DevOps and the Cloud: Chef and Amazon Web Services

WHITEPAPER

Table of Contents

Introduction

This paper is an introduction to how using DevOps patterns with Amazon Web Services (AWS) 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 manages resources. The examples use Chef for automation and AWS as the cloud provider.

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. We call this new way of working DevOps.

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.

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

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 two handoffs.

- **12%** of the knowledge left after three handoffs.

- **6%** of the knowledge left after four handoffs.

- **3%** of the knowledge left after five 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 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 to 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 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. One aspect of an automation platform is that it 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.

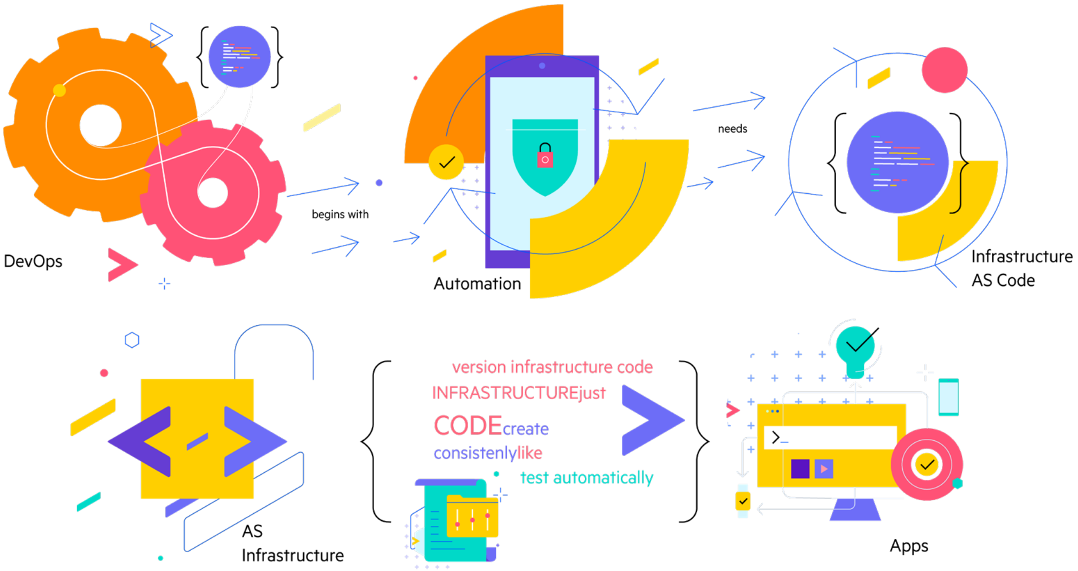
You can treat your infrastructure code just as you would your application code. The code is versionable, testable, and repeatable. You can (and should) use the same deployment pipeline for your infrastructure as you do for your applications.

> **“Technology is hard. Managing constantly evolving, complex systems is very difficult. A disciplined approach to automation is required to keep up with the nonstop change.”**
>
> **Aaron Kraft, Vice President of Engineering**
> **Progress** 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.

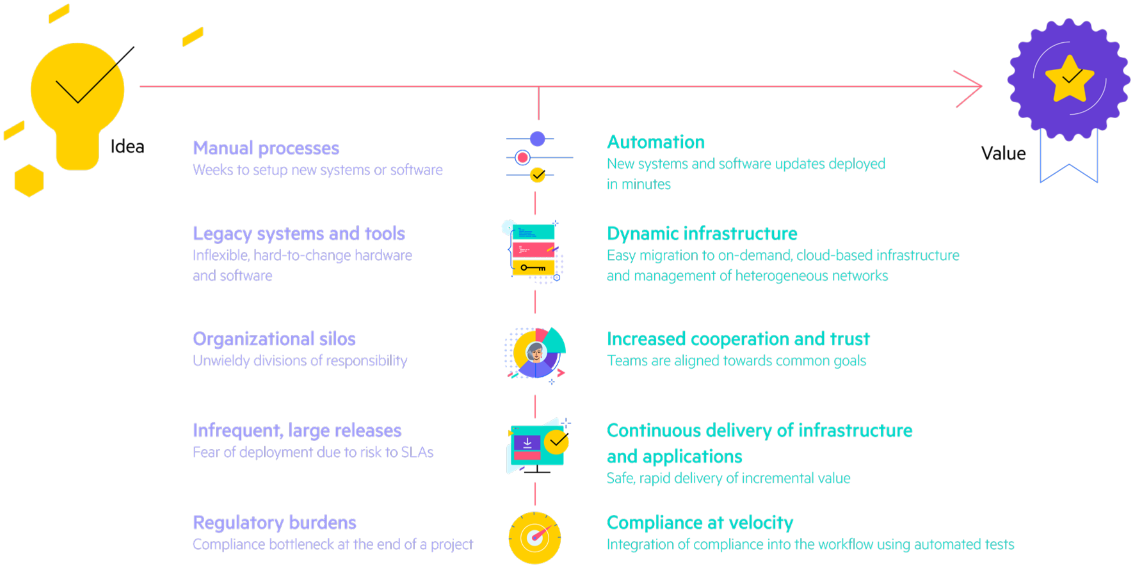
Another advantage to automation is that infrastructure code is expressed as human-readable text files. DevOps encourages transparency. Describing your infrastructure as code means that it is accessible and readable to everyone on the team. In addition, you can keep these files in a source control system, where they are versioned and kept safe. All of 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.



