Automation and the DevOps Workflow

http://www.chef.io

10/2015

Automation and the DevOps Workflow

Executive Summary

The advent of the digital economy has fundamentally changed consumption patterns. Today's customers are accustomed to goods and services that are available online anytime, anywhere and from any type of device. To satisfy these new expectations, every enterprise must transform the way it does business or risk obsolescence. Enterprise IT, once considered a back office support function, must become the front office connection to customers and the linchpin of a mission-critical transformation.

Innovative companies such as Amazon, Google, and Facebook have set the pattern for this transformation. These innovators, along with an active community of web practitioners, have developed a workflow and set of practices, encapsulated by the term DevOps, that brought about many technical and cultural changes to IT and resulted in infrastructures and applications that were extraordinarily fast and scalable.

The hallmarks of a DevOps workflow are velocity, consistency, scale and the ability to incorporate feedback.

VELOCITY CONSISTENCY SCALE FEEDBACK

DevOps is at the heart of becoming a software-led company, and it is no longer for the select few. DevOps is becoming the norm for how enterprises handle change, whether to their infrastructure or to their applications.

However, established enterprises have a challenge that the DevOps innovators did not. They have legacy workloads and regulatory requirements to consider. Migrating these workloads to a flexible technology stack must be done in a way that reduces risk, ensures stability and guarantees regulatory compliance. In addition, established enterprises have existing processes for managing change, that often involves many manual steps. These processes must be replaced by a more automated and collaborative workflow.

This paper focuses on the technical attributes of automation and the DevOps workflow and shows how they help you meet the demands of the digital economy.

Every Business Is Software

Every business is a software business, no matter what industry it's in. Customers, whether outside the corporate firewall or behind it, have higher expectations for speed, reliability, and personalized content. They expect fast response times and great customer service.

These expectations exist because innovative online companies such as Amazon, Google, and Facebook constantly raise consumer expectations. Any business that wants to remain competitive must emulate the patterns developed by the web innovators, which are summarized by the term DevOps. Following DevOps patterns allows companies to quickly respond to customer demands with the software that provides the experience customers want.

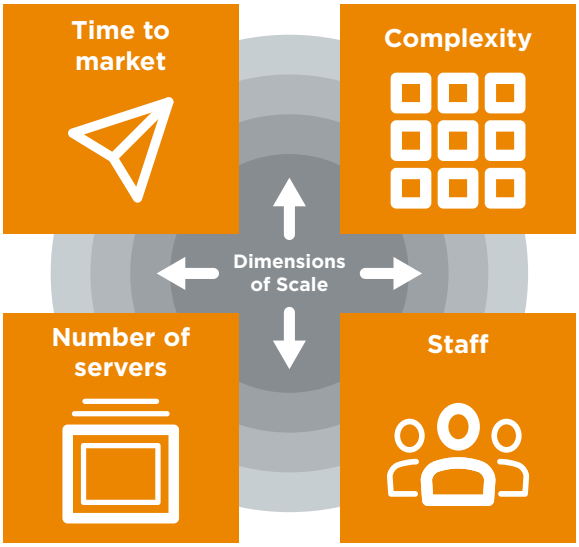
Fast response to consumer demand requires a technology stack that can react quickly to changing business needs while maintaining its resiliency and reliability, and it requires the ability to deliver new application features just as quickly.

Defining DevOps

Web innovators discovered that to promote agility and speed, IT and development must work together to build a technology stack that can safely support rapid deployments and dynamic workloads.

These innovators combined software developers and IT professionals into unified teams aligned around shared business goals. These teams use a workflow that relies, technologically, on automation and socially, on collaboration. This new way of working makes it possible for companies to develop infrastructure and applications that respond quickly to change while remaining stable and reliable. The new process also enables much faster and more reliable delivery of application features.

As the new way of working disrupted the conventional thinking that had shaped IT operations in the past, DevOps became the accepted name used to describe it. Today, DevOps can be thought of as the tools, processes and culture that are central to becoming a software-led organization.



DevOps Patterns

One of the goals of DevOps is to create a compute environment that supports rapid deployment, and has the ability to react swiftly and effectively to changes in business requirements. These attributes scale even as the overall system becomes larger and more complex.

There are some general patterns that characterize how DevOps practitioners achieve this goal. Here are some of the patterns.

DevOps practitioners focus on the digital customer experience. All the technology they invest in and the innovations they create are a means for delivering a great customer experience. These customers can exist either outside or inside of the firewall.

