DevOps and the Cloud: Chef and Amazon Web Services

Introduction

This paper is an introduction to how using DevOps patterns with the Amazon Web Services (AWS) 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.

The third section contains some examples of how to use automation, the technology that is fundamental to a DevOps workflow, to provision and manage resources. The examples use Chef for automation and AWS as the cloud provider. The final section, "Tying it All Together," is an overview of a scenario whose code is available on GitHub, that you can use to get started exploring how Chef integrates with AWS and to experiment with a DevOps workflow.

Finally, there is a case study that shows how Gannett used AWS and Chef to transform their culture, processes and tools.

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 needed to operate at massive speed and scale. It came into existence at the same time as cloud technologies, which make it possible to allocate resources quickly and inexpensively. Traditional IT practices were not designed for the flexibility and speed the cloud offers, and, as a result, development and deployment practices needed to be reimagined. This new way of working is what we call DevOps.

"The term "DevOps" is a combination of "Development" and "Operations" and signifies a close relationship between those two areas of expertise."

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. Often, development and operations are in separate silos. 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.

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 to 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. Rembetsy gave an example of how teams work:

"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 everyone understands design decisions, whether they are for the application or the infrastructure. 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 on the AWS 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 that use DevOps 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.

Automation for DevOps

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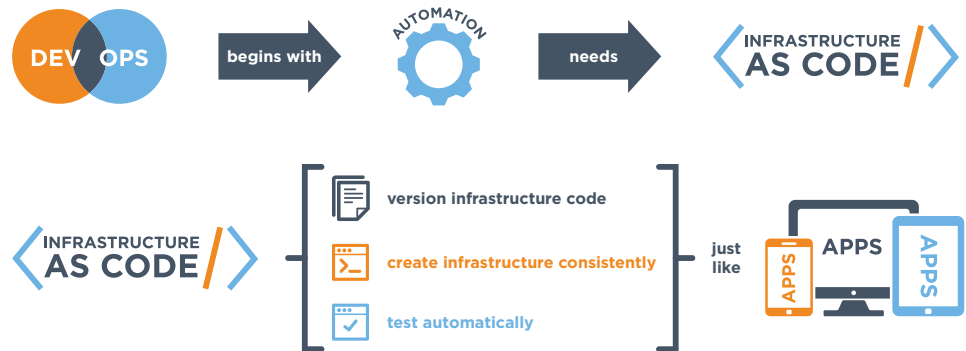
DevOps Solutions

Large enterprises have many challenges that can be addressed with DevOps. The following figure shows the most pervasive obstacles that traditional IT practices create when moving from an idea to realized business value.

The next figure shows how cloud-based DevOps practices and tools eliminate these obstacles.

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
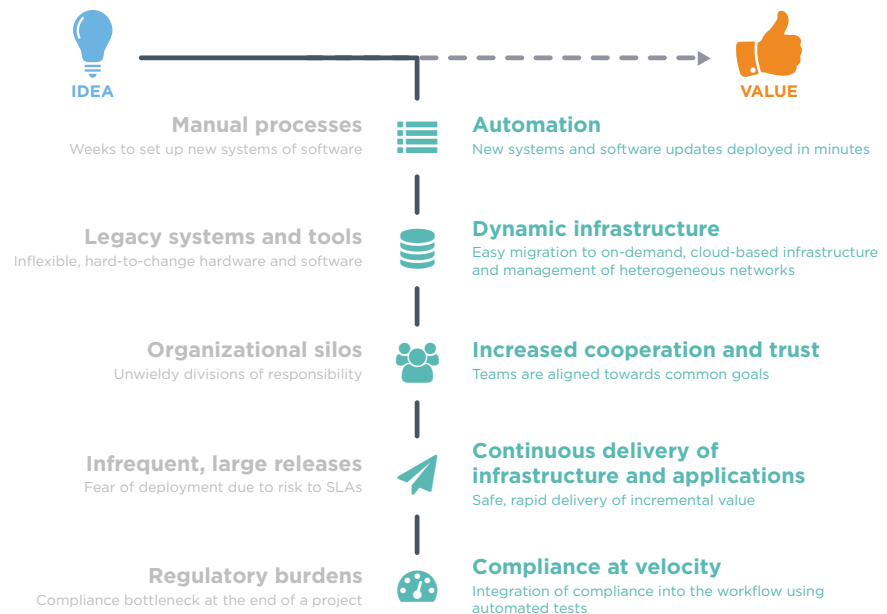
she discussed the results of research she has done on the consequences of practicing DevOps. Her research shows that companies that use DevOps have greater agility and reliability as well as better growth and profitability. Here is a summary of these results:

