Transformation at Gannett

http://www.chef.io

2/2016

Transformation at Gannett

Gannett is a leading media and marketing company with unparalleled local-to-national reach, which successfully connects consumers, communities and businesses. With the iconic USA TODAY, 92 strong local media organizations in 33 states and Guam, and with more than 160 local news brands online in the U.K., Gannett provides rich content through hundreds of outstanding affiliated digital, mobile and print products.

Each month more than 95 million unique visitors access content from USA TODAY and Gannett's local media organizations, putting the company squarely in the Top 10 U.S. news and information category. U.S. newspapers add an additional audience of 9 million readers every weekday.

You may already be familiar with Gannett's talk, delivered at re:Invent 2015, *USA Today Brings Shadow IT into the Light*. In that talk, Franklin Hanson, Senior Manager Platform as a Service Delivery, Erik Bursch, Vice President, Platform as a Service, and Chef's own George Miranda discussed the transformation at Gannett, which involved changes in culture, processes and tools. In this article, we talk a bit more with Frank and Erik about the changes at Gannett and Chef's role in standardizing Gannett's infrastructure.

The Beginning

Gannett's traditional deployment workflow was characterized by multiple handoffs and manual tests. Maintaining accurate, repeatable builds was difficult. There were many build failures and tests were often run in the wrong environments. Deployment and provisioning times could range from a few days to several weeks.

There were two operations teams, each in its own silo both physically, within different data centers, and organizationally. Neither team had access to the cloud or the development environments.

> "Prior to the Gannett Cloud Platform, deployments were difficult. We had too much bureaucracy, too many handoffs, too many tickets. We were depending on people who weren't a part of what we were trying to accomplish."
>
> — John Dietz, Platform Architect

On the development side, a type of "shadow IT" had emerged. Developers would spin up instances on Amazon's EC2-Classic and personal Heroku accounts and tie them to the production DNS. There was no sort of oversight.

Other complications were that every group used its own toolset, and there was no accountability to finance or security. No one knew how much an application actually cost. Security had no way to audit the stacks.

Gannett was ready for change. Developers wanted to deploy their applications quickly. Operations wanted a stable infrastructure where they could build and deploy in a repeatable way. Finance wanted to know the true cost of an application. Security wanted to view and audit all stacks and to be able to track changes.

Finally, the cloud had to become a sanctioned way to deploy applications and no longer be an unauthorized and ungoverned choice. Gannett saw that the cloud offered many advantages. Developers had access to standardized resources. It was easier to handle peaky traffic because of cloud's compute-on-demand model, and handoffs were minimized. In other words, it was time to bring shadow IT into the light.

The Transformation

Chef came into the picture when, about 18 months ago, Frank needed to rebuild a virtual private cloud (VPC) on Amazon Web Services (AWS) for a development environment that would mimic production. None of the tools he was already using were appropriate but he found that Chef worked well with the cloud and both Linux and Windows environments.

When a few other developers saw what Frank was doing with Chef, they became interested. It started a discussion about what it would mean to automate the infrastructure and how Chef could replace the current tool, which used a UI that only the operations teams could access and was closely tied to the on-site data centers.

Aside from the advantages of automation itself, Chef broke down barriers between development and operations and enabled them to work together. Treating infrastructure as code gave everyone a common language. Before Chef, although both groups were knowledgeable in their own areas, neither of them knew how to describe what they needed in terms the other group would understand.

A few precepts arose that were considered core to moving forward:

- The new VPC would use Chef and would use EC2-VPC rather than the older EC2-Classic. It would demonstrate that it was possible to build a development environment that perfectly matched production.
- For an application to move in to the VPC, it had to be provisioned and deployed with Chef.
- Security would be involved early on and would manage the mandatory controls for access to Chef and for maintaining system security standards.

Of course, adopting new processes is never easy, particularly when they challenge long-standing practices. Erik says, "We had to talk about what it meant for us to introduce Chef inside of our ecosystem. It was a true culture change for a lot of different groups."