> We start with the customer and work backwards.[1]
>
> – Jeff Bezos, Amazon.com

DevOps practitioners collect data. Web innovators are great believers in data and collect it on everything they can as often as they can. They improve their processes based on evidence provided by the data they collect. The data helps them understand what's working and what isn't.

DevOps practitioners invest in technical innovation. Successful online companies encourage innovation, and the investments don't have to be large.

Intuit is a great example of a company that encourages innovation. Intuit gives employees 10% of their hours as unstructured time. The legal department created a toolkit that lets product managers try new business ideas without needing to talk to legal. Intuit's IT department leveraged a DevOps workflow to accelerate the setup time for test environments for new web products from two months to two hours.

DevOps practitioners use open source software and dynamic infrastructure. A reliance on open-source software is typical in DevOps environments.

DevOps practitioners also rely on dynamic infrastructure where resources are provided on demand to the company's business units, with service-level agreements that guarantee some level of quality in areas such as availability, performance and security. Individual departments needn't be responsible for planning capacity; instead, capacity planning occurs at the corporate level and is aligned to the needs of the business.

The incentive to use open source software and commodity hardware, and to provide infrastructure as a service (IaaS) is not necessarily cost—it's the ability to customize and control the technology stacks and respond quickly to business needs.

DevOps practitioners tend to use service-oriented architectures. Factoring applications into stable, independent services that use web protocols and associated architectural patterns is very common in the DevOps world. Each service can be independently implemented, deployed, and scaled. As a result, teams become aligned to business functions rather than divided into silos that isolate technical specialties from each other and put drag on the ability to deliver value.

DevOps practitioners improve through multiple iterations. Amazon, Google and Facebook don't wait until they've built what they think is the perfect product only to find out that what they've done isn't what the customer wants. Instead, they start with a minimal implementation and build it incrementally. They use A/B testing to find out what works and what doesn't. In 2011, Google ran more than 7,000 A/B tests on its search algorithm. Amazon.com, Netflix, and eBay are also A/B advocates, constantly testing potential site changes on live users.

DevOps practitioners avoid silos through transparency. Silos are not just organizational divisions but informational ones as well.

When Nordstrom used DevOps principles to build a continuous delivery pipeline, they put developers, web site operations engineers, QA engineers, and configuration management together to form a single team. The team combined expertise across multiple technical disciplines and retained focus and accountability by holding weekly demos. The end result of all that transparency is that now people all over the company, even people in the finance department, email senior vice presidents, asking how can they help with continuous delivery.

Increased communication has many obvious benefits. Within a team, members quickly learn about proposed changes and any problems that exist. Involving a number of different stakeholders, such as product managers, members of the sales force, and consulting can bring in valuable information from areas outside the team members' areas of expertise. Making information available to everyone in the company gives employees a sense of ownership and participation.

DevOps practitioners deploy software very quickly. Part of the reason DevOps teams move so quickly comes from their belief that failure is expected in a culture that innovates and moves rapidly. However, they don't want to fail after having spent months on a product. Instead, they use a very fast stream of incremental releases so that, when there is a failure, it's easy to correct and their investment is small. A flexible, managed infrastructure allows them to quickly put together prototypes and test them. Automated processes, such as continuous integration and continuous delivery, make deployment and change management faster and more reliable.

DevOps practitioners build compliance into the software deployment pipeline. Instead of being tacked on at the end of the production process, compliance is embedded into the software production line. Compliance at velocity uses extensive automation to increase velocity and accuracy.

For example, GE Capital (GEC) realized that, to remain competitive, it needed to streamline the way it developed, delivered and maintained software. At the same time, GEC operates in a highly regulated environment and is obligated to comply with many requirements. GEC's challenge was one that many large enterprises face: How do you remain compliant and still operate at velocity?

"Steve Blank and Eric Ries, both serial entrepreneurs, have studied what allows businesses such as Google to succeed in today's quickly changing world. They found that...the companies that succeed in dynamic marketplaces are those that rapidly develop products with minimal planning and commitment of resources."[6]

For GEC, the answer was DevOps. To ensure that compliance was an integral part of this new software delivery process, they brought the regulatory, compliance, governance, and security teams in early to take tollgates out and get rid of manual processes so they could deliver at velocity and at the same time remain compliant and secure.

DevOps practitioners use version control. Version control systems give transparency to all aspects of orchestration, configuration and deployment. With version control team members can review the history of all change sets at any time.