Manual processes
Weeks to setup new systems or software

Automation
New systems and software updates deployed in minutes

Legacy systems and tools
Inflexible, hard-to-change hardware and software

Dynamic infrastructure
Easy migration to on-demand, cloud-based infrastructure and management of heterogeneous networks

Organizational silos
Unwieldy divisions of responsibility

Increased cooperation and trust
Teams are aligned towards common goals

Infrequent, large releases
Fear of deployment due to risk to SLAs

Continuous delivery of infrastructure and applications
Safe, rapid delivery of incremental value

Regulatory burdens
Compliance bottleneck at the end of a project

Compliance at velocity
Integration of compliance into the workflow using automated tests

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

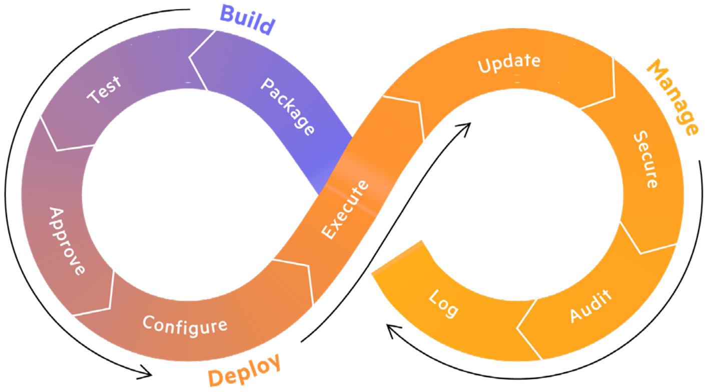
Companies that practice DevOps are twice as likely to exceed profitability, market share and product goals. They exhibit a 50% market cap growth over 3 years.

What Is Chef Automate?

Chef on AWS Marketplace gives you everything you need to build, deploy, and manage your applications and infrastructure at speed.

Collaborate

Workflow → Local Development → Integration → Tooling (APIs & SDKs)



OSS Automation Engines

Progress® Chef®   Progress® Chef® Habitat®   Progress® Chef® InSpec®

Chef® provides an enterprise platform that allows developers, operations, and security engineers to collaborate effortlessly on delivering application/infrastructure changes and compliance at the speed of business.

Collaborate. Chef provides a pipeline for the continuous deployment of infrastructure and applications. Chef also includes tools for local development and can integrate with a variety of third-party products for developer workflows.

Build. Use Chef and its continuous integration and deployment workflow to test and approve code changes across all levels of the stack, then package and publish them to a repository.

Deploy. With Chef, you will provision and update environments quickly and prevent configuration drift.

Manage. Use Chef to make your security and compliance requirements a part of an automated workflow. When compliance is code, you can find problems early in the development.

Cef provides a single dashboard and analytics for infrastructure and DevSecOps automation. It visualizes data provided by Chef's three core engines, provides consolidated observability into configuration, security, and compliance.

Application Automation

Chef Habitat revitalizes legacy applications by repackaging them, applying continuous delivery practices, and deploying them to cloud-native environments.

Security Automation

security policies. Chef provides nearly 500+ resources including various cloud resources – AWS, Ali baba, Azure, and GCP. For more information visit: https://docs.chef.io/inspec/resources/

Infrastructure Automation

Chef Infra makes infrastructure configurations testable, portable, and auditable.

