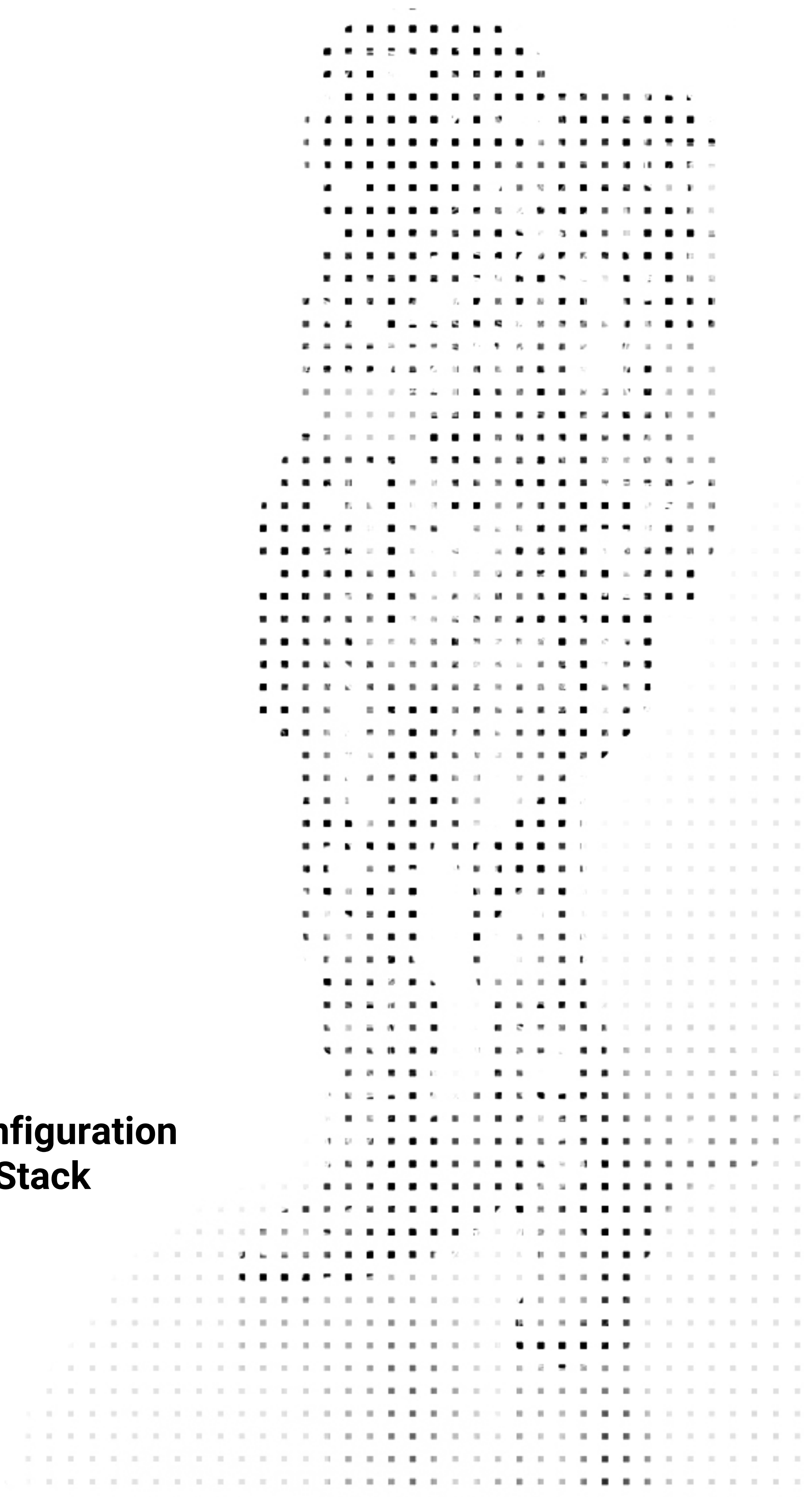
SALTSTACK.

