Automation
and
Web-Scale IT
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.

Web innovators, such as Amazon, Google, Facebook and others, have set the pattern for this transformation. Web innovators developed a set of practices, summarized by the term web-scale IT, which created infrastructures that were extraordinarily fast and scalable. They raised the bar for speed and scale of IT operations, and for the breadth of services that can be delivered online. As a result, all enterprises with a digital presence are under pressure to adopt practices established by web innovators, no matter what industry they are in. Today, every business is a digital business.

Enterprise IT, once considered a back office support function, is becoming the front office connection to customers and the linchpin of a mission-critical transformation. Now, all enterprise IT organizations must adopt approaches that enable resilient, flexible infrastructure. This infrastructure can support the kind of speed, scale and consistency that business demands in the new era of web-scale IT.

Enterprises also have a challenge that the web innovators did not. They have legacy workloads to consider. Migrating these workloads to the new resilient, flexible infrastructure must be done with an automation platform that reduces risk and ensures stability.

This paper explains the attributes of web-scale IT and describes how choosing the right IT automation platform is critical to every business's journey toward web speed and web scale.
Every Business Is Software

Today, every business is a digital 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, Facebook and others constantly raise consumer expectations. Any business that wants to remain competitive must emulate the patterns developed by the web innovators, which all focus on enhancing the digital experience. Following these patterns requires that companies be able to quickly respond to customer demands with the software that provides the experience customers want.

Fast response to consumer demand requires infrastructure that can react quickly to changing business needs while maintaining its resiliency and reliability. The name for this type of infrastructure is web-scale IT.

By 2017, Web-scale IT will be an architectural approach found operating in 50 percent of global enterprises, up from less than 10 percent in 2013.¹

—Industry analyst Gartner, Inc.

Defining Web-Scale IT

The term web-scale IT is used by technology analyst Gartner, Inc. as a comprehensive name for the practices and architecture developed by web innovators such as Amazon, Google, Facebook and others.

Scale in this sense has several dimensions. One dimension is size, which is often measured by the number of physical and virtual servers deployed. A second dimension is complexity, which is defined in terms of the sophistication of the technology stack and the nature of service interdependencies. The number of people required to maintain and support infrastructure is a third dimension of scale.
There are also dimensions for agility and speed. Speed is defined by time to market, and agility is the ability to respond quickly as business needs change. It’s important to remember that web-scale IT also means web-speed IT. Business agility is a central component. Companies of all sizes can adopt the web-scale principles. Web-scale IT is characterized by:

- The use of open-source software and commodity hardware to create infrastructure that can be completely controlled.
- The factoring of applications into resilient, independent services that use web protocols and architectural patterns.
- The elimination of the traditional silos of dev and ops.
- The adoption of workflows that enable agility and speed.

Use of open-source software and commodity hardware

Web innovators carefully control their hardware and software stacks. They begin with open-source operating systems and commodity hardware. Once this foundation is in place, they write their own software that gives them complete control over their environments.

For example, web innovators have been able to implement infrastructure as a service. 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. This example shows that the incentive to use open source and commodity hardware is not cost—it’s the ability to customize and control the technology stacks.
In addition to allowing infrastructure on demand, using open source software and commodity hardware also reduces risk. There’s little reliance on third-party vendors of specialized hardware and software. Instead of buying redundant power supplies and disks, web innovators achieve high uptime with fault-tolerant software techniques running on commodity hardware. Systems as a whole are designed to be resilient in the face of hardware failures that occur in individual servers. Such systems are characterized by larger numbers of smaller, more specialized servers.

Web-oriented service architecture
Web innovators use a service-oriented architecture based on web protocols and architectural patterns. Think of a service as a software component that performs a particular function, such as authenticating a user’s credentials or managing a user’s online shopping cart.

These services are loosely coupled and are designed to be APIs that encourage sharing and reuse. Loosely coupled means that the services operate independently of each other. Loosely coupled web services can be implemented using different languages, runtime environments and operating systems. This level of operational independence allows systems to be designed for maximum uptime and reliability.

Services built for reuse and sharing allow developers to create new applications from existing web-based services.

