DevOps and the Cloud: Chef and Microsoft Azure

http://www.chef.io

5/2015

DevOps and the Cloud: Chef and Microsoft Azure

Introduction

This paper is an introduction to how using DevOps patterns with cloud resources can decrease time to market and reduce costs. The paper first discusses DevOps, which is a cultural and technical movement. With DevOps, companies can deliver value to their customers quickly and safely.

The second section presents common problems many enterprises encounter and the solutions DevOps offers for these problems. Some statistics demonstrate how and why DevOps makes financial sense.

Finally, there are some examples of how to use automation to provision and manage resources in the cloud. The automation platform is Chef and the cloud is Microsoft Azure, but the examples illustrate general principles that apply to a variety of situations.

What Is DevOps?

DevOps is a cultural and technical movement that focuses on building and operating high-velocity organizations. DevOps began with web innovators who wanted to take maximum advantage of the cloud, which made it possible to allocate resources quickly and inexpensively. Traditional IT practices were not designed for the flexibility and speed the cloud offers.

DevOps cultural values

DevOps advocates cultural values that encourage communication and cooperation. The term "DevOps" is a combination of "Development" and "Operations" and signifies a close relationship between those two areas of expertise. In many traditional enterprises these groups are separate. The developers create applications and the operations teams deploy them to an infrastructure they manage. This separation is an example of a *silo*. Silos exist when organizations have strict divisions of responsibility. Often, communication between groups only occurs through a formal mechanism, such as a ticketing system.

While it might seem more efficient to have different groups, each with a well-defined specialty, silos require handoffs from one group to another. Handoffs introduce significant delays and inaccuracies. For example, in companies with silos, it often takes multiple groups to configure a full stack. One group writes the specifications, a second group configures the VM, that group hands the VM off to a third group to install the database, and so on. Each handoff means another delay.

Handoffs also introduce inconsistencies and inaccuracies. In *Implementing Lean Software Development: From Concept to Cash*, Mary and Tom Poppendieck conservatively estimate that each handoff leaves behind approximately 50% of the knowledge that's meant to be transferred. This means that there is:

- 25% of the knowledge left after 2 handoffs.
- 12% of the knowledge left after 3 handoffs.
- 6% of the knowledge left after 4 handoffs.
- 3% of the knowledge left after 5 handoffs.

The costs of handoffs negatively offset the benefits of the cloud's ability to flexibly deliver compute resources. In fact, safely reducing the number of handoffs is one of the primary benefits of the DevOps workflow.

Another problem is that it's common for each silo to have its own procedures and tools. Lack of a common approach contributes to the problems of long build times and errors.

In contrast, companies that have adopted DevOps often use small teams that work together to create applications and provision and manage the infrastructure that these applications use. In his article, "**How Etsy Makes DevOps Work**," John Dix interviewed Michael Rembetsy, VP of Technical Operations at Etsy, who explained how DevOps evolved at his company. He gave an example of how teams work.

"For example, if we have a search team, we don't have a dedicated operations person who only works on search. We have a designated person who will show up for their meetings, will be involved in the development of a new feature that's launching. They will be injecting themselves into everything the engineering team will do as early as possible in order to bring the mindset of, "Hey, what happens if that fails to this third-party provider? Oh, yeah. Well, that's going to throw an exception. Oh, OK. Are we capturing it? Are we displaying a friendly error for an end user to see? Etc.""

Working together on all aspects of a feature eliminates handoffs and problems that come from poor communication and silos. Consensus is easier to achieve and design decisions, whether for the application or the infrastructure are known by everyone. Quick decisions translate into companies that move at higher velocity.

DevOps technical values

Companies that practice DevOps have workflows designed for high velocity. Software moves quickly from development through testing, staging and then to production. Environments, often located in the cloud, are quickly provisioned and configured and are consistent with each other. Software is promoted from one phase of the pipeline to another either automatically or with a straightforward manual step.

To avoid lengthy development times and difficult releases, companies release software iteratively. They begin with a minimum viable product, gather customer feedback, improve the product, and release the software again. The product evolves over multiple cycles. Because each new version of the product has only a few changes, each iteration is easier to debug.

There are a variety of technologies that enable a DevOps workflow but the primary one is automation.