DevOps benefit 1: Improved 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.

DevOps benefit 2: Improved reliability
Companies that practice DevOps have twice the change success rate and 12 times faster mean time-to-recover than peers that do not use DevOps. 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 improves with smaller batch sizes.

- Development environments can closely resemble production environments. Discrepancies between environments is a common reason for software that works in development but fails in production.

DevOps benefit 3: Better growth and profitability
Companies that practice DevOps are twice as likely to exceed profitability, market share and product goals. They exhibit a 50% market cap growth over 3 years.

DevOps Solutions

Chef is a dynamic, policy-based automation platform. Any machine managed by Chef is called a node. It can be physical, virtual, in the cloud, or even a container instance.

A Chef resource describes some piece of environment, such as a file, a template, or a package. A Chef recipe is a file that groups related resources, such as everything you need to configure a web server, database server, or a load balancer. Recipes are developed on local workstations and stored in version control system. Collections of recipes are called cookbooks.

Chef uses a client/server architecture to manage the nodes in your network. 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 environments of any size and complexity.

When the Chef client runs, it only updates the node if a change is necessary. For example, if you have a recipe that installs a package on a server, Chef only performs the action if the package isn't already installed. If it's already present, Chef does nothing. This approach, called test and repair, coupled with periodic runs of the Chef client has these benefits:

- It eliminates configuration drift over time

- It handles errors, and network failures by dynamically changing network topology over time

- It handles complex configuration interdependencies among nodes.

Here is a simple diagram that shows how Chef works.
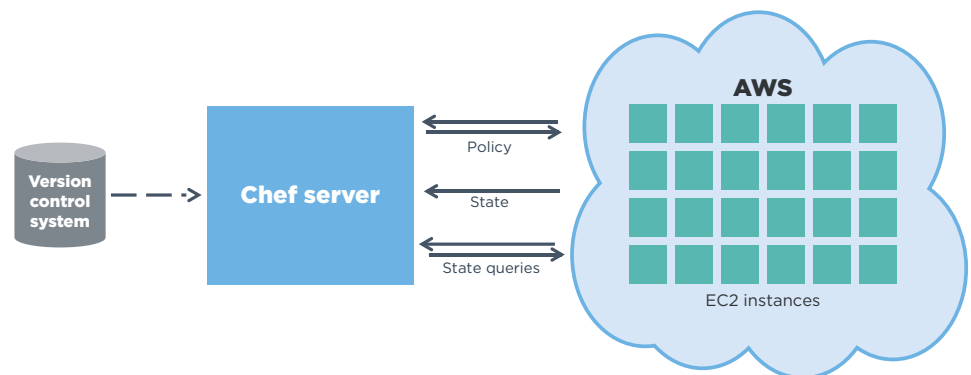
Examples of using Chef with AWS

In this section, you'll see two examples of how you can use Chef with AWS. Although these examples are simple, you can use the same principles to automate large, complex environments.