Automated configuration management enables consistent configurations at scale ensuring configuration policy is flexible, version able, testable, and human readable.

Use Chef to manage AWS resources and services such as EC2 instances, Security Groups, Elastic Load Balancers (ELB), Elastic Block Storage Volumes, Route 53, relational databases (RDS) and more. You can take advantage of cookbooks provided by the Chef community, which contain code for managing AWS resources.

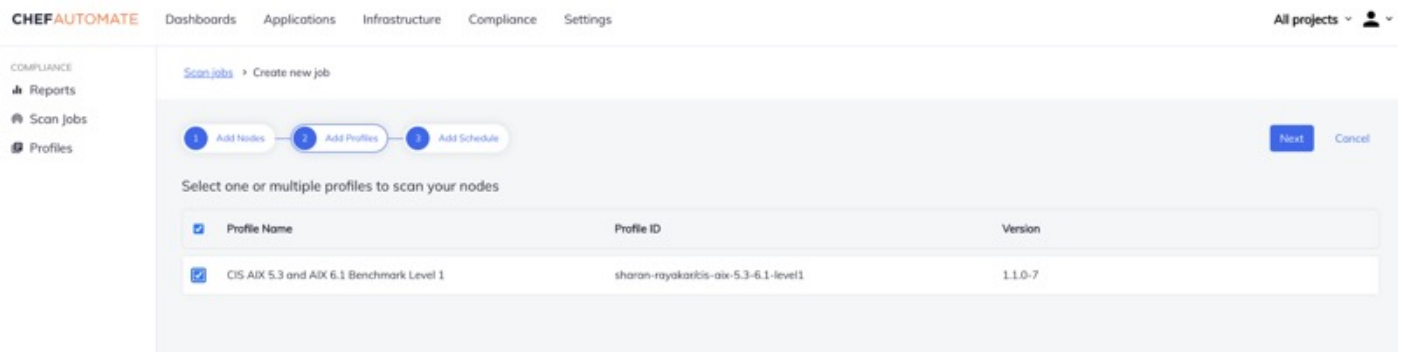
Chef measurably increases the ability to deliver software quickly, increasing speed and efficiency while decreasing risk.

Chef provides operational visibility for today's Coded Enterprise with:

- Real-Time data across the estate: Filterable dashboards collecting configuration and compliance details for every data center, cloud provider, and environment managed in aggregated dashboards.

- Effortless collaboration among teams: Dev, Ops, Security, and Compliance teams who are responsible for delivering software safely and efficiently, share a consistent view of how environments are built and validated.

- Powerful auditing capabilities: Create agentless compliance scans directly in the Web UI. Scan traditional servers, VMs, cloud environments, and SaaS solutions all in one place.