DevOps practitioners rely on automation. An automation platform gives you the ability to describe your entire technology stack as executable code. DevOps practitioners use automation to:

- Standardize development, test and production environments.
- Effectively deploy and manage cloud resources.
- Eliminate error-prone, time-consuming manual tasks.
- Improve cooperation between development and operations.
- Implement automated release pipelines.

What is Automation?

Automation gives you immediate access to the same patterns of success that the web innovators had to develop themselves.

When you automate your technology stack, you describe it with executable code. For example, here's how you can use the Chef automation platform to ensure that the Apache web server package is installed and being run as a service on a RHEL or CentOS server.

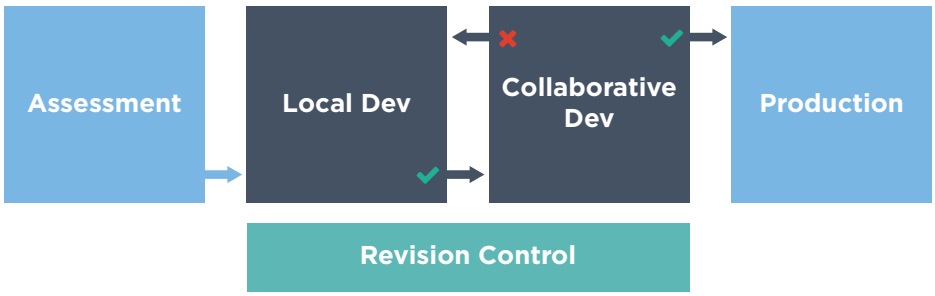
```
package 'httpd'

service 'httpd' do
  action [:enable, :start]
end
```

This code tests to see if Apache is installed and running. If not, it installs the Apache package, called `httpd`, enables the service and starts it when the server boots. The code is also readable by humans so that team members have a shared understanding of the system's intended state.

If you're not using an automation platform, you might perform these tasks by hand. If it's a manual task, imagine what it would be like if you had to do the same procedure for 50 servers, 500 servers or even 50,000 servers. With automation, you simply run the code as often as needed.

Automation platforms have significant advantages over isolated scripts that have traditionally been used for system administration tasks. Platforms like Chef take care of many of the complexities of configuring a server for you. Chef recipes have a global view of your network because they interact at run time with the Chef server, which is important when accounting for dependencies between network components. Standalone scripts can only give you a piecemeal view.

Full-stack automation can revolutionize the way your teams work together. With it, the same practices you follow to ensure the quality and manageability of your applications can now be applied to your infrastructure and all of the services in the stack. You can faithfully encode your entire technology stack in version control, and you can test it. If you lose part of your infrastructure or even all of it, you can recreate it by rerunning the code that describes it.



Benefits of automation

Automation enables *velocity*, *scale, consistency* and *feedback*. All of these qualities are of a piece. Any one depends on the other three—for example, you can't scale unless you are able to quickly add servers with consistent configurations. You can't get feedback automatically without being able to support different real-time testing and monitoring techniques. You can't respond to feedback effectively unless you have a high-velocity way to deliver incremental changes safely.

VELOCITY

Automation increases velocity in many ways. Simply replacing manual procedures with automated ones makes infrastructure management more efficient. However, as your use of automation becomes more sophisticated, you'll find that you'll markedly increase your deployment rate and the ease with which you manage all your resources, both on premises and in the cloud. Automation makes techniques such as A/B testing possible. You can quickly find out what works for your customers before investing huge amounts of time and money. Quick response to changing business needs is essential.

SCALE

Automation allows you to scale up (or down) in response to demand. Automation is a critical component of any strategy that requires dynamic provisioning of infrastructure at scale. Automation enables elastic scale, whether you're operating on premises, in the public cloud, or in a hybrid environment.

CONSISTENCY

Automation ensures consistency across your network. Consistency means conformance to your business's policies. An automation platform checks to make sure that each server is within policy and corrects it if it isn't. In other words, a good automation platform must make it easy to prevent configuration drift over time.

Consistency makes infrastructure more robust and reduces risk in many ways. The immediate benefit is that you have a standardized process for provisioning servers.

A consistent environment makes it much easier to migrate applications to the cloud. Consistency gives you control, and control reduces risk. With automation, moving legacy applications to the cloud is an orderly process.