Registering with an ELB
Here is an example of a recipe that registers the node with an AWS Elastic Load Balancer (ELB) named elb_qa. To do this, the recipe uses the aws_elastic_lb resource that's found in the aws cookbook.

```
# Load your AWS credentials databag
include_recipe 'aws'
aws = data_bag_item('aws', 'main')


aws_elastic_lb 'elb_qa' do
  aws_access_key aws['aws_access_key_id']
  aws_secret_access_key aws['aws_secret_access_key']
  name 'elb_qa'
  action :register
end
```

This code pulls down data bag items to retrieve the keys. A data bag is a collection of key-value pairs that stores configuration state on the Chef server. In this example, the data bag stores AWS access credentials. You can encrypt data bags to protect sensitive information, such as certificates, API keys, and passwords.

[1] https://supermarket.chef.io/cookbooks/aws

Managing user access

Here is an example of using Chef to help manage user access to AWS instances across availability zones and regions. There is a Chef community cookbook called users that solves the problem. It lets you store Linux user data in a data bag, and it automates the process of managing users and their SSH keys.

Below is a JSON output for a data bag item that describes the desired account settings for a user named han. Assume that the name of the data bag itself is users (by default, but the name is configurable).

```
han.json
{
    "id"            : "han",
    "comment"       : "Han Solo",
    "home"          : "opt/carbonite",
    "groups"        : ["rebels", "scoundrels", "sysadmins"],
    "ssh_keys"      : [
        "AAA123…xyz== foo",
        "AAA456…uvw== bar"
    ]
}
```

The users cookbook lets you write recipes that manage account settings based on data in the users data bag. For example, this recipe ensures that the rebels group exists and that it includes the correct user accounts, based on the data bag.

```
users_manage 'rebels' do
    group_id 1138
    action [:create]
end
```

The users_manage resource leverages the user, group, directory, and template resources to ensure that the group rebels is created on the node being configured, that the group rebels is set to gid 1138, that the group has the correct set of users (including user han), and that any SSH keys associated with those users are present.

[2] https://supermarket.chef.io/cookbooks/users

Other examples
There are many other cookbooks that can help you manage your AWS environment:

- The AWS Route 53 cookbook at https://supermarket.chef.io/cookbooks/route53 helps you manage the AWS Route 53 DNS service.

- The AWS Security cookbook at https://supermarket.chef.io/cookbooks/aws_security helps you manage AWS security groups and rules.

- The Amazon Relational Database Service (Amazon RDS) cookbook at https://supermarket.chef.io/cookbooks/aws-rds helps you manage the AWS relational database service.

Chef on the AWS Marketplace

The easiest way to get a Chef server is to use the image available in the AWS Marketplace. If you use any of its enterprise-grade features, the first five nodes are free and, after, that, you are charged on a pay-as-you-go basis. This kind of utility-based billing model for Chef is unique to AWS. You're only charged for the nodes you use, for the hour you use them.

Provisioning AWS with Chef

Chef provisioning lets you create nodes and configure them, including Amazon Elastic Compute Cloud (Amazon EC2) instances. Rather than bringing up individual nodes, you can use Chef provisioning to describe an entire cluster or fleet. You can build your environment as many times as you want in the AWS Cloud, on virtual machines in your data center, or even bare metal.

Chef provisioning for AWS includes the most commonly used AWS resources. For example, you can manage Amazon EC2 instances, Amazon Virtual Private Clouds (VPC), security groups, AWS Identity and Access management (IAM) roles and instance profiles. Chef provisioning can also consume IAM roles so that all the components in the stack that have the appropriate IAM profile are provisioned correctly. You can also manage common AWS services such as Amazon RDS and Amazon Route 53

You can see a complete list of list of AWS resources at https://docs.chef.io/release/devkit/provisioning_aws.html.