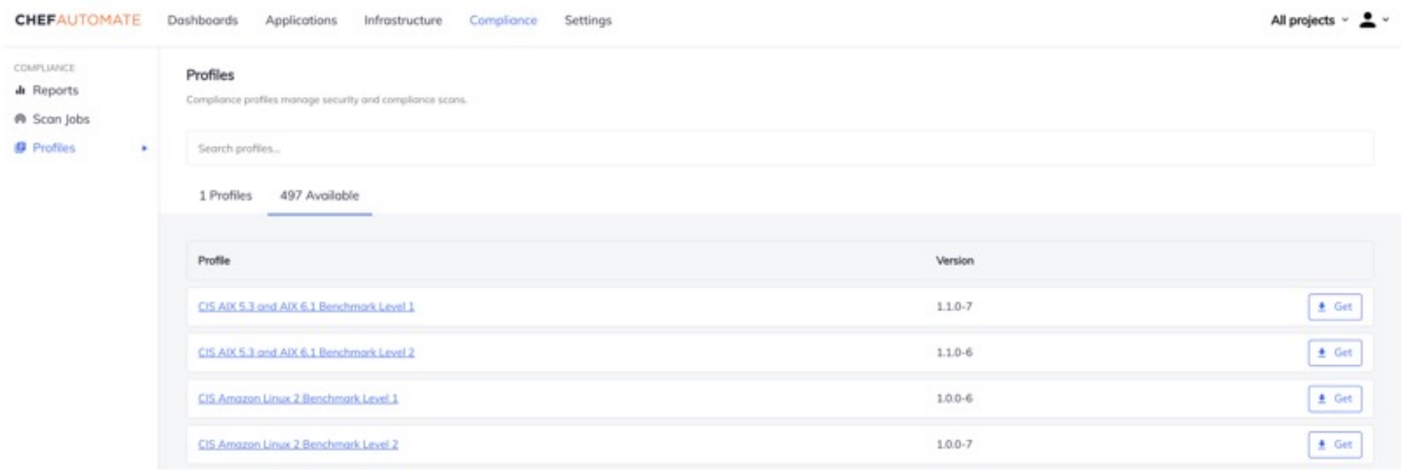


Select one or multiple profiles to scan your nodes

☑	Profile Name	Profile ID	Version
☑	CIS AIX 5.3 and AIX 6.1 Benchmark Level 1	sharon-rayakar/cis-aix-5.3-6.1-level1	1.1.0-7

Intelligent access controls: Ensure the right teams have the right access using existing access control solutions (LDAP/SAML)

- Pre-built, supported compliance assets: Leverage pre-created profiles to validate software patches, system security, and evaluate against industry-standard frameworks such as CIS Benchmarks and DISA STIGs.

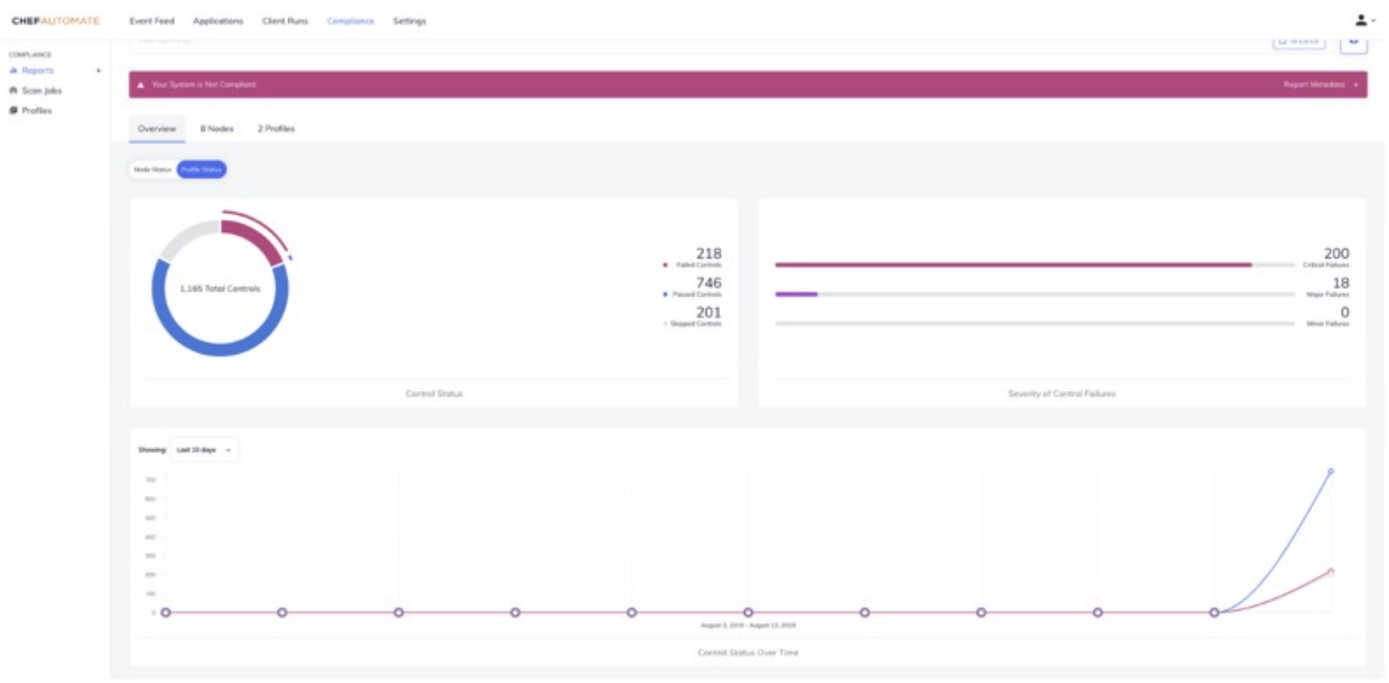


Profiles

Compliance profiles manage security and compliance scans.