Frank adds, "It was really the introduction of DevOps to these teams. To the devs you'd say they could have visibility and responsibility for an application. The pushback was that it was extra work that they didn't used to have to do. Previously, they'd complain that they didn't know what happened to their applications after they finished the dev cycle. Now, it was that they'd finished their dev cycle and why did they have to do all this extra work for testing, and learn to write Chef scripts and learn Linux or learn Windows?

"The pushback from operations was that they weren't interested in learning how to script or how to test their application code. Those who were comfortable scripting were largely from a world where they'd test in production and if it worked they kept it. Now they had to learn how to be a good developer and start locally and then move to dev and a testing phase and then a production release, and to use version numbers.

"I think this resistance is to be expected with such a drastic culture change, at least in a company that operated the way Gannett did for so long."

The initial project with Chef was within the mobile API group, which has many new applications all the time. They began by copying and pasting examples into roles and role cookbooks. Once they started to see success, Frank and Erik began to introduce Chef to teams that handled more core functionality, such as the content management team.

As people became more experienced, they moved beyond copy and paste and started to consider what changes they could make to actually benefit their applications. Growing expertise and confidence led the teams to consider not just how better to use Chef itself but how to improve the process as a whole. For example, they began to use continuous integration for their Chef code and to include integration tests.

Frank says, "We got real buy-in when people understood that Chef was code, the same as any other application. Before that, they thought it was just scripts to get into the environment. "

Training happened in a variety of ways. Erik says, "We pointed people to the **Learn Chef** site. There were also grassroots, side-by-side, little mini training sessions going on. We went through training services inside of Chef and brought Chef on site to get more of an audience but, in general, there was a lot of mutual support from what we call the Gannett community. It's a kind of open-source community, but inside Gannett. There were a lot of different developers or different teams making advances with Chef that caught on and got picked up by other teams. It really started to spread like wildfire after grassroots efforts began to pay off."

Cultural change is the most important and most challenging aspect of the Gannett transformation. Erik says, "We embed ourselves with teams to help them and to understand what their goals are and to see how well Chef can integrate with their practices. Because there isn't a lot of process in place in the first place, there's a lot of establishing how to do continuous integration (CI) with their existing toolsets rather than moving from what they currently have to Chef. We're doing a lot of legwork explaining what Jenkins or TeamCity is and how to do automated testing and how to publish back and forth to GitHub for tagging and releases—things that were previously done in manual handoffs."

> "Prior to using Chef, developers had very little knowledge of how to spin up a box. There was a big disconnect between development and coding versus infrastructure. With the introduction of Chef, we've empowered developers on how to spin up a box, how to deploy their code. It helps with timelines and it helps by giving developers a better understanding of what sort of infrastructure their applications need."
>
> — Alon Motro, Manager Content Platform Teams

Chef and DevOps are also affecting the structure of the teams. For example, Erik began to include members of the QA team with the development teams so that they could incorporate QA requirements into their code from the beginning rather than have the application reviewed once it was completed. He and Frank have gotten input from their security teams, who are learning to frame their requirements in terms of Serverspec. People from operations and architecture have also joined development teams. They are there to help design the environments that will support the applications and to write cookbooks that will provision those environments.

The Workflow for Cookbooks

Gannett has a private Supermarket where it maintains a library of cookbooks developed by the Gannett community, and people try to use those cookbooks as much as possible. They also use cookbooks from the Chef open-source community. Role cookbooks, which have no actual Chef code in them aside from includes and attributes, are managed by individual teams. Utility cookbooks, which are used by multiple teams, are managed by Frank's team.

Role cookbooks have one GitHub repo per team. All of the applications that use that role cookbook are also in the repo. Utility cookbooks have their own repo.

All cookbooks, whether internally developed or from the open-source community, follow the same release process. They're first tested locally with Test Kitchen and then with ChefSpec, Foodcritc and RuboCop.

Once a cookbook passes its local tests, a pull request (PR) is generated in GitHub against the repo the cookbook belongs to. The PR is automatically tested against Test Kitchen and EC2, using a Jenkins server to run the same standardized testing. If the PR passes, it's marked with a check box and one of the senior developers does a sanity check and makes sure that the change follows best practices.