AUTOMATION

Automation underlies all the patterns and practices that constitute DevOps. An automation platform gives you the ability to describe your *infrastructure as code*. When infrastructure is code, you can:

- Eliminate error-prone, time-consuming manual tasks.
- Standardize development, test and production environments.
- Build automated release pipelines.
- Improve cooperation between development and operations.

You can treat your infrastructure code just as you would your application code. The code is versionable, testable and repeatable. You can (and should) use the same deployment pipeline for your infrastructure as you do for your applications.

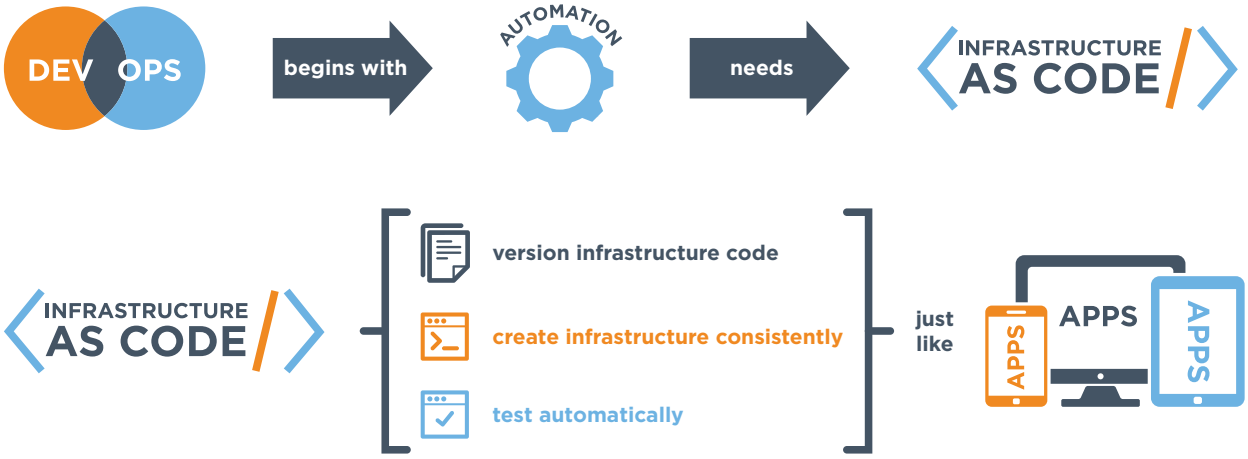
> "The tools we use reinforce the behavior; the behavior reinforces the tool, thus if you want to change your behavior, change your tools."
>
> - Adam Jacob, CTO, Chef

Because automation turns your infrastructure into code, you can use automated tests. You can build compliance and security tests into the deployment pipeline, thus catching problems earlier rather than later. Instead of making changes whose effects are unknown to your production environment, you can ensure that new configurations are safe and stable.

By its nature, automation discourages silos. To take full advantage of the cloud, you use automation to quickly spin up resources and configure the entire stack. Scale up, down or horizontally by running a program that provisions and configures your network in minutes, not weeks. Because the process is automated, you know the results will be consistent from one run to the next. Everyone on the team uses the same process to spin up a stack. There are none of the silos, handoffs or conflicting procedures that cause delays and errors.

Another advantage to automation is that infrastructure code is expressed as human-readable text files. DevOps encourages transparency. Describing your infrastructure as code means that is accessible and readable to everyone on the team. In addition, you can keep these files in a source control system, where they are versioned and kept safe. All the advantages of using a source control system with your application code apply equally to your infrastructure code. Examining differences between versions of your configuration recipes shows *exactly* what has changed since the previous known stable state of the system. Such visibility is critically important.