Using AWS CloudFormation with Chef

AWS CloudFormation is a provisioning service that is based on templates. A template is a description of your AWS resources that is written in JSON. A basic CloudFormation template includes:
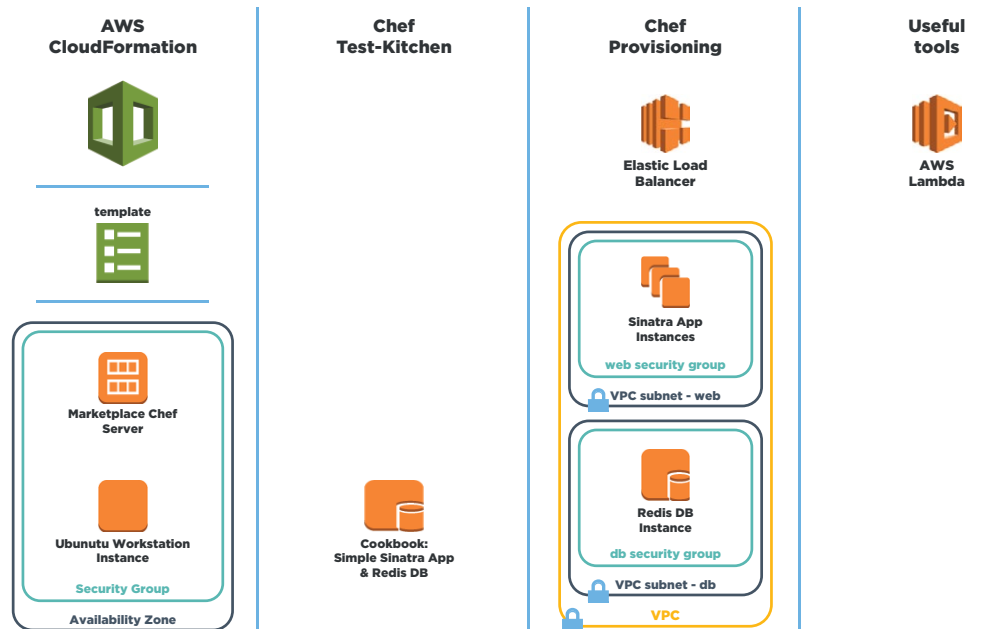
- **Format version**. This is the latest version of the template.

- **Description**. This lets you know what the template does.

- **Parameters**. These let you customize a template with specific values, such as a domain name or database password.

- **Resources**. These are the smallest pieces of infrastructure that you can describe, such as a load balancer or an elastic IP address.

- **Outputs**. These return values, such as the public name of an Amazon EC2 instance.

One reason to use AWS CloudFormation is because you want to configure AWS resources that aren't described by Chef provisioning. Often, people use AWS CloudFormation in conjunction with Chef. They use the templates to set up the AWS workloads, the Chef server and the nodes. Then, they use Chef to handle the network and to configure the applications.

Keep in mind that Chef provisioning automatically performs some of the bootstrapping actions for you that you must do manually with AWS CloudFormation. For example, Chef provisioning takes care of how a new piece of environment authenticates against the Chef server. To see everything you need to account for, go to https://docs.chef.io/install_bootstrap.html.

Tying It All Together

A good way to get started exploring how Chef integrates with AWS, how to use AWS CloudFormation and Chef provisioning, and to experiment with a DevOps workflow is with the code included in the GitHub repo located at https://github.com/FastRobot/chef_aws_demo. This repo contains a demo that sets up a simple Sinatra application with a Redis database. Here is a diagram of the topology.

AWS CloudFormation	Chef Test-Kitchen	Chef Provisioning	Useful tools
template		Elastic Load Balancer	AWS Lambda
Marketplace Chef Server		Sinatra App Instances	
		web security group	
		VPC subnet - web	
Ubunutu Workstation Instance	Cookbook: Simple Sinatra App & Redis DB	Redis DB Instance	
Security Group		db security group	
Availability Zone		VPC subnet - db	
		VPC	

First, an AWS CloudFormation template asks you a few questions, and then spins up a Chef server from the AWS Marketplace (t2.medium) and a preconfigured Ubuntu workstation (t2.medium) that can authenticate against the Chef server. These two components exist within their own security group.