> When you represent [infrastructure] in code, you can version that code, and you can say, "This is what the machine looks like today and this is what it looked like last week, and this is when somebody changed it." When you have that, you can almost print these new machines like you have a factory press. You just put the code in and Chef takes care of it. Chef prints out brand new machines for you, faster than ever.[7]
>
> —Jamie Winsor, software engineer, Riot Games

Requirements for an automation platform

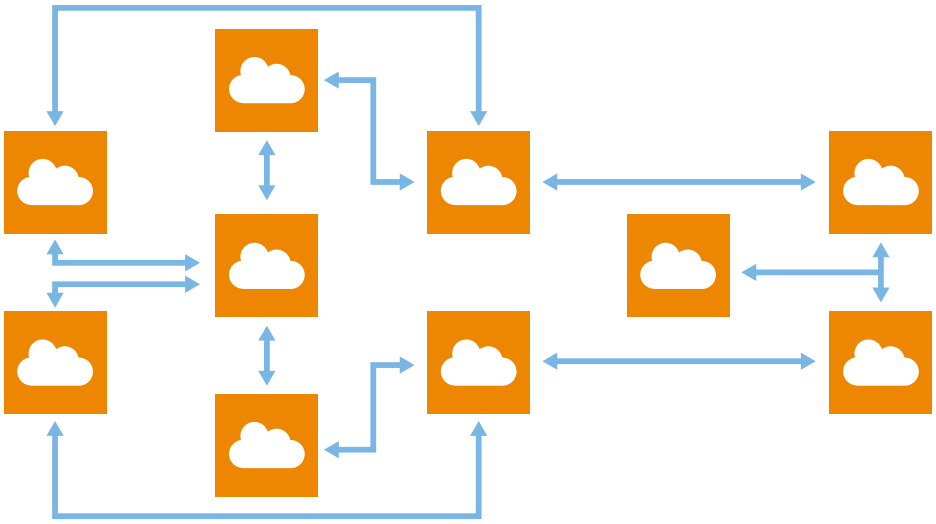
To provide the foundation for building and managing infrastructures according to the principles of DevOps, a good automation platform has these essential characteristics:

- It creates a dependable view of your entire network's state.
- It can handle complex dependencies among the nodes of your network.
- It is fault tolerant.
- It is secure.
- It can handle multiple platforms such as RHEL, Ubuntu and Windows Server, as well as legacy systems.
- It can manage cloud resources.
- It is the basis for an efficient workflow.
- It provides a foundation for innovation.

Let's examine these points in more detail.

Your automation platform should create a dependable view of your network. A good automation platform knows the state of your entire network at any given time. You need a global view of your network. Scripts can't provide this capability.

Your automation platform should handle complex interdependencies. Most infrastructures have many dependencies between servers. For example, a load balancer needs to know when a new application server is available. Isolated scripts can't handle complex dependencies that require distributed coordination.



Implementing distributed coordination requires specialized techniques that take a holistic view of the network. It's more than just running scripts or deploying "golden images."

A good automation platform will allow the network to *converge* to its desired state over time and provide search-based configuration that allows nodes to query the automation platform for information about other nodes in the network. This is sometimes called *policy-based convergence*.

Your automation platform should be scalable. Infrastructures tend to become larger and more complex over time. To ensure scalability, a good automation platform will have a distributed, rather than a centralized architecture. With a centralized architecture, most of the work occurs on the server, which can become a bottleneck as networks grow. With a distributed architecture, the work occurs on the nodes, and a node only has to take care of its own configuration.

Your automation platform should be fault tolerant. A good automation platform is able to recover when network connections go offline or when a system needs to be rebooted. It should also be able to handle errors and unexpected conditions. A good automation platform will eventually converge to the desired state, even if faults occur.

Your automation platform should be secure. A good automation platform ensures that communications between the server and the nodes are secure. It enables granular control over who can access different resources.

Your automation platform should handle multiple platforms. Many infrastructures include multiple operating systems. For example, there may be Windows, AIX and Linux machines in the same network. A good automation platform supports heterogeneous networks.

Your automation platform should handle legacy systems. Most infrastructures include legacy systems that don't fit any standard configuration model. An automation platform should be extensible and not just a set of fixed capabilities.

Your automation platform should be able to manage cloud computing environments. A good automation platform is cloud capable and provides the structure and consistency needed to make moving legacy workloads to the cloud a low-risk operation.

Your automation platform should automate your workflow, not just your infrastructure. DevOps is a new way of working together safely and at high velocity. Your automation platform needs to understand the demands of the DevOps workflow and help you replace cumbersome change management processes with proven, modern approaches taken from the experience of web innovators.