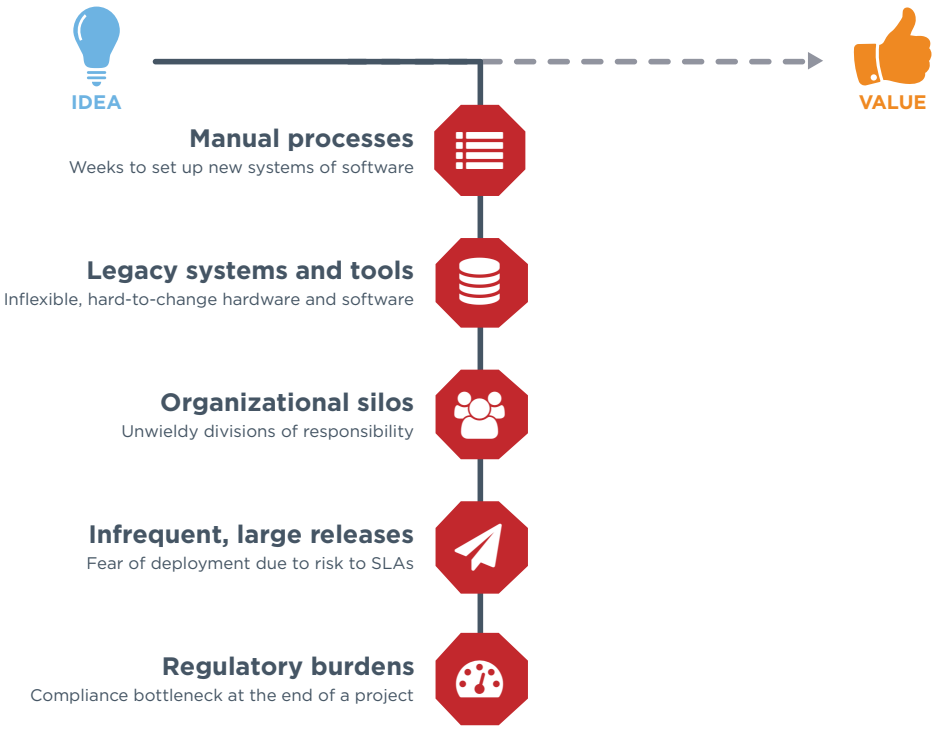
The following illustration summarizes the relationship between automation and DevOps.



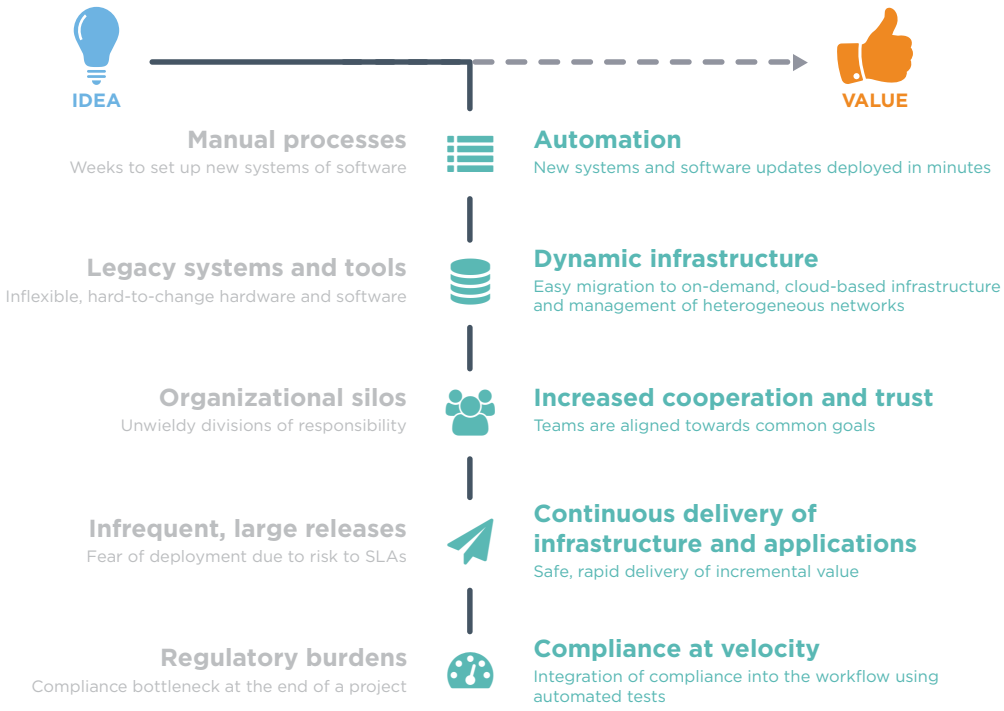
The illustration stresses the relationship between DevOps and automation. Everything depends on infrastructure as code.

