Getting Started with DevOps Automation

Cisco eBook

by Scott Sanchez, Director of Strategy

Table of Contents

1 Introduction

In the time it took you to download and open this eBook, you could have spun up and configured hundreds of virtual machines in a public or private cloud, deployed your application, database, middleware, messaging, load balancers, storage, and other components to those configured servers, linked them all together, and had your app live in production – all with a single click.

This isn't the future we're describing, it's right now, and thousands of development teams around the world are doing it. At a high level, we're describing automation that cloud and APIs enable, and at a low level, we're describing the outcome of adopting a DevOps mentality and the associated tools. This eBook will look at the basics of DevOps, automation, and configuration management to get you started down the right path.

2 Background

Most companies today are still relying on some level of manual deployment and configuration of their apps. Virtualization and cloud users have the benefit of being able to take VM images or snapshots of their installed and configured apps, and re-deploying those as needed. This works fine for static or long-running environments, but if you're constantly pushing code into your environment (like you should be), or are dealing with web-scale applications that have a ton of moving pieces, the "gold image" model quickly gets unmanageable.

You want to migrate from one availability zone to another? Dev to Test? Test to Prod? Prod to DR? You may be tempted to look at something like a live migration technology from your hypervisor. But the "DevOps" way is to have your configuration stored, and just deploy a fresh environment with a new deployment target, then move your data, and turn off the old environment. Gold images are replaced with gold configurations, and your named servers, that you probably keep track of in a spreadsheet somewhere, are replaced with raw resources that you can easily replace.

Over the past decade, and especially in the past few years, the tools and integrations that enable this automated model have come a really long way. What started with people writing basic scripts to automate patching across groups of servers has turned into powerful platforms to abstract the complexities of the underlying infrastructure and turn it into amazing configuration and automation interfaces for developers.

3 Getting Started

All automation and configuration management platforms share common concepts:

- Your programming language, your cloud, your web server, your continuous integration platform, your testing framework, your load balancers, etc.).
- A way to document and create a configuration (often called a "recipe")
- A way to link a number of configurations into a complex automation routine (often called a "cookbook")
- An execution and workflow engine to run your configurations and automation routines
- A management and visibility interface to keep track of what's happening

Ultimately if you can do something in one platform, like deploy and configure a cloud server, or push code through your continuous integration and testing framework to production, it's highly likely you can do it in another. You'll find significant differences in the scripting syntax, user experience, and available integrations between the platforms. One thing is for sure; all the platforms we talk about below are powerful, widely used, extremely flexible, and have large Open Source communities behind them. You'll be hard pressed to find a use case that they can't support.

4 Selecting a Platform

At the time of this writing, Chef and Puppet were the most widely adopted configuration management and automation platforms, with Ansible and SaltStack as the new kids on the block. These four platforms are the most widely used by Cisco customers, but by no means represent the entire scope of possibility. A quick Google search for DevOps or configuration management platforms will turn up a mind-blowing number of options.

Each platform we cover below has a list of features and benefits that overlap widely, and often the choice for which fits your needs best will come down to where your team is coming from and personal preferences/skill sets. Which one is right for you? That's hard to say; it's a lot like Coke and Pepsi, both are delicious and nearly identical in 'features,' but people still have a strong preference for one over the other.

Chef®

We've noticed that developers tend to initially gravitate towards Chef, with it's Ruby-based scripting syntax for writing configuration and automation routines. It has a huge number of recipes and cookbooks that have been contributed by the Open Source Chef community over the years, and popular recipes (e.g. for using the OpenStack® APIs) have very large communities of contributors keeping them up to date and well tested.

Puppet®

If you're more comfortable in a Linux terminal than you are writing Ruby code, or would self-identify as a sysadmin more than a developer, you should probably take a good look at Puppet. It uses a Python-based syntax that long-time Linux admins will feel right at home with. Like Chef, you'll find large, well maintained libraries of configurations and integrations where Puppet can plug right in.

Ansible® & Salt®

These new arrivals on the scene in the past couple of years have taken a slightly different approach than Chef and Puppet did. While you'll still find large libraries of well-tested integration points and pre-built configurations, you'll also find nicely polished user interfaces and management consoles. Taking the approach that developers really just want to write code, Ansible and Salt have invested heavily in the overall user experience. That doesn't mean that you sacrifice power or extensibility for experience, it's just a different approach than Chef and Puppet started with. At the time of this writing, we're seeing a growing number of Cisco customers using or evaluating Ansible and Salt – many using them in combination with Chef or Puppet to give them the best of both worlds.