Your automation platform should provide a foundation for innovation. Even if you're automating basic configuration tasks now, what do you want to be doing a year from now, or five years from now? You need a platform that meets your present needs but won't limit you as you grow.

Why Chef?

Chef[8] is the automation solution for the DevOps workflow. Chef was born with the DevOps movement, and experience with DevOps thinking and best practices have been distilled into every aspect of Chef.

Chef is IT automation platform for DevOps

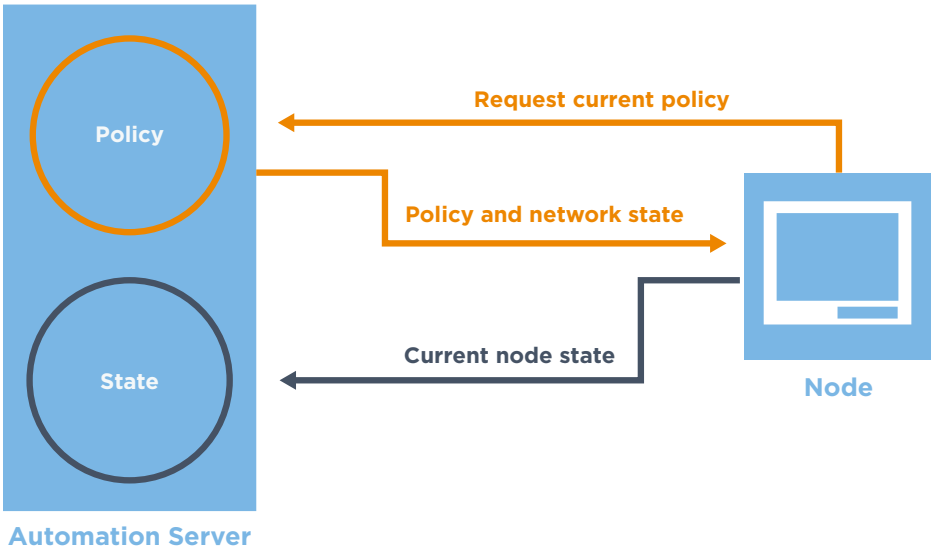
Only Chef is a dynamic, policy-based automation platform that securely distributes intelligence across the entire network. What does this mean? It means that

- Chef has a unique ability to scale, from start-ups to Facebook to GE.
- Chef has a unique ability to ensure consistency in complex, highly dynamic environments.
- Chef is fault tolerant.
- Chef grows with you. When it comes to solving configuration and automation challenges, Chef makes the easy things easy and the hard things possible.

CHEF IS HIGHLY SCALABLE

Chef is constructed so that most of the computational effort occurs on the nodes themselves rather than on the Chef server.

With Chef, the intelligence about the desired state of the network is distributed across the network itself. Each node of the network periodically executes the current instructions from the Chef server. This iterative process ensures that the network as a whole converges to the state envisioned by business policy.



Chef is the only automation platform that uses a fully distributed approach, and this has some implications that make it uniquely suited for the massive scale of today's applications. Chef's unique ability to scale is one of the reasons Facebook uses Chef for its production systems.

> There are three dimensions of scale we generally look at for infrastructure – the number of servers, the number of different configurations across those systems, and the number of people required to maintain those configurations. Chef provided an automation solution flexible enough to bend to our scale dynamics without requiring us to change our workflow.[9]
>
> —Phil Dibowitz, Production Engineer at Facebook

CHEF CAN HANDLE COMPLEX, HIGHLY DYNAMIC ENVIRONMENTS

Chef has a unique ability to ensure consistency in complex, highly dynamic environments. This is a weak spot of a centralized approach where a server blocks while waiting for a response from a node, or of an approach where logic executes on the server.

Chef handles bidirectional dependencies. If a network uses database replicas, each replica must know about the others in order to remain in sync. Symmetric dependencies such as these create a sequencing problem that can only be solved by using policy-based convergence. Full configuration doesn't occur in a single step, but the network as a whole eventually converges to its desired state.

Chef handles reboots and network resets. Centralized approaches that rely on long-lived network connections break down when the networking service goes offline or the system needs to be rebooted as a part of the requested operation. A more distributed approach, where the node itself initiates contact with the server, allows the node to update state after coming back online. The Chef server can orchestrate a complex series of operations, even when nodes under management require network resets or must reboot as part of the process.