DevOps Solutions

Large enterprises have many challenges that can be addressed with DevOps. The following figure illustrates the most pervasive problems that result from traditional IT practices.



The next figure shows the alternatives offered with DevOps.



Moving away from traditional processes to a DevOps workflow has dramatic effects on a business. Dr. Nicole Forsgren gave a talk entitled **DevOps and the Bottom Line** at DevOps Enterprise Summit 2014, where she discussed the results of research she has done on the consequences of practicing DevOps. She states that companies that use DevOps have greater agility and reliability as well as better growth and profitability.

Greater agility

Companies that practice DevOps have 30 times faster deployments and 8,000 times faster lead times than their peers. (*Lead time* is the total time, from start to finish, that it takes to develop a product or service and deliver it to customers.) Two of the reasons for greater agility are:

- Infrastructure, runtime environments and applications are delivered using a unified process.
- The number of handoffs and service tickets is greatly reduced.

Greater reliability

Companies that practice DevOps have 2 times the change success rate and 12 times faster mean time to recover than their peers. Some of the reasons for greater reliability are:

- Integration of compliance and security into the development process removes blockers.
- Testing catches problems prior to deployment.
- Shipping frequency, smaller batch sizes.
- Development environments closely resemble production environments.

Better Growth and Profitability

Companies that practice DevOps are 2 times more likely to exceed profitability, market share and product goals. They exhibit a 50% market cap growth over 3 years.

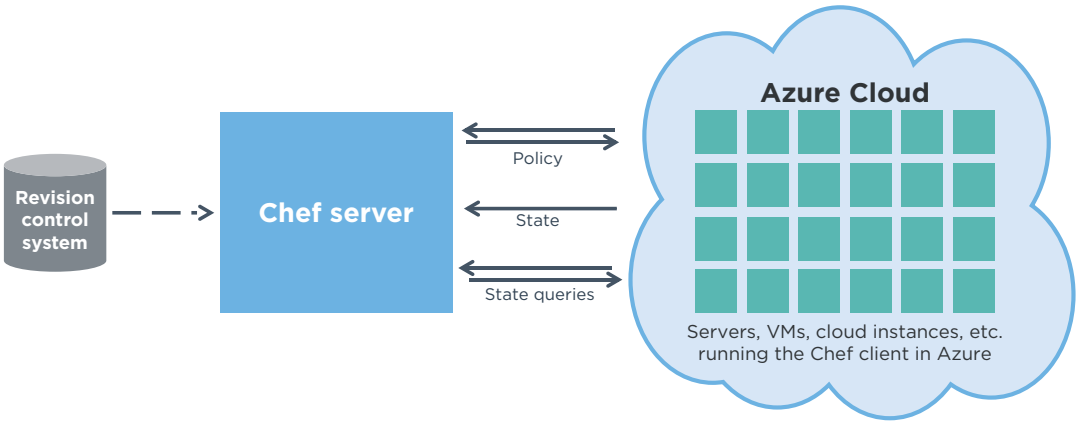
What is Chef?

Chef is a dynamic, policy-based automation platform. Chef uses instructions, called recipes, to configure nodes. A node is an element of your infrastructure, such as a server. It can be physical, virtual, in the cloud, or even a container instance. A node can run Windows, AIX, or almost any flavor of Linux.

Recipes reside on the Chef server. They are developed on local workstations and stored in source control.

The Chef client is installed on every node and periodically polls the Chef server for the latest recipes. The Chef client runs the recipes and brings the node to the correct state, over time. Chef clients also notify the server of their state and can query for the state of other nodes. Because most of the work happens on the nodes, the Chef server never becomes a bottleneck and you can scale up to manage infrastructures of any size and complexity.

Here is a simple diagram that shows how Chef works.



WHAT IS A RECIPE?

When you automate your infrastructure, you describe all the tasks you want to perform as executable code. Here is an example taken from a Chef recipe.

```
azure_storage_container 'my-node' do
  storage_account 'my-account'
  access_key azure['access_key']
  action :create
end
```

This code creates a storage container on Azure that is named **my-node**. Chef only takes action when the desired condition doesn't exist. In this case, if the container is already there, Chef does nothing.