A Chef provisioning recipe sets up the AWS workloads and the networking. It also deploys the web application and local database. It separates the application instance from the Redis database instance and gives each its own Amazon VPC and security group. It also places them both behind a load balancer.

Using Test Kitchen for automated testing

Test Kitchen is a tool that runs your infrastructure code in an isolated environment that resembles your production environment. With Test Kitchen, you continue to write your Chef code from your workstation, but instead of uploading your code to the Chef server and applying it to a node, Test Kitchen applies your code to a temporary environment, such as a virtual machine on your workstation or an Amazon EC2 instance. For the demo, Test Kitchen is automatically installed on the workstation.

Tying It All Together

A good way to get started exploring how Chef integrates with AWS, how to use AWS CloudFormation and Chef provisioning, and to experiment with a DevOps workflow is with the code included in the GitHub repo located at https://github.com/FastRobot/chef_aws_demo. This repo contains a demo that sets up a simple Sinatra application with a Redis database. Here is a diagram of the topology.

First, an AWS CloudFormation template asks you a few questions, and then spins up a Chef server from the AWS Marketplace (t2.medium) and a preconfigured Ubuntu workstation (t2.medium) that can authenticate against the Chef server. These two components exist within their own security group.

A Chef provisioning recipe sets up the AWS workloads and the networking. It also deploys the web application and local database. It separates the application instance from the Redis database instance and gives each its own Amazon VPC and security group. It also places them both behind a load balancer.

Using Test Kitchen for automated testing
Test Kitchen is a tool that runs your infrastructure code in an isolated environment that resembles your production environment. With Test Kitchen, you continue to write your Chef code from your workstation, but instead of uploading your code to the Chef server and applying it to a node, Test Kitchen applies your code to a temporary environment, such as a virtual machine on your workstation or an Amazon EC2 instance. For the demo, Test Kitchen is automatically installed on the workstation.

To get you started, the demo includes unit tests written with ChefSpec and integration tests written with Serverspec. ChefSpec verifies that resources are behaving appropriately. Serverspec verifies that your servers are configured correctly. You can experiment by making changes to the Chef code and then, with Test Kitchen and automated tests, get fast feedback on whether your changes do what you intended.

Once you are happy with your tested changes, you can use a different cookbook, also included in the repo, to automatically deploy the sample application to Amazon EC2 instances that Chef will provision for you, put into different secure subnets and register with an AWS ELB for redundancy.

Automated testing and DevOps
Automated testing is a critical part of a DevOps workflow. With it, you can easily collaborate with others to write code and accept contributions to your codebase because you can be confident that the code works before you deploy it to production. Automated tests can also be incorporated into continuous delivery pipelines such as the one included as part of Chef Automate.
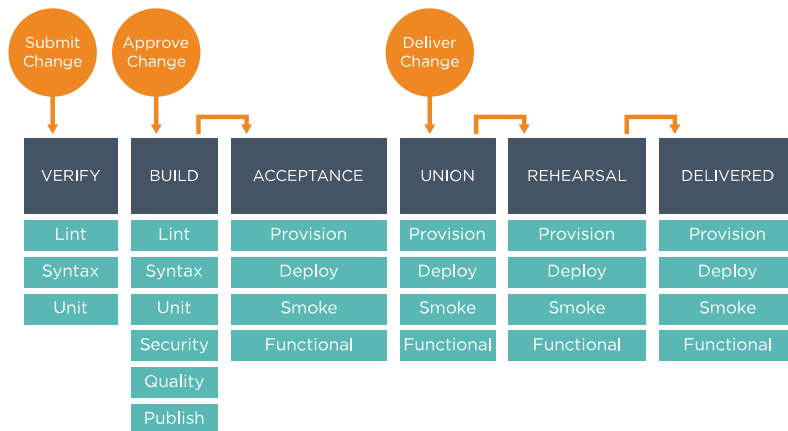
What is Chef Automate?

