in to a particular vendor or service provider, and that you may give expertise and direct access to your data away. I believe that these are helpful approaches for auxillary services, but that ideally the core services of a library, publisher or other provider of scientific

infrastructure should not be outsourced. Developing software for scientific infrastructure that you want organizations to install locally should therefore always include work on integration with IT infrastructure, and just providing manual installation instructions isn't good enough anymore.

Article-Level Metrics (ALM) and the related altmetrics are becoming increasingly popular. The collection and display of this information is a complex process, as it requires the integration of information from several upstream APIs which may be temporarily unavailable, have changed their data format, or put up restrictions on how you can use the data. In turn this information has to be processed and aggregated, and then reliably be provided to downstream users. This kind of information gathering fits perfectly with a service provider model, and organizations such as Altmetric, ImpactStory and Plum Analytics. PLOS is collecting and displaying this information with its own tool. The simple reason is that PLOS started doing this several years before the services above became available, and none of them currently provide the same comprehensive set of information about citations, usage stats and altmetrics (although there are of course a lot of things they do better than the PLOS ALM application).

But there is also the question of whether Article-Level Metrics are a core service for every publisher and are best collected in-house. This not only makes it easier to collect information from some sources (e.g. usage stats or CrossRef citations), but also gives unrestricted access to the data in real-time. When I took over as technical lead for the PLOS Article-Level Metrics project last May, I therefore not only worked on improving the ALM application for PLOS, but we are also working hard on making it easier for other publishers to install and use the application. We want to provide an attractive alternative for organizations for which the service provider model is not the best option.

To that end I want to announce the latest feature which allows the automated installation of the PLOS ALM application on an Amazon Web Services (AWS) EC2 instance. This option is great not only for setting up an ALM production service, but because of the EC2 pricing model by hour (about $1 a day for a small EC2 instance) without setup costs is a great way to test-drive the application for a publisher, to analyze a particular set of papers from different publishers for a research project, or to set up a PLOS ALM server for a hackathon or workshop.

There are of course many options to automate software deployment on a production server, including the PaaS (platform as a service) providers Heroku, CloudFoundry and OpenShift, and the recently announced Amazon OpsWorks. I am a big fan of the Vagrant software development tool in combination with Chef for automation, and in March Vagrant added support for Amazon AWS. This makes deployment of the PLOS ALM application to AWS really simple:

1. Install Vagrant and the vagrant-aws plugin
2. Setup an Amazon Web Services Account
3. Check out the PLOS ALM source code from Github
4. run the command **vagrant up –provider aws**

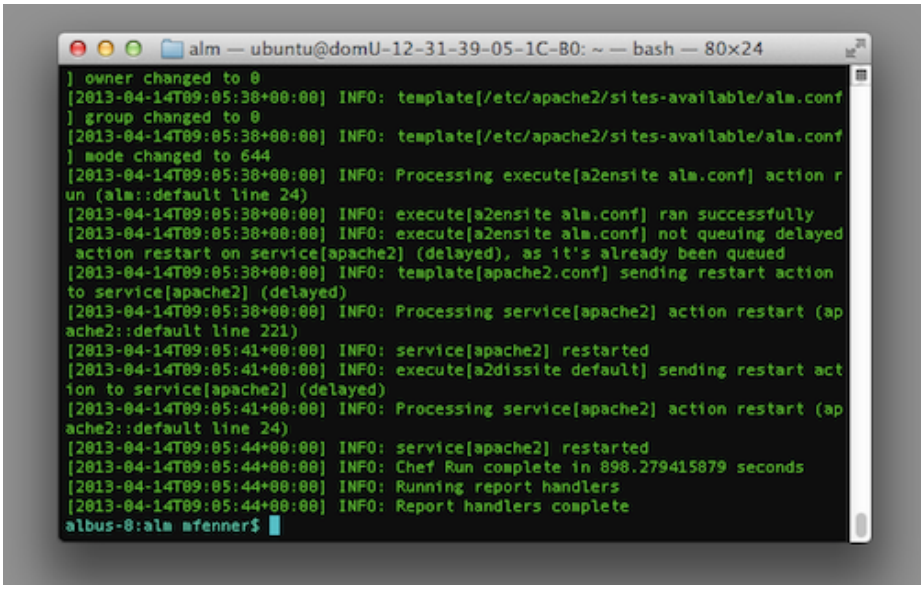

Figure 1:

Step #4 took 898 sec or about 15 min on my computer (see screenshot), and at the end I had a PLOS ALM server where I could access the admin dashboard via the web interface. If you are familiar with Amazon Web Services – you have to think about the right size for the EC2 instance, an appropriate AMI, security groups, elastic IPs, and DNS service – then the whole process should be done in well under an hour. I will use this instance to load and analyze some articles from a publisher for a presentation next week. When I'm done, another command (**vagrant destroy**) will destroy this server and Amazon will stop billing me. During testing I have created and destroyed many servers, and the vagrant-aws video shows you how easy this process is.

At this stage the installation process is working (and has been working for a local Virtualbox install for many months), but needs testing and documentation. I therefore invite everyone interested in testing this out to contact me so that we can make this well-documented and working reliably.