

FREE
DVD

6 Rescue Systems

BARE METAL
DEPLOYMENT

ADMIN
Network & Security



ADMIN

Network & Security

ISSUE 64

Bare Metal Deployment

IroniC

Automatically install and
set up compute nodes

Tinkerbell

Granular bare metal deployment
and life-cycle management

EKS

Automated deployment of Amazon
Elastic Kubernetes Service

6 RESCUE SYSTEMS FROM ANTIVIRUS VENDORS

PowerShell Helpers

Manage Windows AD

oCIS

A complete ownCloud
rebuild

Kind

Run Kubernetes in a container

RapidDisk

Accelerated speeds with
RAM disks

AUTOPSY

Forensic analysis

LINUX NEW MEDIA
The Pulse of Open Source

WWW.ADMIN-MAGAZINE.COM



JOIN US!

Registrations are Open!

ONE MAJOR DRUPAL COMMUNITY EVENT FOR ALL

DrupalCon Europe 2021 brings together the European Community (and beyond) by hosting regional camps online at the main event.

Attendees

will participate in all events with **one ticket**. There is **less screen time** for everyone with shorter days and an **online optimized format**.

Camp

organizers **craft their own spaces** while can leave many tedious parts of the organization to DrupalCon.

Sponsors

can target camps or the whole DrupalCon and **reach more people than ever**.

Register now at

<https://events.drupal.org/europe2021/registration-information>

To know more about the camps

<https://events.drupal.org/europe2021/join-your-local-community-drupalcon>

To become a sponsor

<https://events.drupal.org/europe2021/become-drupalcon-sponsor>

For more information, drop us an email at

drupal@kuoni-congress.com

A Failure to Communicate

How do you document failure?

I had an interesting discussion with my colleagues at work a few days ago that led to the question, “How do you document failure?” The discussion led us to the conclusion that failures aren’t widely documented. They should be. Think about it. When you search for a solution to a problem, you want an answer that worked for someone else. You probably don’t care about what didn’t work or the trial and error process of solving the problem – you only want to know about the successful solution. Whoever said, “Those who don’t know history are destined to repeat it,” was a genius. That person understood, intuitively perhaps, that documenting your failures is a wise choice.

Applying this to system administration, or any technology job, is easy. Quite possibly, you also tend to document only what works; so when you search your own knowledge bases, you find successful outcomes. However, when you’re faced with a “new” problem, how do you go about resolving it? If you’re like everyone else since humans first walked upright, you fail until you succeed. How do you know what not to try next time? You don’t know because most of us don’t document our failures.

I’ve had many experiences of consulting a senior-level coworker about a particular problem only to hear, “We tried that, and it didn’t work.” And I didn’t just hear it once. That sentence should have been on a loop that someone played for me every time I walked into their cubicle. How do I know you really tried it? How do I know you’re not guessing that it won’t work? I’m from a time and place that makes me require proof rather than just take someone’s word for something, so because of who I am, I will try all the same things that failed before eventually coming up with a common solution. Why? Because no one documented their failures.

If I could see a list of things that failed, I would be convinced that they didn’t work, especially if some explanation was also present.

For example, back in the old days of the 1990s, when sys admins always had to recompile the Linux kernel to remove bloat and to enable features that weren’t available in the default kernel, some things just didn’t work. Certain video drivers, Ethernet drivers, and other hardware peripherals didn’t work even with a recompile. Also, not that many hardware devices were Linux compatible. Unless you were a programmer, you couldn’t get them to work, either. It would have been nice to see documentation around all the things that didn’t work. It would be sort of like a hardware compatibility list in reverse.

A hardware incompatibility list or a software incompatibility list could have helped a lot. I had to deal with some quirky hardware, and it seems like each model or submodel was so different from every other one that it was almost impossible to ever create a “golden” image to make once and install everywhere. I didn’t have the luxury of adhering to the hardware compatibility list. I had to work with what I was given.

The Linux kernel has evolved and so have devices and their drivers, but many other issues could do with good failure documentation. (Insert your own list here.) The solution is to document your failures in your knowledge base or wiki. Someone eventually will need that information and will thank you for taking the time to document it.