Chef Automate is enterprise software that extends Chef with features that make it easy for companies to implement high-velocity DevOps processes. With Chef Automate, enterprises that are struggling with long-established silos and manual procedures can jumpstart their adoption of DevOps so that they too can deliver their products quickly and reliably in cloud-only and hybrid environments.

Chef Automate lets you manage changes to both environment and application code, giving your operations and development teams a common way to develop, test, and deploy cookbooks, applications, and more. Chef Automate is based on DevOps best practices for continuous deployment.

Changes to environment and applications are tracked by a version control system such as Git and pass through quality gates. When to release the change is a business decision. When that decision is made, an authorized person presses the "Deliver" button in the user interface. Chef Automate then promotes the artifact to a series of environments that comprise the automated release process.

Here is a diagram of the Chef Automate workflow.

Automated tests move a change from a developer's workstation all the way to production. Your team approves a proposed change, and, after acceptance tests, releases the corresponding artifact for delivery into production.

Submit Change	Approve Change		Deliver Change		
VERIFY	**BUILD**	**ACCEPTANCE**	**UNION**	**REHEARSAL**	**DELIVERED**
Lint	Lint	Provision	Provision	Provision	Provision
Syntax	Syntax	Deploy	Deploy	Deploy	Deploy
Unit	Unit	Smoke	Smoke	Smoke	Smoke
	Security	Functional	Functional	Functional	Functional
	Quality				
	Publish				

The Union and Rehearsal stages enable cross-team collaboration—they check that components produced by one team or organization work consistently with components produced by other teams. After a final series of tests, components enter the Delivered stage, which corresponds to the production environment.

Chef Automate provides change management and rapid deployment for the entire technology stack. With it, companies have a well-defined process that moves changes, features and new services to production quickly and safely.

Case Study: Gannett

Gannett is a national and local newspaper and media company. Its national brand is USA Today. It also owns more than 92 media companies in 33 states, such as the Arizona Republic and the Indianapolis Star.

For many years, Gannett's deployment workflow was characterized by multiple handoffs and manual testing. 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.

"When the opportunity arose to rebuild a hybrid architecture on AWS for a development environment that would mimic production, the team decided to use Chef to manage the infrastructure"

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.

As a reaction to the situation, a kind of "shadow IT" evolved on the development side, with developers spinning up Amazon EC2 instances and using personal Heroku accounts, and then linking these to production DNS. However, there was no oversight over the costs of applications, and security had no way to audit the stacks.

"It was a situation that had to change," says Erik Bursch, Gannett's Vice President for Platform as a Service. "Our task is to consistently provide customers with the fastest and best digital experiences possible. Critical to that effort is our ability to react faster. We can't underestimate the value of speed and consistency in getting our products to the market."

Bursch says Gannett recognized the cost and agility benefits of the cloud. He also knew that, with AWS, developers would use standardized tools and resources and would benefit from AWS's scalability and its cost-efficient, compute-on-demand model.

When the opportunity arose to rebuild a hybrid architecture on AWS for a development environment that would mimic production, the team decided to use Chef to manage the infrastructure. The improvement was immediate. Other developers noticed, and soon discussions turned to the possibility of automating both Gannett's development and operations processes using AWS and Chef.

The benefits have been impressive. Different teams are working more closely together, there is greater visibility for tracking and auditing changes throughout the environment and, most importantly, applications share a common deployment methodology that can be customized to expedite application delivery.

Read the complete story at https://aws.amazon.com/solutions/case-studies/gannett/

Watch the video, USA Today Brings Shadow IT into the Light at https://www.youtube.com/watch?v=goRTQs7oGsk

"We've been able to reduce application deployment times from weeks using our former on-premises infrastructure to just minutes using AWS and Chef," says Bursch. "Instead of a single desktop application deployment in a week, like we experienced in the past, we're now deploying an average of 25 per week. That means more timely services for our customers by using the latest digital technologies to build and retain readership."

