Getting Started with Configuration Management

CORY LUENINGHOENER

Cory Lueninghoener helps keep some of the fastest computers in the world running at Los Alamos National Laboratory. He hosts the Getting Started with Configuration Management BoF at LISA and has been active in the Bcfg2 development community since its early stages.
cluening@lanl.gov

Configuration management tools take time to set up and maintain, but using one is well worth the effort. In this article I will present technical and social problems, solutions, and advice related to getting started with a configuration management tool. Many of the problems I will cover are much easier to fix when tackled early on, while the advice on where to actually get started should help ensure a smooth deployment.

This article has grown out of the Getting Started with Configuration Management BoF that I have led at LISA for the last three years. During that time we've talked about a lot of issues people run into when getting started; too many, in fact, to write about in a single article. Here, I'll just look at the very beginning problems: convincing yourself and your coworkers that you need a configuration management tool, where to get started, and how to plan for the future as you start out.

Why?

A configuration management (CM) tool is a robot that does work for you, keeping track of the files, packages, services, and other pieces of machines in your environment and keeping them up-to-date for you. But why should you use one? Why spend the time up front to deploy the tool and later to keep it up-to-date? In anything beyond a trivial case, a solid configuration management infrastructure will give you faster machine deployment, faster disaster recovery, and greater flexibility than by-hand methods. Everybody has had the experience of bringing up a machine by hand, making a series of "I'll only do this once" changes to it, and then needing to do the exact same thing again two months later. Even with careful notes, it is easy to miss details. By consolidating all of your host configuration to a central repository, bringing up the second (or third, or hundredth) copy of that original machine is a trivial task: just tell your CM tool to build another machine with the same class as the original and make the minimal changes needed for the new hardware. The same is true for disaster recovery: if you lose the disk in one machine, bringing it back can be as simple as popping in a new disk, telling it to boot off the network, and letting the CM tool do the rest. If you've put the effort in up front, that machine can be back doing its job without any further work on your part.

It should be clear that spending the time to deploy a tool and keep it up-to-date will eventually help you out and save you a bunch of time down the road. But there are plenty of other benefits of having a full CM tool deployed that you won't even think of until *after* you've been using it for a while. You'll gain greater flexibility in

your ability to bring up new machines, as you won't need to design each one from scratch every time. Need to bring up an SSL-enabled Web server? Model it after the plain one you already have. Need to bring up a mail server running Ubuntu 11.04 instead of your current 9.10? Start with the configurations you already have and make the minor tweaks needed for the new software. Having all of your configurations in one repository gives you great flexibility to mix-and-match pieces in a way that would be very difficult in an unmanaged environment.

What about documentation? Do you find it hard to make time to write accurate documentation about your current set of machines? Consolidating your infrastructure into one configuration repository does a lot of the work for you: you get a list of machines you run, what jobs they do, and exactly how they are configured all stored in one place. Handing a copy of the repository to new hires gives them a complete view of your network in its current state without them needing to track down every machine owner to find out what exists. Similarly, being able to point auditors at the current state of your network at any time becomes possible; instead of telling them what packages and configurations you think are installed and what machines you think have been retired, you can show them exactly what you have *right now*.

But!

Since deploying a configuration management tool can be a far-reaching project, you will likely run into objections related to time, your environment, and your coworkers. This section may sound negative, but it is a reality you need to be prepared for: addressing roadblocks early on is the best way to ensure a smooth deployment.

Time

The problem of time plagues every project, and your tool deployment is not going to be an exception. The obvious big time sink is the initial one: picking a tool, learning to use it, and deploying it across your infrastructure. Eventually you will also need to worry about the maintenance costs of teaching your coworkers to use the tool and keeping both the repository and the configuration management software itself up-to-date. For this article I'll just focus on the early difficulties, as their solutions map nicely to the later ones.

Picking a tool is mainly a matter of preference. There may be tens to hundreds of tools out there in various stages of completeness, but it is easy to name "the big four" that currently have very active development and large install bases: Cfengine [2], Bcfg2 [1], Puppet [4], and Chef [3]. All of these tools are mature and (perhaps with a little extra coding) can do anything you need to run your environment. If you are strapped for time, rest assured that you can't go wrong with one of these four. Descriptions of each appeared in *;login:* [7], and each of them (as well as many others) were compared in depth for LISA '10 [5]. Check out the language, style, and community of each and pick the one that fits your or your organization's ideals.

Finding time to actually plan and deploy a tool is the next difficulty. Time management itself is completely outside the scope of this article, but the best advice I can give is also shared by experts [6]: "Just start. Once you get started, it won't be as difficult as you thought." At the LISA BoF that inspired this article, I jokingly told people that after ignoring their day-to-day work for one week at the conference they should also ignore it the next week and use that time to get a base configura-

tion management system installed. The point is to make use of the conference momentum to get started on the new task before real life sneaks in again.