Beyond application configuration management with SaltStack

By David Strom



When it comes to building online applications, you can build them with old tools and attitudes or with new methods that are purpose-built for solving today's problems and infrastructures. Back in the days when mainframes still walked the earth, setting up a series of online applications used some very primitive tools. Now, while we have more integrated development environments that embrace SaaS apps running in the cloud, it is more of a half-hearted acceptance. Few tools really have what it takes for handling and automating online apps.

Today's IT environments are in a constant state of flux and moving at an unprecedented velocity. The tools used to manage these environments weren't designed for this level of complexity nor designed for rapid changes in resources. The modern data center requires juggling numerous open source repositories, handling multiple cloud providers, being able to rapidly scale up and down its resources, orchestrating changes, and populating builds across multiple servers and services.

Matters are only going to become more complex. More non-digital businesses are moving into the cloud, creating new applications that make use of mobile devices that tie them closer to their customers, suppliers, and partners. Digital-first vendors are adding features and integrating their websites with a variety of third parties that both increase their security risks and complicate their applications' flow and logic. The old days of manual labor for handling these situations are looking more than ever like the days when we last made buggy whips.

TYPICAL USE CASES

Salt was initially created to handle remote execution across complex application development environments, allowing its users to execute commands across thousands of servers concurrently and automatically. But today we need more from our toolsets than just the ability to run code remotely. Since it began in 2012, Salt has expanded its role to thrive on a mixed open and closed source environment that spans cloud and on-premises infrastructures. Here are some typical scenarios for its use:

- A developer needs to **schedule tasks to run in a particular sequence**, waiting for a dependent server to reach a particular state before it can be launched. While this can be done manually, it can be tedious and error-prone and begs for more automated methods.
- An IT manager needs to **install a particular set of updates and patches** to their environment. However, these must be done in a certain order and only when one is successfully installed can the next step be initiated. To add to this complexity, the IT department manages a mixed collection of Windows, Macs, and Linux machines that carry particular pieces of their applications infrastructure. Again, this could be done manually, but not in a reasonable time when these patches have to be applied to a thousand different servers.
- An application development manager needs to **deliver the latest build of their software stack to their production environment** while ensuring that the code is

secure and solid. Manual methods are inadequate to handle the velocity of coding changes and applications provisioning in any timely fashion.

- An infrastructure engineer needs to **set up a multi-tiered web and database application** that will require a combination of servers, networks, storage, and security devices. The complete collection spans multiple VMs, Docker containers, and physical servers, all of which have separate and complex configurations where one misstep could mean a large amount of downtime and debugging.
- A **new security exploit is discovered that has massive implications** across a variety of operating systems and system configurations. Security researchers recommend wholesale updates be done as quickly as possible to avoid any potential intrusions by hackers. Using "sneaker-net" or running from server stack to stack will take weeks to accomplish, not counting the time needed to verify the changes are made correctly.
- An engineer wants to **automatically enable auto-scaling features of their cloud provider** to match the resources needed as demand rises and falls. While the major cloud vendors offer the ability to spin up and down VMs as needed, more coordination is needed to install the right series of application servers on the new VMs and to balance the overall loads appropriately. This is nearly impossible to accomplish manually.
- An enterprise wants to **migrate its entire cloud infrastructure from AWS to Azure**, which involves moving hundreds of virtual servers in a particular order and under certain specifications for each VM. Doing this manually would involve weeks of work and workers need automation to help with the migration.