Profile	Version
CIS AIX 5.3 and AIX 6.1 Benchmark Level 1	1.1.0-7
CIS AIX 5.3 and AIX 6.1 Benchmark Level 2	1.1.0-6
CIS Amazon Linux 2 Benchmark Level 1	1.0.0-6
CIS Amazon Linux 2 Benchmark Level 2	1.0.0-7

- Actionable Insights at Any Scale: Chef provides actionable insights with enterprise-scale and performance across multiple data centers and cloud providers with an auditable history of changes to environments.



Understanding Chef

Any machine managed by Chef is called a node. A node can be physical, virtual, in the cloud, or even a container instance.

A Chef resource describes some piece of infrastructure, 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 load balancer. Recipes are developed on local workstations and stored in the version control system. Collections of recipes are called cookbooks.

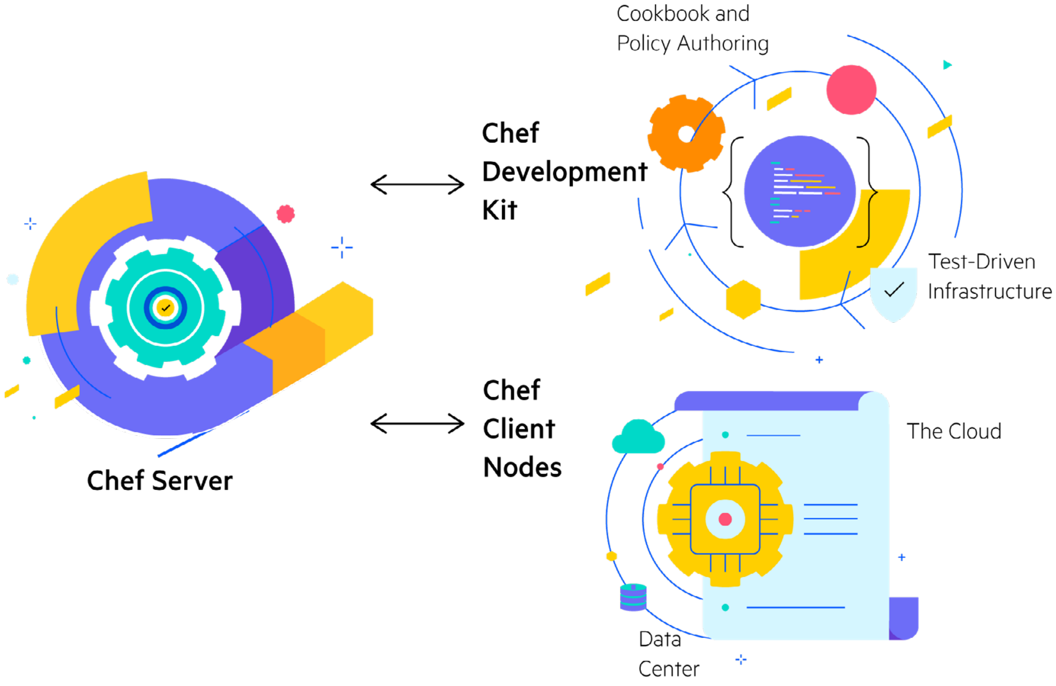
Chef uses a client/server architecture to manage the nodes in your network. The Chef client is installed on every node and periodically polls the Chef server for the latest recipes. The Chef client runs the recipes and brings the node to the correct state, over time. Chef clients also notify the server 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 infrastructures 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 infrastructures.

Registering with an ELB

Here is an example of a recipe that registers the node with an Elastic Load Balancing (ELB) load balancer 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.

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.2 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 are created on the node that is being configured; that the group rebels are 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.

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 Amazon RDS.

Examples of using Chef InSpec with AWS

As of Chef InSpec 2.0, we have expanded our platform support beyond individual machines and now include support for select AWS, Alibaba, GCP and Azure resources.

Using InSpec, you can use several Chef InSpec resources to audit properties of your cloud infrastructure - for example, an Amazon Web Services Elastic Load Balancer.

AWS Platform Support in InSpec

Setting up AWS credentials for InSpec

Chef InSpec uses the standard AWS authentication mechanisms. Typically, you will create an IAM user specifically for auditing activities.