Chef Integration with Microsoft Azure

Chef is tightly integrated with Azure. There are three options for creating a virtual machine (VM) and bootstrapping it with the Chef client.

- **Use the Azure Management Portal**. Creating a VM that runs the Chef client takes only a few clicks if you use the Azure Management Portal. All you need is the Chef configuration file (client.rb), a .pem file with a validator key and a run list that contains the Chef recipes you'll use to configure the new instance.
- **Use the Set-VMExtension cmdlet**. Use the Set-VMExtension cmdlet to install the Chef client on a Windows machine. The cmdlet is distributed as part of the Azure SDK.
- **Use the knife command line**. Use Chef's knife-azure plugin to create and manage instances from the knife command line. For example, you can create instances bootstrapped with Chef, enumerate instances and images, and delete images from Azure and the Chef server.

Just as the Chef client image is available on Azure, so is the Chef server. You can select a VM that has the Chef server already installed from the collection of VM images on **Azure Marketplace**.

Chef fully supports Microsoft's PowerShell extension, Desired State Configuration (DSC). DSC provides a layer of abstraction over Windows resources such as Active Directory, SQL Server and IIS, and incorporates Microsoft's knowledge of Windows components. Any DSC resource can be included in a Chef recipe.

Other Chef Tools

Chef has other tools you can use with Azure. The **azure-cookbook** includes resources for storage management and SQL Azure. (A cookbook is a collection of recipes.) The chef-provisioning-azure feature lets you declare your infrastructure in a Chef recipe and create it on Azure. You can use provisioning to easily create and orchestrate multi-tiered applications.

Examples of using Chef and Azure

In this section, you'll see how you can use Chef with Azure to spin up resources quickly and reliably. Although these examples are simple, you can use the same principles to automate large, complex infrastructures. For example, MSN uses Chef to manage thousands of nodes on Azure. For an explanation of how MSN uses Chef and Azure, see the talk **Microsoft Presents: Chef in Action on Azure**, given at ChefConf 2015.

ENABLING AN APACHE SITE

The first example shows how to enable an Apache site. Azure hosts both Windows and Linux machines.

```
package "httpd" do
  action :install
end

apache_site site_name do
  port site_data["port"]
  notifies :restart, "service[httpd]"
end

include_recipe "apache::fwrules"

#Service Restarting/Start
service "httpd" do
  action  [ :enable, :start ]
end
```

This recipe installs `httpd`. It also passes in some data elements for the site port and the site name. These values can be overridden in an attributes file. In other words, Chef recipes separate the recipe's logic from its data. The recipe also calls another recipe to initiate the appropriate firewall rules. Finally, the recipe enables and starts the Apache service.

DYNAMIC SERVICE DISCOVERY

Often, you have a tiered application and you need to dynamically discover what machines and services you're using. For example, you might launch many instances inside of Azure as well as a database service and other components, all of them depending on each other.

One way to solve this problem is to use recipes that take advantage of Chef's powerful search capabilities. Each node provides data about itself to the Chef server, where it is stored and indexed. You can query this data either with the command line or with recipes. Here is an example of a recipe that configures a HAProxy server.

```
pool_members=search("node", "role:webserver")

template "/etc/haproxy/haproxy.cfg" do
  source "haproxy.cfg.erb"
  variables (:servers => pool_members.uniq)
  notifies :restart, "service[haproxy]"
end
```

The recipe first queries for all of the nodes that have the role of "webserver." With the cloud, infrastructures can change all the time to suit current needs. Each time the recipe runs (the default is every 30 minutes) it discovers what instances are now available and automatically configures HAProxy (with haproxy.cfg.erb) so that the correct nodes are in its load balancer pool.

LAUNCHING MULTIPLE INSTANCES

You often want to launch many instances on Azure at once. Chef provisioning allows you to do this. It gives you programmatic access to instances and allows you to create complete stacks. Here is a simple example.

```
#provision 3 webservers synchronously
machine_batch do
  (0..2).each do |i|
    machine "myweb-web#{i}" do
      role 'webserver'
    end
  end
end
#provision 1 load balancer
machine "myweb-lb" do
  role 'LoadBalancer'
end
```

The Chef provisioning `machine_batch` resource creates instances in parallel. In this recipe, it creates three webservers synchronously. Creating the webservers at the same time rather than sequentially is possible because none of the webservers need to discover each other. (For a more detailed explanation of using Chef provisioning on Azure, see the **Chef Provisioning to Azure** demo on Channel 9.)