Key Points

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

- DevOps is a cultural and technical movement that allows companies to deliver value to their customers quickly and safely.

- DevOps cultural values emphasize communication and cooperation and discourage 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.

- Chef is a dynamic, policy-based automation platform.

- Chef and AWS are tightly integrated. Chef has many resources that are specific to AWS and that allow you to manage your entire AWS stack.

- You can use AWS CloudFormation in tandem with Chef to provision your network.

- Chef Automate provides enterprise-level features that let organizations jumpstart their adoption of DevOps.

- Together, AWS and Chef can radically lower the time it takes to deploy, and encourage DevOps values such as collaboration and transparency.

Resources

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

Chef cookbooks
https://supermarket.chef.io/cookbooks/aws
https://supermarket.chef.io/cookbooks/aws-rds
https://supermarket.chef.io/cookbooks/route53
https://supermarket.chef.io/cookbooks/aws_security
https://supermarket.chef.io/cookbooks/users

Chef tools
Chef Development Kit (Chef DK) at https://downloads.chef.io/chef-dk/
ChefSpec at https://docs.chef.io/chefspec.html
Serverspec at http://serverspec.org/

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
Chef web site at https://www.chef.io/

Learn Chef web site at https://learn.chef.io/

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

List of Chef provisioning resources for AWS at https://docs.chef.io/release/devkit/provisioning_aws.html

Chef bootstrapping considerations at https://docs.chef.io/install_bootstrap.html

AWS services
AWS CloudFormation service at https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html

Amazon CloudWatch service at https://aws.amazon.com/cloudwatch/
AWS IAM service at https://aws.amazon.com/iam/

AWS Key Management Service (KMS) at https://aws.amazon.com/kms/
AWS Lambda service at https://aws.amazon.com/lambda/

AWS and Chef

Chef Server 12 in AWS Marketplace at https://aws.amazon.com/marketplace/seller-profile/ref=srh_res_product_vendor?ie=UTF8&id=e7b7691e-634a-4d35-b729-a8b576175e8c

Demo repo at https://github.com/FastRobot/chef_aws_demo

Automatically Delete Terminated Instances in Chef Server with AWS Lambda and Amazon CloudWatch Events blog post at https://aws.amazon.com/blogs/apn/automatically-delete-terminated-instances-in-chef-server-with-aws-lambda-and-cloudwatch-events/

Gannett Case Study at https://aws.amazon.com/solutions/case-studies/gannett/

Cooking with AWS at https://www.youtube.com/watch?v=NWhiWB87Wok

Chef and AWS: Your Path to DevOps at https://www.chef.io/webinars/?commid=202161

USA Today Brings Shadow IT into the Light at https://www.youtube.com/watch?v=goRTQs7oGsk

Chef on AWS: Deep Dive at https://www.youtube.com/watch?v=ZL6quvVvnig

AWS DevOps overview at https://aws.amazon.com/devops/

AWS What is DevOps? https://aws.amazon.com/devops/what-is-devops/

About CHEF

Chef, an Advanced Technology Partner in the AWS Partner Network (APN), provides DevOps automation tools. For more information about how Chef can help your company build, deploy, and manage your infrastructure, see the Chef listing in the AWS Partner Directory.

About AWS

For 10 years, Amazon Web Services has been the world's most comprehensive and broadly adopted cloud platform. AWS offers over 70 fully featured services for compute, storage, databases, analytics, mobile, Internet of Things (IoT) and enterprise applications from 33 Availability Zones (AZs) across 12 geographic regions in the U.S., Australia, Brazil, China, Germany, Ireland, Japan, Korea, and Singapore. AWS services are trusted by more than a million active customers around the world – including the fastest growing startups, largest enterprises, and leading government agencies – to power their infrastructure, make them more agile, and lower costs. To learn more about AWS, visit http://aws.amazon.com.