You’ve heard that you shouldn’t try to reinvent the wheel, and it’s true, you shouldn’t. Instead, you should try to improve the wheel, but you’ll need something to go on first. I don’t want to reinvent the wheel. I just want a wheel that works, but when it doesn’t, I need some help. Perhaps I can summarize it with my version of an old rhyme we all learned as children: If at first you don’t succeed, document your failures so that I don’t have to try, try again.

Ken Hess • ADMIN Senior Editor

ADMIN

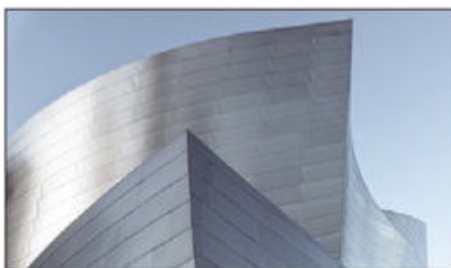
Network & Security

Features

Setting up, automating, and managing bare metal deployments gets easier with the tools presented in this issue.

10 **Ironic, Bifrost, and OSISM**

OpenStack built-in resources help automate the installation and initial setup of compute nodes.



16 **Kiwi NG**

Configure, build, and deploy operating system images in a suitable format to facilitate and accelerate the installation of new systems.



20 **Tinkerbell**

Bare metal deployment and life-cycle management, with intervention allowed in every phase of the setup.

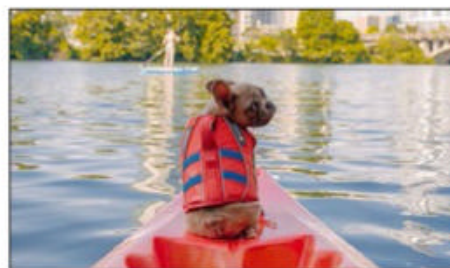


Tools

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

26 **Emergency Rescue CDs**

Antivirus vendors offer Linux-based rescue systems to repair and recover data from Windows computers.



36 **ownCloud oCIS**

A complete rebuild delivers the speedy and scalable ownCloud Infinite Scale file platform.

44 **Vembu BDR Suite**

Comprehensive software supports flexible configuration when backing up virtual production operations.

News

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

8 **News**

- Linux Kernel 5.13 released
- Linux Foundation launches new GitOps training
- Canonical and Google team up for Ubuntu Pro on Google Cloud
- Linux now set to always reserve the first 1MB of RAM

Containers and Virtualization

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

48 **Automated EKS Cluster Build**

Run a Kubernetes cluster in the cloud with ease by automated deployment with the AWS-managed Kubernetes service.



52 **Kind Multinode K8s Distro**

Create a full-blown Kubernetes cluster in a Docker container with just one command.

Security

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

56 **Autopsy Forensics Platform**

Forensic admins can use the Autopsy digital forensics platform to perform an initial analysis of a failed system, looking for traces of a potential attack.