Integration of dev and ops
Web innovators know that to promote agility and speed, IT and development must work together to build an infrastructure that is resilient and can support rapid deployments and dynamic workloads.
The web innovators often combine software developers and IT professionals into unified teams aligned around shared business goals. This practice is called DevOps. These teams focus on automation and operational efficiency. The trend toward DevOps is disrupting much of the conventional thinking that has shaped IT operations in the past. With its emphasis on automation, communication and trust, DevOps has made it possible for companies to develop infrastructure that responds quickly to change while remaining stable and reliable.

**What Web-Scale IT Enables**

All the practices developed by Web innovators aim to create an infrastructure that supports rapid deployments, has the ability to react swiftly and effectively to changes in business requirements, and is resilient. These practices scale even as the infrastructure becomes larger and more complex.

There are some general patterns that characterize how web innovators operate. These patterns for agility and speed are possible because web innovators know how to leverage infrastructure rooted in web-scale IT principles and architectures. Here are the patterns.

**Web innovators 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. Remember, these customers can exist either outside or inside of the firewall. For example, a retail business has customers outside the firewall who buy its products. Any business might have customers within the firewall. For example, developers might want to be able to order the environments they need by selecting them from a service catalog.

**Web innovators 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.

**Web innovators 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 tool kit that lets product managers try new business ideas without needing to talk to legal. Intuit’s IT department leveraged web-scale practices to accelerate the time it took to set up test environments for new web products from two months to two hours.

**Web innovators 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.

**Web innovators avoid silos through transparency.** Silos are not just organizational divisions but informational ones as well. For example, devs know all about the code in their applications but don’t understand the production environment in which it will run. Operations knows little about the application and everything about the company’s infrastructure and release environment, which may not match what the devs use. The consequence is a release process that’s slow and painful. When the two groups share information, the release process is much more efficient.

-- Jeff Bezos, Amazon.com

---
When Nordstrom wanted to build a continuous delivery platform, they put developers, web site operations engineers, QA engineers, and configuration management together. They formed a real team even though they were from different silos. They also held 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.\(^5\)

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.

Web innovators deploy software very quickly. Part of the reason web innovators 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, the large web innovators 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. Again, a flexible, managed infrastructure allows them to quickly put together prototypes and test them.

**Web innovators use automation.** Automation underlies all the patterns and practices developed by the web innovators. An automation platform gives you the ability to describe your infrastructure as executable code. It provides the means for:

- Eliminating error-prone, time-consuming manual tasks.
- Standardizing development, test and production environments.
- Building automated release pipelines.
- Improving cooperation between development and operations.

Automation solves immediate problems, such as repetitive configuration tasks and also gives you the power to create and manage an infrastructure that meets the demands of today’s digital economy. With automation, you can move from infrastructure that is inconsistent and static to one that is consistent and elastic.

### What is Automation?

Automation is a major underpinning of web-scale IT. Using an automation platform gives you access to the same patterns of success that the web innovators had to develop themselves.

When you automate your infrastructure, you describe all the tasks you want to perform as executable code. For example, here's how you can use the Chef automation platform to install an Apache package and run it as a service on a RHEL or CentOS server.

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

This code installs the Apache package, called httpd, starts the service and enables it when the server boots.

“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.”\(^6\)
If you’re not using an automation platform, you may perform this task 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 once and all the servers are configured.

You might try an approach based on scripting, but automation platforms are orders of magnitude better than isolated scripts. Platforms like Chef take care of many of the complexities of configuring a server for you. Also, automation platforms can give you a global view of your network, which is important when accounting for dependencies between network components, while scripts can only give you a piecemeal view.

Finally, consider the advantages of describing your infrastructure as code. The same practices you follow to ensure the quality and manageability of your applications can now be applied to your infrastructure. You can put it in version control and you can test it. If you lose part or even your entire infrastructure, you can recreate it by rerunning the code that describes it.

Benefits of automation

Automation enables speed, scale and consistency. All of these qualities are of a piece. Any one depends on the other two—for example, you can’t scale unless you are able to quickly add servers with consistent configurations.

**SPEED**

Automation increases speed 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. An automated infrastructure 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.
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. Once you've expressed your infrastructure as code, you can test it to make sure it performs as you intended. You can also save it in version control. You'll be able to keep track of changes and, if one version doesn't do what you thought, you can rollback to the previous one.