My advice to readers of this article is similar: once you finish reading this, decide you aren't going to get anything else done for the rest of the day. Download one of the tools mentioned above, install it on one of your machines, and start playing with it. You won't get your whole environment under configuration management control in one day, but making it over the hurdle of downloading and installing the tool will get you far in the process. After that it is much easier to do incremental work as you move toward complete control. The important thing is to make the time *now* before falling into the trap of routine and never getting started.

Environment

Your next big hurdle is likely to be your environment. Preexisting scripts and already deployed machines are the two biggest problems here. Fortunately, these are also relatively easy to fix with the right mindset.

If you currently manage parts of your infrastructure with homegrown scripts, don't throw them out. All of the major tools have ways to call shell scripts, and in some cases that is the best way to get a task done. In most cases, however, you will want to translate those scripts into your tool's native language or process. Doing so will let you take advantage of the perks of using a consolidated tool: comprehensive reports, sanity checking, abstraction, and predictable behavior are all features to look forward to. But keeping your old scripts around to translate from and check against is very useful in the beginning.

Having a large set of already-deployed machines will make any deployment take longer than starting from a fresh slate, but has its advantages. You'll want to start out by spending some time on a whiteboard categorizing your machines according to their similarity, and then begin by focusing on one machine from each category. As you progress with each, aim to make your configurations as generic as you can. You should find that large parts of your configuration will fit all of the machines, and that the individual machines need relatively little personalization. By focusing on the smaller number of machines and the general configuration first, you will find yourself making great coverage over your environment with much less work.

Coworkers

No matter how well your machines take your tool deployment, you are almost guaranteed to get objections from your coworkers. They have a right to do this: you are deploying something that they see as making changes on their machines without telling them. Especially for people who are intimately familiar with all parts of your infrastructure, this can be a very unwelcome change. Plain old communication is often the best way to defuse this situation: tell your coworkers what you are doing, send them documentation to look through, invite them to "shoulder surf" while you migrate a service into your new tool, and keep them up-to-date as you make progress. Your goal here is not to force your tool on them, but to get them to jump on the bandwagon too. The four big tools (and plenty of the smaller ones) are all great pieces of software. Don't be afraid to help your coworkers see that.

Where to Start

Okay, you're convinced. When you reach the end of this article, you're going to set the article down, download a tool, and install it. But you have 300 machines, each of which has 1500 packages installed, and each of those have their own configuration files, services, directories, and other details to care about. After the tool is installed on a test machine, where do you start with your configuration?

There are three important things to keep in mind when getting started: start simple, start safe, and start with something that others won't argue over. Starting simple is obvious, but easy to forget; it is very tempting to make a big splash with the new tool to prove to yourself or your coworkers what it can do. Instead, remember what they say: "Think globally, act locally." Start simple, getting to know your new tool and planning for the future as you go along. It will save you a lot of refactoring later.

Similarly obvious but easy to forget is the need to start with safe configuration elements. Everybody invariably makes mistakes with their CM tools, and nothing derails a project quite like a spectacular failure early on. Remember, your CM tool is like a new robotic employee that needs to learn your environment. If you wouldn't trust having an intern try something during his or her first week, don't try to make your CM tool do it during its first week either. Start by configuring redundant or seldom-used services, packages, and machines. As you become comfortable with your new tool and prove to those around you that it is safe, move on to the more dangerous (and generally more important) areas of your infrastructure.

Finally, and perhaps most importantly, start with something that won't start an argument. It is perfectly normal for coworkers to be wary of a new tool that will make changes on its own, and it is important to not turn them off to the tool early on. Starting with a set of machines or packages that you "own" and that others won't be upset at you for changing is a great way to prove that your new tool works and that it should be deployed on a larger scale. Keep others informed about your progress (and mistakes) so they see how great the tool is. Eventually you'll need to convert them, and you don't want to make them hate your new tool before you make any progress with it.

So, what configuration items should you start with? If you're really paranoid, start by managing /tmp/foo.conf on all of your machines. This is (hopefully!) not actually used anywhere, so you can put whatever contents you want in it. If you put a current timestamp in it, you can easily update it every morning for a couple of days and monitor how it propagates through your machines. This file fits all of the above requirements too: it is simple, safe, and not likely to be argued over by anybody else.

If you're a little braver, try managing /etc/motd on all of your machines. This is another safe, simple file to manage (in most environments, nothing should break if you mess up the message of the day), but in this case it is something visible to others. This is one of those simple confidence-builders—when you've been managing a file that everybody sees when they log in to a machine with no complaints, you know you're starting out well.