- Create an IAM user in the AWS console, with your choice of username. Check the box marked “Programmatic Access.”
- On the Permissions screen, choose Direct Attach. Select the AWS-managed IAM Profile named “ReadOnlyAccess.” If you wish to restrict the user further, you may do so; see individual Chef InSpec resources to identify which permissions are required.
- After generating the key, record the Access Key ID and Secret Key.

Using Environment Variables to provide credentials

You may provide the credentials to Chef InSpec by setting the following environment variables: `AWS_REGION`, `AWS_ACCESS_KEY_ID`, and `AWS_SECRET_ACCESS_KEY`. You may also use `AWS_PROFILE`, or if you are using MFA, `AWS_SESSION_TOKEN`

Once you have your environment variables set, you can verify your credentials by running:

```
$ inspec detect -t aws://
== Platform Details
Name:      aws
Families:  cloud, api
Release:   aws-sdk-v2.10.125
```

Using the Chef InSpec target option to provide credentials on AWS

Look for a file in your home directory named `~/.aws/credentials`. If it does not exist, create it. Choose a name for your profile; here, we're using the name 'auditing'. Add your credentials as a new profile, in INI format:

```
[auditing]
aws_access_key_id = AKIA....
aws_secret_access_key = 1234....abcd
```

You may now run Chef InSpec using the `--target / -t` option, using the format `-t aws://region/profile`. For example, to connect to the Ohio region using a profile named 'auditing', use `-t aws://us-east-2/auditing`.

To verify your credentials, run

```
$ inspec detect -t aws://
== Platform Details
Name:      aws
Families:  cloud, api
Release:   aws-sdk-v2.10.125
```

Chef supplies nearly 500+ resources including various cloud resources – AWS, Ali baba, Azure and GCP. For more information visit : https://docs.chef.io/inspec/resources/

For e.g : Use the `aws_alb` InSpec audit resource to test properties of a single AWS Application Load Balancer (ALB). This resource is available in the Chef InSpec AWS resource pack.

Ensure that an `aws_alb` exists

```
describe aws_alb(‘arn:aws:elasticloadbalancing’) do
  it { should exist }
end
```

```
describe aws_alb(load_balancer_arn: ‘arn:aws:elasticloadbalancing’) do
  it { should exist }
end
```

Parameters :

`load_balancer_arn` : This resource accepts a single parameter, the ALB Arn which uniquely identifies the ALB. This can be passed either as a string or as a `load_balancer_arn`: ‘value’ key-value entry in a hash.

Properties :

`load_balancer_name` : The name of the load balancer.

`load_balancer_addresses` : A collection of the load balancer addresses.

`canonical_hosted_zone_id` : The ID of the Amazon Route 53 hosted zone for the load balancer.

`dns_name` : The DNS name of the load balancer.

`availability_zones` : The Availability Zones for the load balancer.

`security_groups` : The security groups for the load balancer. Valid only for load balancers in a VPC.

`scheme` : The type of load balancer. Valid only for load balancers in a VPC.

`state` : The state of the load balancer.

`subnets` : A collection of the subnet ids.

Example. : **Test that an ALB has its availability zones configured correctly.**

```
describe aws_alb(‘arn::alb’) do
  its(‘zone_names.count’)  { should be > 1 }
  its(‘zone_names’)        { should include ‘us-east-2a’ }
  its(‘zone_names’)        { should include ‘us-east-2b’ }
end
```

Test whether the access log is enabled for the Application Load Balancer.

```
describe aws_alb(‘arn::alb’) do
  its(‘zone_names.count’)  { should be > 1 }
  its(‘zone_names’)        { should include ‘us-east-2a’ }
  its(‘zone_names’)        { should include ‘us-east-2b’ }
end
```

Other Examples

There are many other resources that can help you audit your AWS environment using Chef InSpec:

- The AWS S3 Resources helps you to audit S3 related properties:
  aws_s3_access_point Resource
  aws_s3_access_points Resource
  aws_s3_bucket Resource
  aws_s3_bucket_object Resource
  aws_s3_bucket_objects Resource
  aws_s3_bucket_policy Resource
  aws_s3_buckets Resource

- The AWS Route 53 Chef InSpec resources helps you audit the AWS Route 53 DNS service.
  aws_route53_record_set Resource
  aws_route53_record_sets Resource
  aws_route53resolver_resolver_endpoint Resource
  aws_route53resolver_resolver_endpoints Resource
  aws_route53resolver_resolver_rule Resource
  aws_route53resolver_resolver_rule_association Resource
  aws_route53resolver_resolver_rule_associations Resource
  aws_route53resolver_resolver_rules Resource