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.

Another advantage is that, because your infrastructure is code, you can always recover it. Even if disaster strikes you can recreate your entire infrastructure, from the bare metal up.

**Defining the automation platform**

To provide the foundation for building and managing web-scale IT infrastructures, 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 provides a foundation for innovation.

Let's examine these points in more detail.

**Creates 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.
Can 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.

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.

Is 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.

Is 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.

Is 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. Additionally, it allows you to control security on particular nodes.
Can 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.

Can 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.

Can manage cloud computing environments
Are you already in the cloud or considering the advantages of migrating some of your applications there? You need the same capabilities for managing cloud resources as for your on-premises resources. 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.

Provides a foundation for innovation
A good automation platform lets you plan for the future. 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 an automation platform that can handle everything from configuring a server to building a continuous delivery pipeline. In other words, you need a platform that won't limit you as you grow.

Why Chef?
Chef® is the automation solution for web-scale IT. 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 distributes intelligence across the network
Chef is constructed so that most of the computational effort occurs on the nodes themselves rather than on the Chef server. In this highly distributed architecture, the Chef server acts as a trusted repository that stores policies and data about the state of the network. It doesn't configure the nodes. Instead, it simply sends them the instructions they should follow.

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.
If Chef were a centralized system, then the Chef server would be responsible for ensuring that all the nodes were in compliance. Given how infrastructures can grow, both in scale and complexity, a centralized server can quickly become a bottleneck. Chef is the only automation platform that uses a fully distributed approach, and this has some implications that make it uniquely suited for web-scale IT.

Chef is highly scalable
Chef’s unique ability to scale is one of the reasons Facebook uses Chef for its production systems. Such a high level of scalability would simply not be possible if Chef did not distribute most of the work to the nodes.

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 a distributed approach where logic executes on the server.

Here are some examples of how Chef handles these types of environments.

**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.

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. —Phil Dibowitz, Production Engineer at Facebook
It can take several iterations for a server to comply with policy. (An iteration occurs when a node and the server communicate with each other to see if the node is compliant.) Full compliance may not occur in a single iteration. Even in complex situations, with many dependencies, Chef ensures that the network as a whole continually moves closer to the state defined by business policy.

**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.

**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.

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 bug 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.

The practice of automatically testing your configuration code is called test-driven infrastructure. Testing the code that provisions and configures your infrastructure has the same benefits as testing your application code. You’ll find errors earlier, which means they’ll be easier (and less expensive) to fix. You’ll also have confidence that your infrastructure is behaving as it should. A major consequence is that problems with your infrastructure won’t delay your deployments.

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

When you install Chef, you get a set of tools that support automated testing. For example, the tool called foodcritic is a static analysis tool that checks that your Chef cookbooks are syntactically correct. The tool called chefspec lets you unit test your Chef code. Unit tests check small pieces of code in isolation and ensure that they do what they should.

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

---

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.

—Rob Cummings, Infrastructure Engineer, Nordstrom
Chef Handles Diverse Environments

Chef has other advantages that make it the automation platform for web-scale IT.

Chef is secure

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 has support for 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 uses the Ruby programming language, which means it 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, Splunk uses Chef to manage its cloud offering, Splunk Storm. Cycle Computing uses Chef to manage tens of thousands of nodes. A large percentage of customers use Chef to automate and manage cloud resources.

Chef provides a foundation for innovation

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 laptop 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.

—Alex Munk, Product Manager at Splunk
Summary
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 an approach toward infrastructure that is called web-scale IT.

• Web-scale IT is not only for large enterprises. It’s for any business that wants to increase the speed with which it delivers its services.

• Web-scale IT depends on open source software, commodity hardware, and a service-based architecture.

• Web-scale IT encourages the adoption of DevOps.

• The foundation of any web-scale IT infrastructure is automation.

• Automation 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 application deployment, beginning with a developer’s laptop and ending with the production servers.
NOTES
5. https://www.youtube.com/watch?v=Ot5H2KfWAXI
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
11. http://www.youtube.com/watch?v=XqFrvYJ6IGc
12. http://www.youtube.com/watch?v=XqFrvYJ6IGc