For a more complex task, try managing NTP with your tool. This is a great piece of software to manage, as it includes the three big pieces of configuration: a package, a configuration file, and a service. This gives you a perfect chance to become

familiar with how your tool handles each of these items. But more importantly, NTP doesn't tend to fail catastrophically. If you end up pushing out a bad package install, an unparsable configuration file, or a dead service, the system's on-board clock will generally back you up until you fix the problem.

Finally, try managing your CM tool with itself. Documentation on how to make a management system bootstrap itself, update itself, and ensure that it itself is running is standard for the big tools, making this an especially good task to tackle early on. This task is a little more dangerous than the others, since you have the potential to push a bad configuration out to the tool and cause it to wreak havoc, but if you keep the "simple" and "safe" ideals in mind while preparing the deployment, you shouldn't have any problems.

After completing these tasks you should have enough familiarity with your tool to move on to more complex configuration. Everybody has a different environment, but most configuration elements build on the same basic principles used to get through the above tasks. Focus on those first and you'll gain experience with your tool and make lots of headway with fully managing your infrastructure before you get to the hard parts.

Some Best Practices

The concept of configuration management has been around for many years now, and over that time many best practices have been identified. Although many of them are site-dependent, it is easy to identify a short list of practices that are easy to overlook but are very helpful at the beginning of a configuration management tool deployment:

Don't lose early momentum. After you've made some early progress with your deployment, keep making progress by spending 15 minutes a day making it better. Most configuration elements are not directly related to each other and there are always old files that can be cleaned up, so it should be easy to find small tasks that can be done while waiting for a meeting to start or for other people to get to the office in the morning.

Use version control. Don't use it in lieu of backups, but do keep your entire configuration repository in an SVN, Git, RCS, or any other revision control repository you feel comfortable with. This extra step gives you simple features such as commit comments and simple rollback, as well as complex features such as branching and cloning. Even if you just use the revision control system to keep comments from all committers, you will be a very large step ahead of a non-controlled repository.

Put ticket numbers in commit lines. When you make any change related to a ticket in a ticketing system, include the ticket number in the revision control system's commit comment. This makes it much easier to identify this set of changes in the future, whether to fix problems related to them or to make similar changes for a similar ticket.

Use dry-run modes. Most (perhaps all) of the modern tools have some sort of a dry-run mode, so use it. Before making big changes, make sure what you think will happen is really what will happen. You can use this same mode to generate nightly reports on noncompliance or to double check the work of a coworker. The dry-run mode is a simple but powerful tool.

Generate reports. You should know what your tool is doing (or not doing) in order to make the most effective use of it. Set up an email list, dashboard, Web page, or some other way of keeping tabs on how your system is performing. It will make it much easier to identify when some old unpatched machine is brought back onto the network after a year, as well as when conflicts arise and your tool can't make the changes you requested. It is also a great thing to show to auditors and managers to prove that your network is really in the state you claim it is in.

Simplify! Your environment is not the most unique out there. If you find yourself feeling like you need to rewrite a tool to make it fit your environment, it is likely that you are taking the wrong approach.

Conclusions

Now that you've read this article, the next step is to pick a tool and try it. The big four all have tutorials and examples to get you started, and the tasks listed earlier are great places to start in your own environment. As you progress, keep in mind that there are mailing lists and conferences full of people who have already done the same thing. Use those resources! The configuration management community is a generally friendly one with an abundance of helpful people.

After you've started your deployment, be careful not to lose steam and forget about your tool. It does take time and effort to keep your tool and its repository up-to-date, but you should find that that time and effort are much less than what you would spend doing the work by hand. Don't lose the battle after deployment!

The ideas covered in this article should be enough to get you started with a configuration management tool. Hopefully, you will quickly outgrow the topics covered here and move on to the next step in the processes: helping other people get their configuration under control too.

References

[1] Bcfg2: http://www.bcfg2.org/.

[2] Cfengine: http://www.cfengine.org/.

[3] Chef: http://www.opscode.com/chef/.

[4] Puppet: http://www.puppetlabs.com/.

[5] T. Delaet, W. Joosen, and B. Vanbrabant, "A Survey of Configuration Tools," in *Proceedings of LISA '10: 24th Large Installation System Administration Conference*, pp. 1–14.

[6] T.A. Limoncelli, *Time Management for System Administrators* (O'Reilly, 2005), p. 31.

[7] A. Tsalolikhin, "Summary, Configuration Management Summit," *;login:*, vol. 35, no. 5 (October 2010), pp. 104–105: http://www.usenix.org/publications/login/2010-10/openpdfs/ConfigMgt10reports.pdf.