5 Mindset

Picking, learning, and implementing the tools are just one part of the equation—the bigger challenge will be adopting the right mindset. In a DevOps world, the lines between which team "owns" the servers and which team "owns" the apps are very blurred. These teams often become one, working together to write the configurations, testing processes, deployment routines, and other components of the automation.

You stop thinking about cloud servers as named devices, and start thinking of them as raw resources that you configure and deploy to, automatically. If a particular cloud virtual machine starts behaving badly, you don't log in and fix it. You've written automation routines in your tools that are monitoring its health, and they spin up a fresh server, automatically configure it, deploy your apps to it, connect it to your environment, and kill off the "sick" server.

It should come as no surprise that "hybrid" cloud isn't a cloud at all, it's a deployment model where you split pieces of your app between multiple public or private cloud availability zones. When you're naming your servers and doing everything manually, hybrid cloud seems impossible. But when you adopt a DevOps and automation mindset, treating your cloud resources as just that- "resources," hybrid cloud simply becomes an exercise in changing your deployment target.

6 Hybrid Cloud & DevOps

The most common hybrid use case that we see at Cisco is a customer coming to us and moving a static (base) workload from AWS or another public cloud to a Cisco OpenStack® Private Cloud for better efficiency, economics, and control. Their app gets split when it's (automatically) deployed – with the variable workload portion going to the public cloud, and the base workload going to their Cisco OpenStack® Private Cloud. Both environments have auto-scaling, courtesy of their automation platform, and everything is just a raw cloud resource that gets configured and used, until it's not needed anymore or replaced with something new to support updated application code.

Hybrid isn't about an infrastructure configuration, it's about how you deploy, manage, and scale your application. Want to move all or part of your app to a different availability zone? Just change the target in your configuration management tool; that's what hybrid really means. Cisco OpenStack Private Cloud customers like Tapjoy are great examples of massive scale, highly automated applications that are split between public and private cloud. You can find the Tapjoy case study on the Cisco website for more information.

7 Conclusion

Moving to a model where you treat your cloud resources are “raw,” and your administration and configuration as “hands off” is a journey. Here’s a quick list of things you should do to get started, in priority order:

1. Decide who from your existing ops/infrastructure/sysadmin/dev teams will make up your new DevOps team.

2. Get crystal clear on the roles your remaining infrastructure team, new DevOps team, and focused dev teams will have in the new model (who does what). Throw out everything you know about how you manage infrastructure and deploy/manage apps today. Here’s your chance to do it right.

3. Identify the human and technology touch points between the new teams. Where are the handoffs between them? Who relies on whom? For what? How loosely or tightly should those touch points be defined to fit your culture (and apps)?

4. Take stock of the existing technologies you’ll need to configure and automate, and compare that to available pre-built integration points in the various configuration management and automation platforms.

5. Narrow down your tool search by comparing how well it fits in to your technology stack(s) with how well it fits with your people.

6. Get training on the tools you pick. Every platform we mentioned here has a huge amount of free and paid training material available both online and in-person around the world.

7. Don't bite off more than you can chew. Don't try to convert all your apps to be automated. Focus on the new apps. Focus on the cloudy apps.

8. Even with your new apps, don't try to automate everything. Start with deploying and configuring thae cloud virtual machines. Add things like load balancers and virtual networks. Add a testing framework, CI, auto-scaling and auto-resiliency, and the hundred other things that get you from DevOps to "NoOps" later.

9. Don't feel like you're alone. Thousands of development teams have made this journey before you. Configuration management and automation may feel like the future to you, but it's very likely that every major website you visited today is already using it for a good portion of their stack. The sheer amount of blog posts, how-tos, and mailing lists dedicated to DevOps is proof. Cisco often helps our new private cloud customers navigate the DevOps waters.

10. Contribute back. The reason most of these platforms are so powerful and do so much is because of the thriving communities of people, just like you, around the world that share their configurations and integrations back with the ecosystem. If you improve a configuration or automation routine you found online, share it!

Smart companies like Tapjoy are using Cisco OpenStack® Private Cloud for a private cloud with powerful DevOps automation tooling to power their fast growing applications.

To learn more about Cisco OpenStack Private Cloud, visit our website. To review tutorials about how to use many of the automation tools mentioned in this ebook, visit our Community page.