58 **Securing Shell Scripts**

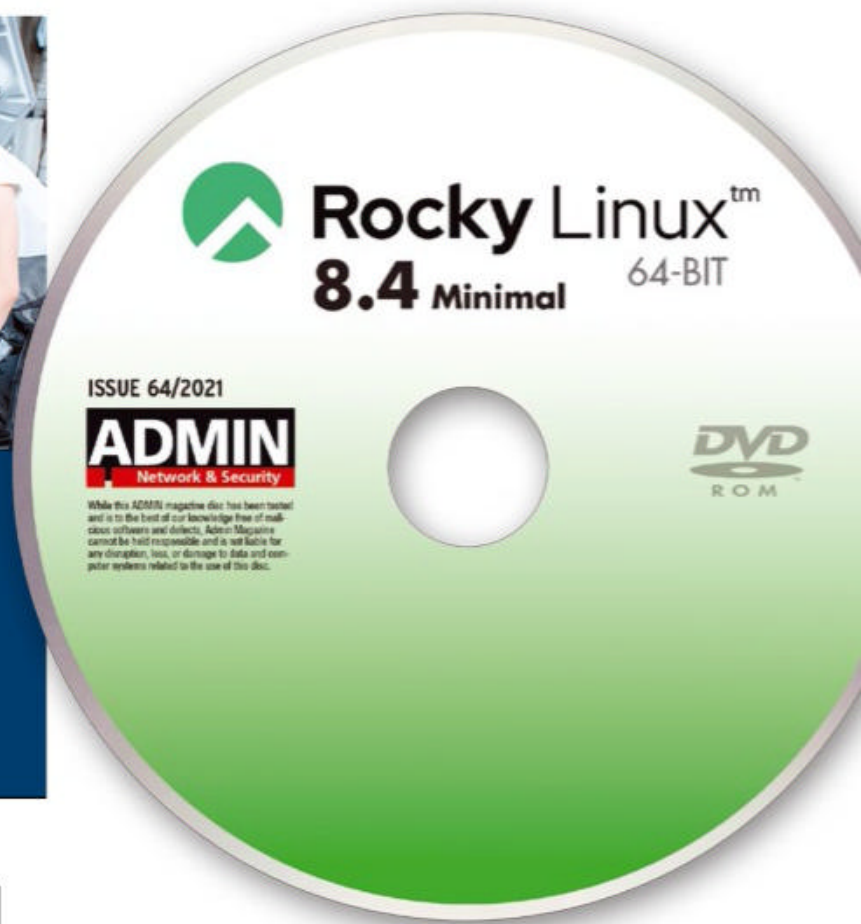
The lax syntax verification of shell scripts and a lack of attention to detail in programming can create impressively dangerous security vulnerabilities.



- 36 ownCloud oCIS**
A complete redesign of ownCloud with Go, Vue.js, and microservices promises better performance and easier configuration.



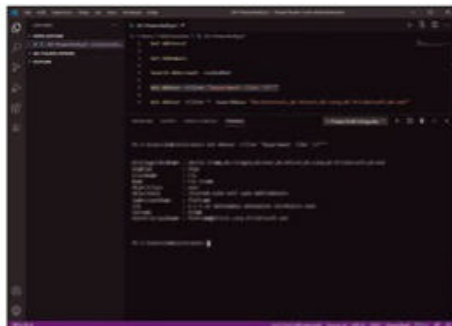
- 82 I/O Workloads**
We share a few tricks on how to isolate and diagnose the root cause of your performance troubles when it originates with underutilized or slow data storage drives.



Management

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

- 64 Manage AD with PowerShell**
PowerShell helpers let you automate searches in Active Directory and secure critical accounts.



- 70 Jenkins Self-Signed Certs**
Convince Jenkins as a Docker container to recognize self-signed certificates, and verify that the instance is connecting to the correct online service and that your traffic is transmitted in an encrypted format.



Service

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

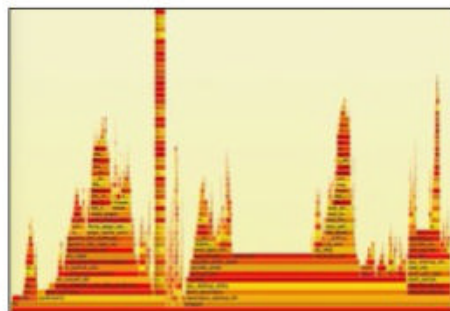
Nuts and Bolts

Timely tutorials on fundamental techniques for systems administrators.

- 74 RAM Drive Acceleration**
Enable and share performant block devices across a network of compute nodes with the RapidDisk kernel RAM drive module.
- 78 Preload Trick**
By using the LD_PRELOAD environment variable, you can improve performance without making changes to applications.



- 82 I/O Workloads**
When disk drives underperform, you have a few tools on hand to diagnose the problem.



- 93 Performance Dojo**
The implementation of terminal user interface libraries enables an impressive variety of in-terminal graphics.