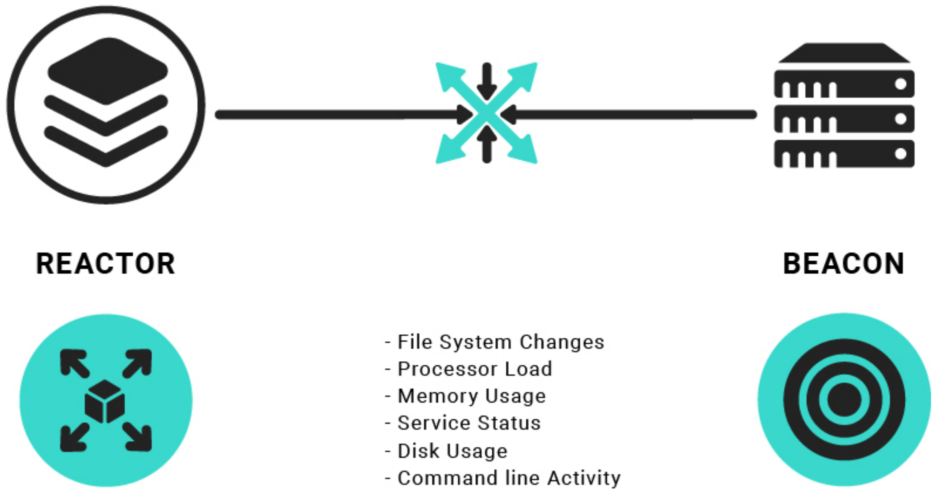
SALT'S KEY FEATURES

In each of these cases, the old-school manual methods are inadequate for reasons of time, accuracy, security, or just the sheer effort involving coordinating expensive and highly-skilled IT staffers. That is where Salt comes into play. Here are some of its key features:

Salt's **event-drive automation tools** make these tasks much easier to programmatically happen, without a lot of manual operator intervention.

Salt also **understands orchestration** and how the sequencing of various steps has to occur. Salt can handle the necessary conditional logic that control the various configuration and installation steps.

Event Driven Infrastructure



It also **contains cloud controls** that can manage public, private, and hybrid clouds. It can extract the infrastructure layer and spin up VMs under certain conditions and with certain configurations. This makes moving from one cloud provider to another easier and less error-prone.

Salt comes with sensors that react under certain conditions, such as the presence or absence of a particular application or detection of a particular OS version level.

As we said earlier, Salt originally was created for remote execution tasks. It deploys both **push and pull architectures**. This differs from many other configuration management tools that make use of one or the other methods. Salt has the flexibility to mix both, making scheduling and message-connected events simple. It addition, it can handle both **agent and agentless options** to give its automation processes the maximum level of flexibility and support to the widest collection of endpoint devices, servers, and services.

Finally, to support all these automated methods, Salt has **solid configuration management features** that can detect and manage a wide variety of circumstances. All of its scripts are written in Python, making them more accessible to a wider collection of developers who have learned this language. Other tools have their own proprietary scripting tools that have steeper learning curves.

Salt is used by a wide variety of digital businesses to manage tens of thousands of VMs and physical servers, including those at LinkedIn and eBay. At the former, it is used to serve up massive amounts of data at very low latencies to improve usability. Salt enables LinkedIn to, "...quickly and dynamically provision caching layers for many of the services that make up [their] site."

SaltStack is profoundly powerful, and to view it as simply a configuration management tool is to ignore the ways it can completely revolutionize your business's digital infrastructure. Connect with a SaltStack representative to learn how event-driven automation can optimize both your IT and your business.

ABOUT THE AUTHOR

David Strom (@dstrom, strom.com) is one of the leading experts on network and Internet technologies and has written and spoken extensively on topics such as VOIP, convergence, email, cloud computing, network security, Internet applications, wireless and Web services for more than 30 years. He has had several editorial management positions for both print and online properties in the enthusiast, gaming, IT, network, channel, and electronics industries, including the editor-in-chief of *Network Computing* print, DigitalLanding.com, and Tom's Hardware.com. He has also written two books on computer networking. He began his career working in varying roles in enduser computing in the IT industry. He has a Masters of Science, Operations Research degree from Stanford University, and a BS from Union College.