Then, the change is merged to master and goes through the same suite of tests again. When those tests pass, the cookbook is uploaded to Supermarket. If it fails Supermarket it generally means that a dependency is missing. Otherwise, it goes from Supermarket to the Chef server. Gannett uses one Chef server for its data centers and the cloud, and it's located in production.

All master branches are considered to be in production. Cookbooks are continuously delivered and versioning controls what goes out.

The Results

Over the past 18 months Gannett has made significant advances. Their deployments are quicker and more reliable. Application provisioning and deployment, which once could take weeks, now takes minutes.

All new applications are deployed to the cloud with Chef. Those applications are deployed to all environments the same way that they're deployed to production. Also, testing occurs in each environment, so that the deployments are reliable.

All infrastructure is treated as code, which greatly increases visibility into any changes that occur. Development, operations, security and finance all benefit.

> "Having visibility into the entire pipeline in terms of how the servers are configured, how the apps are configured, has allowed us to optimize not only the deployment platform but the end-user experience. We can turn on features that are really specific to one application without having to worry about breaking other platforms or breaking other teams, and we can speed up everything."
>
> — Jay Merrifield, Principal Developer

Erik says, "Chef has increased the effectiveness and the speed of our development cycle. We've been able to use the economies of scale that Chef gives us to move more quickly across the board. If development team X is using a new technology, they write their Chef cookbook and push it out. Development team Y comes along and chooses the same technology. That cookbook is already in place and they're able to run it."

Currently, about 30% of Gannett's technology organization is using Chef but Gannett has aggressive plans for 100% adoption in 2016. Both Erik and Frank are confident that they can meet this goal. Strong grassroots support, impressive results, and a culture that is committed to going faster are some of the reasons for this confidence, but there is another component as well, which is, in addition to grassroots support, support from above.

Erik says, "Jamshid Khazenie is our CTO and his buy-in to our automation strategy, which includes Chef, has resonated throughout our technology organization. He understands where Chef sits in our ecosystem and has reiterated that in meetings across the technology organization. That is a huge benefit to us moving forward at the speed we have. How were we able to make these gains in a short amount of time? The fact is we have a leadership directing the whole organization and saying that this is the way we will do things. We will follow this automated method that we've established in our Gannett cloud platform."

Erik also cited Chef's role in Gannett's success. "I'll put out a nod to you guys. You've been a good partner and acted as a sounding board on a lot of items. I use "partner" rather than "vendor" as a key difference in my book. You've worked step-by-step with us on this process. It's been a really good relationship."

Of course, along with 100% adoption of their automation strategy, Erik and Frank have plans for the future. While deployment rates are dramatically faster, they want to shorten them even more. Another goal is to work with their Payment Card Industry (PCI) department to begin automating those requirements. They're also investigating how to move away from Serverspec toward Chef's compliance language, InSpec.

Call to Action

When asked what advice they would give to others who are deciding whether to begin their own journey, both Erik and Frank shared some of the things they've learned.

> "You have to jump in somewhere. You can't wait for that time when you have two weeks free on your schedule."
>
> — Erik Bursch, Vice President, Platform as a Service

Frank says, "Don't try to do everything perfectly the first time. If you wrap yourself up in the complexities that come with the Chef environment from the first run, you'll never get off the ground.

"When we started we looked at all kinds of issues and the biggest pushback was always, 'Oh, it doesn't cover this, it doesn't cover that.' It doesn't matter. Pick something, even if it's something small, and do it. Take something small and iterate on it and don't try to be perfect or pretend to be perfect. The most important thing is make sure you have a process defined where you can improve."

Erik adds, "Around here, we say deal with the failure. We probably failed 100 times before we got the right pieces in place. We spun it up. If it worked, great, if it didn't we'd continue to tweak it.

Frank sums it up. "The big thing is pick something and do it. Take action. Before I came to Gannett, process improvement was something I didn't have time for. We'd keep saying we'd get to it and were miserable forever. Bite the bullet and reap the rewards."


CHEF