- **migrate2rocky tool**
- **Multiprotocol label switching**
- **Proactive memory compaction**
- **Persistent Pacemaker resource agent**



See p 6 for details



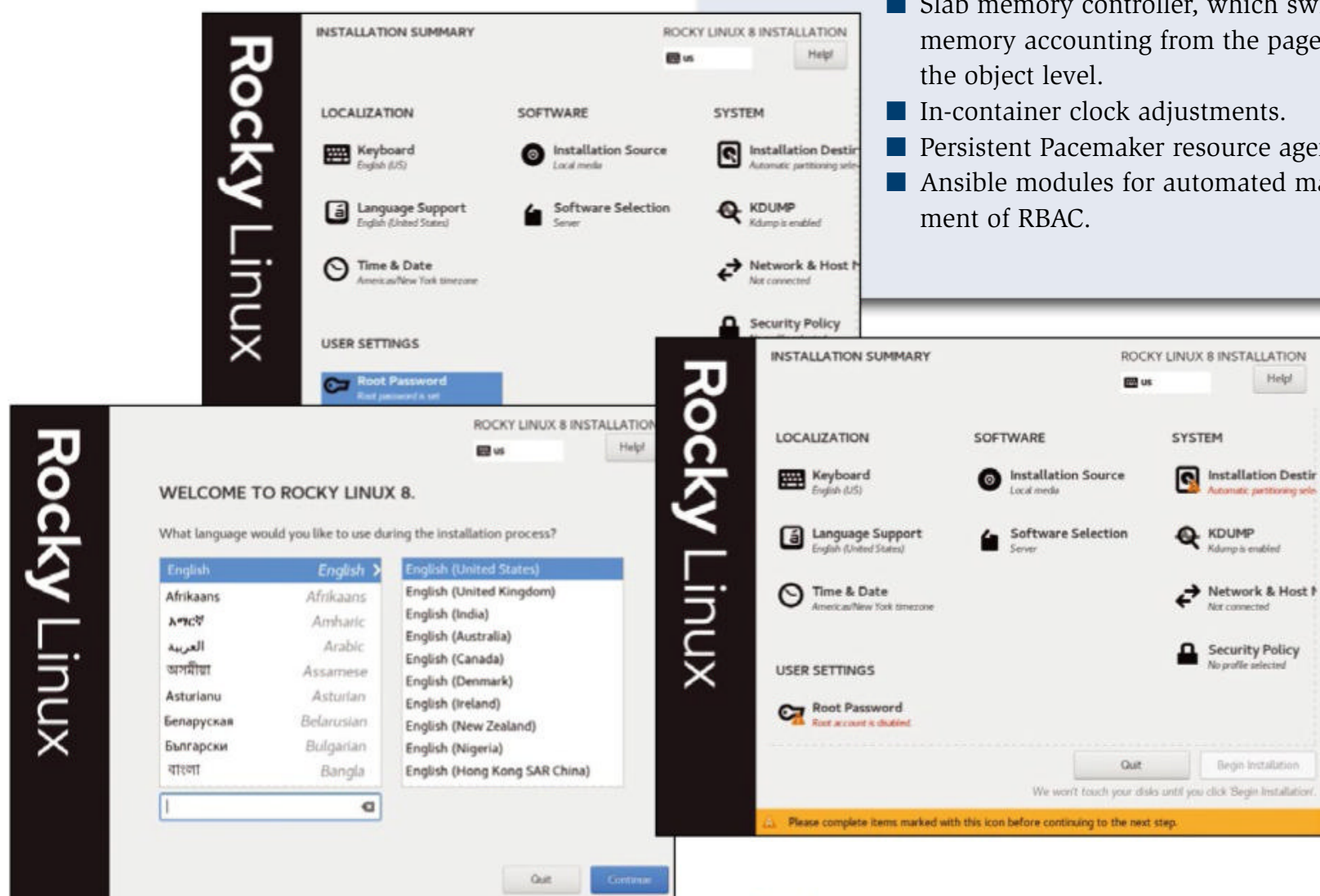
Rocky Linux 8.4 (Minimal Install)

On the DVD

This first release of Rocky Linux is “a community enterprise operating system designed to be 100% bug-for-bug compatible” with Red Hat Enterprise Linux 8.4 [1]. The Rocky Enterprise Software Foundation (RESF) strategy is to maintain a downstream build, which means releases are built after they have been added by the upstream vendor, not before, as with CentOS. The minimal edition on this DVD is intended for servers and will not include `rsyslog.service` or `/var/log/messages`, although the `rsyslog` package can be installed from the Rocky Linux AppStream repository immediately after reboot.