Chef is fault tolerant. When intelligence is distributed to the nodes, appropriate recovery measures can be taken when an error or unexpected condition occurs. It is more difficult for a centralized server to respond in this case.

> By bringing in Chef, we were able to automate a very heterogeneous infrastructure that included both legacy and new applications and we were able to open up some interesting career paths for our engineers. We have hardcore UNIX engineers now happily automating Windows infrastructure because they can do it through code.[10]
>
> —Rob Cummings, Infrastructure Engineer, Nordstrom

Chef is secure on every type of network. Chef uses SSL to ensure that a Chef server responds only to requests made by trusted users. When a node is configured to run the Chef client, bidirectional validation of identity occurs between the Chef server and the newly added node. This makes Chef suitable for managing nodes on every type of network, even public networks.

Chef supports multiple platforms and legacy systems. Chef supports many flavors of Linux and Unix, as well as Windows Server. For example, Chef includes support for Microsoft's Desired State Configuration (DSC) PowerShell extension. Chef also supports containers. With Chef, engineers can use the same skill set to manage every platform in your network.

Chef comes with a large number of pre-defined building blocks, called resources, which describe pieces of infrastructure, such as files, templates, and packages. The Chef community has also written many collections of configuration instructions called cookbooks, which cover many situations.

However, if you need to write your own configuration steps for a particular system, you can do it with Chef. Chef isn't constrained by a limited, domain-specific language. You have the flexibility you need to describe any piece of infrastructure you have.

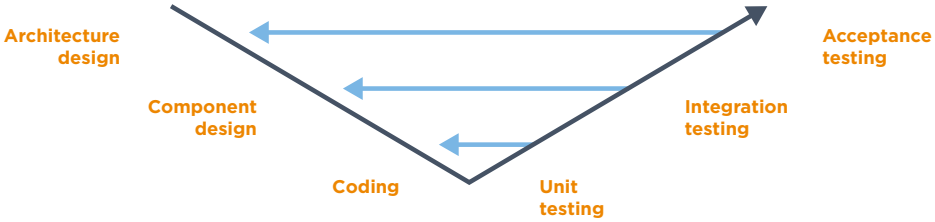
Chef can manage cloud resources. If you're already in the cloud or thinking about moving some servers there, Chef has proven to be a great way to manage your resources. For example, Cycle Computing uses Chef to manage tens of thousands of public cloud nodes. A large percentage of customers use Chef to automate and manage cloud resources.

Chef enables test-driven infrastructure

When your infrastructure is described as code, you can treat that code just as you would your application source code. For example, do you have unit tests for your applications that are initiated automatically whenever there is a check-in to your version control system? You can now do the same with your configuration code.

With automated testing, you will catch problems earlier, before they impact your release cycle. The earlier you catch a problem, the easier and less expensive it is to fix, and this is why testing is such an important part of DevOps practice.

The V-diagram, common in software engineering, illustrates this.



As you can see, each kind of testing activity (on the right side of the V) checks a particular phase of development (shown on the left side). The cost of rework rises as defects are discovered later in the project. It's better to begin testing at the vertex of the V, with unit tests. Catching a defect during a unit test is much easier than trying to fix it when it's being tested with other components that might make it difficult to discover where the actual problem lies.

> We're looking at cloud architecture. We're looking at public cloud, we're looking at private cloud. We want to do some completely different things that we haven't been able to do before. The only reason that I'm able to consider those is because of what we did with Chef. It's now opened a completely new capability that I hadn't foreseen. I view Chef as the tool that has had the single biggest impact in our transformation.[11]
>
> —John Esser, Director of Engineering Productivity and Agile Development, Ancestry.com

The practice of automatically testing your infrastructure code is called *test-driven infrastructure*. Testing the code that provisions and configures your infrastructure gives confidence that your infrastructure will behave as it should when put into the production environment.

Chef has a strong commitment to test-driven infrastructure as part of the DevOps workflow. In fact, it is the only company that provides commercial support for a full suite of tools for test-driven infrastructure.

The following diagram shows how you can use the different tools Chef provides to test your code at all stages of development.