- The Amazon Relational Database Service (Amazon RDS) Chef InSpec resources help you audit Amazon RDS service
  aws_rds_cluster Resource
  aws_rds_clusters Resource
  aws_rds_db_cluster_snapshot Resource
  aws_rds_db_cluster_snapshots Resource
  aws_rds_db_proxy Resource
  aws_rds_db_proxy_endpoint Resource
  aws_rds_db_proxy_endpoints Resource

Getting Chef on the AWS Marketplace

There are three ways to get Chef on AWS, in an AWS environment to automate large, complex infrastructures.

Private Offers

Purchase Chef Licenses using your AWS dollars with the transaction completed through the marketplace with an offer customized for your needs.

Chef on AWS Marketplace

Get all the benefits of Chef Automate in an easy to deploy model that lets you manage your upgrade and back-up strategy.

Chef BYOL Self-Hosting

If you want complete control of your Chef Automate installation, you can also install Chef Automate on Amazon EC2 instances yourself.

Provisioning AWS with Chef

Chef provisioning lets you create nodes and configure them, including 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 infrastructure as many times as you want in the cloud, on virtual machines in your data center, or even bare metal.

Chef provisioning for AWS includes the most used AWS resources. For example, you can manage Amazon EC2 instances, Amazon Virtual Private Clouds (VPCs), security groups, 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 Relational Database Service (RDS) and Amazon Route 53.

AWS CloudFormation is an AWS provisioning service that is based on templates. A template is a description of your AWS infrastructure 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 server.

A reason to use AWS CloudFormation is that you want to configure AWS resources that aren't described by Chef provisioning. Often, people use CloudFormation in conjunction with Chef. They use the templates to set up the AWS infrastructure, the Chef server, and the nodes. Then, they use Chef to handle the network and configure the applications. Remember that you can also bootstrap your node with CloudFormation.

Learn Chef Tutorials

A good way to get started exploring how Chef integrates with AWS is to try out the tutorials on Learn Chef. Start with the "Manage a node" tutorial. In this tutorial, you'll use Chef to configure a node, check that its configuration is up to date, and change that configuration. You can use either AWS OpsWorks for Chef Automate or Amazon Web Services to set up the Chef server.

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 to production. Automated tests can also be incorporated into continuous delivery pipelines such as the one included as part of Chef Automate.

Using Test Kitchen for Automated Testing

If you want complete control of your Chef Automate installation, you can also install Chef Automate on Amazon EC2 instances yourself.

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 Elastic Load Balancer (ELB) for redundancy.

Case Study: Gannett

Gannett is a national and local newspaper and media company. Its national brand is USA Today. It also owns more than 120 media companies in 46 states, such as the ReachLocal, UpCurve, and WordStream.

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.

There were two operations teams, each in its own silo both physically (i.e. 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 Elastic Compute Cloud (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.

> **"We've been able to reduce application deployment times from weeks using our former on-premises infrastructure to just minutes using AWS and Chef."**
>
> **Erik Bursch, Vice President for Platform as a Service**
> **Gannett**

“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.

“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.”

- 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 should include the ability to describe your infrastructure as code.
- Chef Automate gives you everything you need to build, deploy, and manage your applications and infrastructure at speed.

- Chef Automate 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.

- Together, AWS and Chef Automate can radically speed up deployment times.

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 Automate

Chef Automate at https://www.chef.io/automate/
Chef at https://www.chef.io/chef/
InSpec at https://www.chef.io/inspec/
Habitat at https://www.habitat.sh/

Read the complete story at https://www.chef.io/customers/gannett

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

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

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

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

AWS KMS service at https://aws.amazon.com/kms/

AWS Lambda service at https://aws.amazon.com/lambda/

AWS and Chef Automate

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

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

AWS

For 15 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.

About Progress

Progress (NASDAQ: PRGS) provides the leading products to develop, deploy and manage high-impact business applications. Our comprehensive product stack is designed to make technology teams more productive and enable organizations to accelerate the creation and delivery of strategic business applications, automate the process by which apps are configured, deployed and scaled, and make critical data and content more accessible and secure—leading to competitive differentiation and business success. Learn about Progress at www.progress.com or +1-800-477-6473.

f /progresssw
/progresssw
▶ /progresssw
in /progress-software