Rocky Linux also features:

- The `migrate2rocky` tool, which lets you convert to Rocky Linux 8.4 from other Enterprise Linux systems.
- Multiprotocol label switching (MPLS), an in-kernel data-forwarding mechanism to route traffic flow across enterprise networks.
- Proactive memory compaction to reduce latency.
- Slab memory controller, which switches memory accounting from the page level to the object level.
- In-container clock adjustments.
- Persistent Pacemaker resource agent.
- Ansible modules for automated management of RBAC.



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] Rocky Linux: [\[https://rockylinux.org\]](https://rockylinux.org)
- [2] Release notes: [\[https://docs.rockylinux.org/release_notes/8.4\]](https://docs.rockylinux.org/release_notes/8.4)
- [3] Wiki: [\[https://wiki.rockylinux.org\]](https://wiki.rockylinux.org)
- [4] Documentation: [\[https://docs.rockylinux.org\]](https://docs.rockylinux.org)



2020
Archives
Available
Now!

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.

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

News for Admins

Tech News

Linux Kernel 5.13 Released

Linus Torvalds, the creator of Linux, has made the latest kernel available, after what was one of the smoothest development processes in recent memory. Torvalds wrote in his weekly “State of the Kernel” post: “So we had quite the calm week since rc7, and I see no reason to delay 5.13.” Torvalds continued to say, “if the last week was small and calm, 5.13 overall is actually fairly large. In fact, it's one of the bigger 5.x releases, with over 16k commits (over 17k if you count merges), from over 2k developers.”

What can you expect in the 5.13 kernel? Some of the features that saw the most commits include Apple M1 support, early support for wireless wide area networks (WWANs), Microsoft's Azure Network Adapter, the Advanced Configuration and Power Interface (ACPI) spec for laptops, early work for ARM64 Hyper-V guests, RISC-V enhancements, support for Lenovo's ThinkPad X1 Tablet Thin Keyboard, support added for Apple's Magic Mouse 2, new drivers for Amazon's Luna game controller, support for AMD's Navi GPU, and new virtIO drivers for audio devices and Bluetooth controllers.

Although the 5.13 kernel is now available for downloading(<https://git.kernel.org/torvalds/t/linux-5.13.tar.gz>), you won't find it hitting the repositories for your distributions of choice for some time. For example, Ubuntu most likely won't see the 5.13 kernel appear until the 21.10 daily builds are released.

Linux Foundation Launches New GitOps Training

In conjunction with the Cloud Native Computing Foundation and the Continuous Delivery Foundation, the Linux Foundation is now offering two new classes, focused on GitOps. These two self-paced, online training courses are designed to teach all of the skills necessary for admins to start implementing GitOps into their company's workflow. The two classes offered are as follows:

- Introduction to GitOps (LFS169, <https://training.linuxfoundation.org/training/introduction-to-gitops-lfs169>) is a free introductory class that provides the foundation for admins to grasp the GitOps principles, tools, and best practices for the implementation of cloud-native applications running on a Kubernetes cluster. This course teaches how to set up and automate continuous delivery pipelines to a Kubernetes cluster, to increase productivity, reliability, and efficiency.
- GitOps: Continuous Delivery on Kubernetes with Flux (LFS269, <https://training.linuxfoundation.org/training/gitops-continuous-delivery-on-kubernetes-with-flux-lfs269/>) is a course directed at software developers who are interested



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

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



Photo by Marvin Meyer on Unsplash

Lead Image © vlastas, 123RF.com

in learning how to deploy cloud-native applications with GitHub-based workflows. This course will provide a deeper dive into the GitOps practices and offers in-depth training on how to implement those best practices using Flux CD. You can enroll in this course for \$299. You should be familiar with Kubernetes pods, ReplicaSets, deployments, services, namespaces, kubectl, and YAML specs at a minimum.