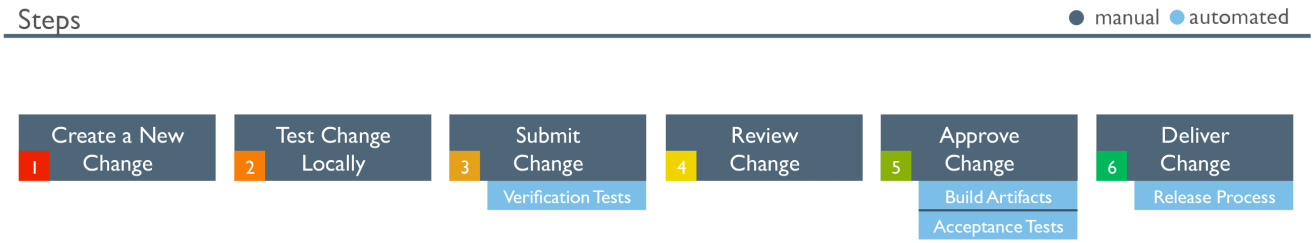
Chef automates enterprise change management

The DevOps workflow allows you to safely and rapidly deploy changes to applications and infrastructure. However, it can be challenging to implement the actual pipeline that moves your code from development to production. Not only must the pipeline's design address technical challenges, but it should also encourage practices that support a DevOps workflow. Those practices are a large part of the reason that DevOps make it possible to quickly move code out to production and realize its value.

> Code has ... given us a single way to communicate. Before we had different groups operating with different tools, and different mindsets in how they approached things. By distilling it all down to code, we're able to leverage the same practices among different groups. It allows us to be more agile, move faster and respond when the business needs us to respond.[12]
>
> —Rob Cummings, Infrastructure Engineer, Nordstrom

Integrating DevOps practices with automation is the foundation of a new approach to enterprise change management. Chef Delivery, a recent addition to the Chef platform, is an example of such a system. Its design is based on Chef's years of experience with its enterprise and big web customers. You use Chef Delivery for both your infrastructure and application code, giving your operations and development teams a common platform for developing, testing and deploying cookbooks and applications. Chef Delivery also incorporates DevOps best practices, such as using source control and automated testing, into its design.

Here is the Delivery workflow:



The first step of the workflow is to create a change on a developer's workstation. The next step is to test the change locally using the automated tools described above in the section "Test-Driven Infrastructure." While it is still a new practice to many people, local testing is an excellent way to discover bugs early in the development process, when they are easier and cheaper to fix.

Once a change passes local tests, the person making the change commits it using the revision control system and submits the change for review using Delivery. At this point, Delivery runs automated verification tests, typically the same tests that should have run locally. Delivery runs them again as a safety check, to make sure the code does what it should.

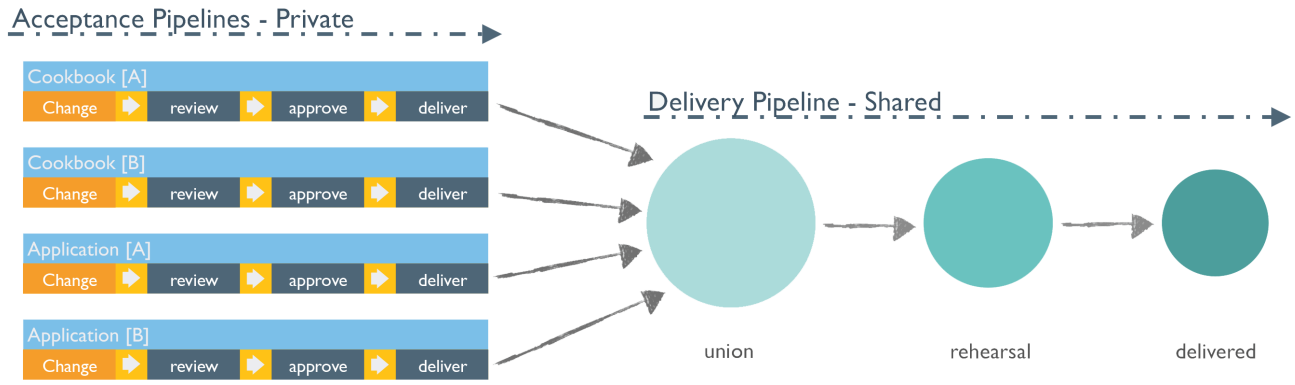
Once the tests pass, the change is ready for review. Code review is an important best practice that Delivery encourages. Teams can create any number of policies over how the code is reviewed and who participates. Code review becomes the cornerstone of change management.

Once the change is approved, Delivery initiates an automated build process to create an *artifact* (packaged set of files). If, for example, the change is a cookbook, Delivery creates a new cookbook with a new version number. If the change is to application code, Delivery might, for example, build a package in a format like WAR (Web Application Archive).