The recipe then creates another instance that is a load balancer. It will be dynamically configured to load balance the three webservers.

Key Points

This paper discusses the intersection of DevOps, automation and the cloud.

- DevOps is a cultural and technical movement.
- With DevOps, companies can deliver value to their customers quickly and safely.
- DevOps cultural values emphasize communication and cooperation.
- DevOps discourages handoffs and silos.
- Companies that have adopted DevOps use small teams that work together to create applications and provision and manage the infrastructure that these applications use.
- Automation is the underlying technology for DevOps.
- An automation platform gives you the ability to describe your infrastructure as code.
- When infrastructure is code, it is versionable, repeatable and testable.
- Companies that use DevOps have greater agility, reliability and better profitability than companies that use more traditional processes.
- Chef is a dynamic, policy-based automation platform.
- Chef and Azure are tightly integrated.
- Chef fully supports Microsoft PowerShell DSC.
- Together, Chef and Azure can support large, complex infrastructures such as MSN.

Resources

Here is a list of the resources mentioned in this paper along with some others you might find helpful.

Chef cookbooks

azure-cookbook at **https://github.com/chef-partners/azure-cookbook**

Chef tools

Chef Development Kit (Chef DK) at **https://downloads.chef.io/chef-dk/**

DevOps and Lean

DevOps and the Bottom Line at **https://www.youtube.com/watch?v=V6DrGBg-w40**

Dix, John. "How Etsy makes DevOps work" at **http://www.networkworld.com/article/2886672/software/how-etsy-makes-devops-work.html** *Network World* February 19, 2015.

Humble, Jez, et al. *Lean Enterprise*. Sebastopol: O'Reilly, 2015. Print.

Poppendieck, M. and Poppendieck, T. *Implementing Lean Software Development: From Concept to Cash*. Boston: Addison-Wesley, 2006. Print.

Level Up the Change in Your Enterprise—Nordstrom at **https://www.youtube.com/watch?v=Ot5H2KfWAxI**

The Lean Enterprise at **https://www.chef.io/webinars/**

Learning Chef

Learn Chef web site at **http://azure.microsoft.com/en-us/downloads/**

Chef documentation at **https://docs.chef.io/**

Microsoft Azure

Chef Server 12 on Azure Marketplace at **http://azure.microsoft.com/en-us/marketplace/partners/chef-software/chef-server-chefbyol/**

Microsoft Azure web site at **http://azure.microsoft.com/en-us/**

Microsoft Azure SDK at **http://azure.microsoft.com/en-us/downloads/**

Install and Configure the Azure Cross-Platform Command Line at **http://azure.microsoft.com/en-us/documentation/articles/xplat-cli/**

Microsoft Azure and Chef

Automating Azure Virtual Machine Deployment with Chef at **http://azure.microsoft.com/en-us/documentation/articles/virtual-machines-automation-with-chef/**

Azure+Chef=Awesome, Episodes #1, #2 and #3 at **https://www.chef.io/webinars/**

Chef Provisioning to Azure Demo at **http://channel9.msdn.com/Shows/Edge/Chef-Provisioning-to-Azure-Demo**

Managing Your Systems on Microsoft Azure with Chef at **http://www.microsoftvirtualacademy.com/training-courses/managing-your-systems-on-microsoft-azure-with-chef**

Microsoft Presents: Chef in Action on Azure at **https://www.youtube.com/watch?v=nFV1AyAyVfw**

OpenSource Automation on Azure using Chef at **https://www.chef.io/webinars/**

Windows PowerShell Desired State Configuration

Seattle Chef Meetup 7/15/14: Chef and Microsoft's Desired State Configuration (DSC) at **https://www.youtube.com/watch?v=mXaAIawzNic**

Windows PowerShell Desired State Configuration Overview at **https://technet.microsoft.com/en-us/library/dn249912.aspx**

Windows PowerShell Unplugged with Jeffrey Snover at **https://www.youtube.com/watch?v=t4a2l0ryU5Y**