After the build stage, the artifact moves to the Acceptance stage. Delivery runs additional automated tests and optionally allows for manual testing to occur. When all tests of the Acceptance stage pass, the change is ready for delivery.

When to release the change is a business decision. When that decision is made, an authorized person presses the "Deliver" button in the Delivery user interface. Chef Delivery then promotes the artifact to a series of environments that comprise the automated release process.

Here is a diagram of the Chef Delivery pipeline. You can see that it is made up of two parts: per-project acceptance pipelines and a shared delivery pipeline. An acceptance pipeline encompasses all the workflow steps up to when someone releases an artifact for delivery. The artifact then moves on to the Union, Rehearsal and Delivered stages of the shared pipeline. These make up the release process.



For example, you might have an application and its associated infrastructure that you want to deploy. The infrastructure is managed by a Chef cookbook. The application is might be composed of microservices that communicate with one another using web protocols. Each microservice can be tested independently in its own acceptance pipeline.

At the end of an acceptance pipeline, when someone decides to deliver the artifact, it moves into the Union stage, where it is tested with the latest versions of the other components that make up the application and its supporting infrastructure.

Once the application and infrastructure pass the Union stage, they move on to Rehearsal. More tests are run and, assuming they pass, the components move to Delivered.

Delivery includes a graphical user interface that gives visibility into the entire process. For example, you can tell at a glance which organizations include which projects. Dashboards let you track each change and see its status as it moves through the pipeline.

Delivery extends the capabilities of Chef server automation to provide change management and rapid deployment for the entire technology stack. With Delivery, companies have a well-defined workflow that moves changes, features and new services to production quickly and safely.

Summary

No matter what your plans are for your company, Chef can help you create and maintain the infrastructure that will make those plans possible. Chef gives you end-to-end control over application deployment, beginning with the developer workstation and ending with the production servers. Chef's scalability means that no matter how large and complex your network becomes, Chef can handle it. Chef's flexibility means that you can describe any configuration you have.

Companies such as Intuit, Nordstrom, and Disney use Chef to spur innovation, create new offerings, and speed deployments.

Here are the key points of this paper.

- Web innovators such as Amazon, Google, Facebook and others have set customer expectations for what the digital experience should be. These customers can be either outside or inside the firewall.
- Any company that wants to remain competitive must follow the lead set by the web innovators, who have redefined the customer experience.
- To provide a fast, responsive, personalized experience to their customers, web innovators have adopted practices that, collectively, are known as DevOps.
- DevOps is for any business that wants to move at velocity.
- The technical foundation of DevOps is automation. There are also important cultural and organizational aspects of DevOps.
- Automation supports a workflow that provides speed, scale and consistency.
- A good automation platform is scalable, can handle complexity, and has features that let enterprises plan for the future.
- Chef's distributed architecture lets it securely scale to any infrastructure, no matter how large or complex.
- Chef supports multiple platforms, including Windows, AIX, Linux, and containers.
- Chef is the only automation platform with commercial support for automated testing tools for IT automation.
- Chef helps you through all aspects of full-stack application and infrastructure deployment, beginning with a developer's workstation and ending with the production servers.
- Chef Delivery is a full-stack deployment pipeline whose design reflects Chef's experience with its customers in how to deploy applications and infrastructure quickly and safely. It integrates automation with DevOps best practices and modernizes enterprise change management.

Notes

1. http://www.slate.com/articles/news_and_politics/newsmakers/2009/12/we_start_with_the_customer_and_we_work_backward.html
2. http://www.forbes.com/sites/bruceupbin/2012/09/05/four-ways-intuit-keeps-new-ideas-flowing/
3. http://www.wired.com/2012/04/ff_abtesting/all/
4. https://www.youtube.com/watch?v=Ot5H2KfWAxI
5. http://www.forbes.com/sites/benkepes/2015/04/17/ge-capital-and-its-devops-reinvention/
6. *Fail Fast, Fail Often: How Losing Can Make You Win*, Ryan Babineaux and John Krumbolz
7. http://www.youtube.com/watch?v=URi4_DLrFxQ
8. http://www.chef.io
9. http://www.youtube.com/watch?v=YA8p65qcD0I
10. http://www.youtube.com/watch?v=XqFrvYJ6IGc
11. http://www.youtube.com/watch?v=5enyC4lyPWE
12. http://www.youtube.com/watch?v=XqFrvYJ6IGc