Canonical and Google Team Up for Ubuntu Pro on Google Cloud

Ubuntu Pro on Google Cloud is a new platform that not only offers a 10-year maintenance commitment, but live kernel patching; officially certified components to enable operating environments under compliance regimes such as FedRAMP, HIPAA, PCI, GDPR, and ISO; certified FIPS 140-2 components; security dashboard for Security Command Center and Managed Apps; and all standard optimizations found in the regular Ubuntu releases.

According to Jung Yang, VP and GM Compute at Google Cloud, “The availability of Ubuntu Pro on Google Cloud will offer our enterprise customers the additional security and compliance services needed for their mission-critical workloads.”

To make this even more appealing, Ubuntu Pro on Google Cloud will only be 3-4.5 percent of your average computing costs, so the more resources you consume, the smaller percentage will go to Ubuntu Pro. So not only does Ubuntu Pro on Google Cloud offer a much longer range of support, it'll help you save money on deployments at scale.

Images for Ubuntu Pro on Google Cloud can now be purchased directly from Google Cloud by selecting Ubuntu Pro as the operating system (from the Google Cloud Console). For more information on Ubuntu Pro on Google Cloud check out the official documentation (https://cloud.google.com/compute/docs/images/os-details#ubuntu_pro) and read the Google announcement (<https://cloud.google.com/blog/products/compute/ubuntu-pro-available-on-google-cloud>) about the new offering.

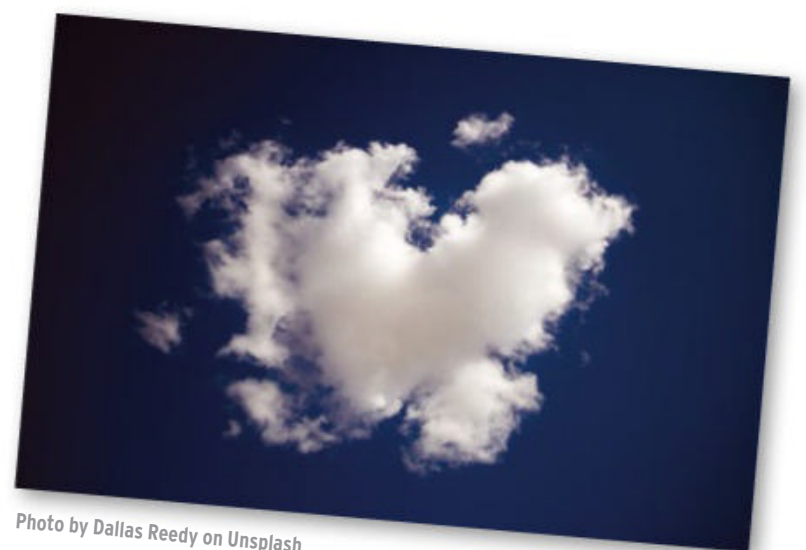


Photo by Dallas Reedy on Unsplash

Linux Now Set to Always Reserve the First 1MB of RAM

To avoid issues that have plagued some systems, the Linux kernel will now unconditionally reserve the first 1MB of RAM.

Linux developers have always known the first 64K of system memory can be easily corrupted by certain BIOSes. Another issue they've had to deal with was Intel Sandy Bridge graphics chips accessing memory below the 1MB mark. Both of these situations can lead to serious problems with system performance or even booting.

Recently, however, a bug report was filed dealing with unbootable AMD Ryzen systems when the Linux kernel version 5.13 was in use. This happened after developer Linus Chemist discovered a system with an AMD Ryzen 7 3700x was unable to boot kernel 5.13 and discovered booting worked fine with CONFIG_X86_RESERVE_LOW=640 set.

That bug was set to urgent and had an interesting (and telling) response attached to this git pull (<https://lore.kernel.org/lkml/YLx%2FiA8xeRzwhXJn@zn.tnic/T/#u>):

“Do away with all the wankery of reserving X amount of memory in the first megabyte to prevent BIOS corrupting it and simply and unconditionally reserve the whole first megabyte.”

However, Linus Torvalds had this to say about the comment:

“This seems a bit draconic. How does this work at all under Windows? There must be some windows knowledge about what the BIOS updates that we're not aware of.”

In the end, the patch was merged and the Linux kernel will now reserve the first 1MB of RAM to avoid problems in the future.

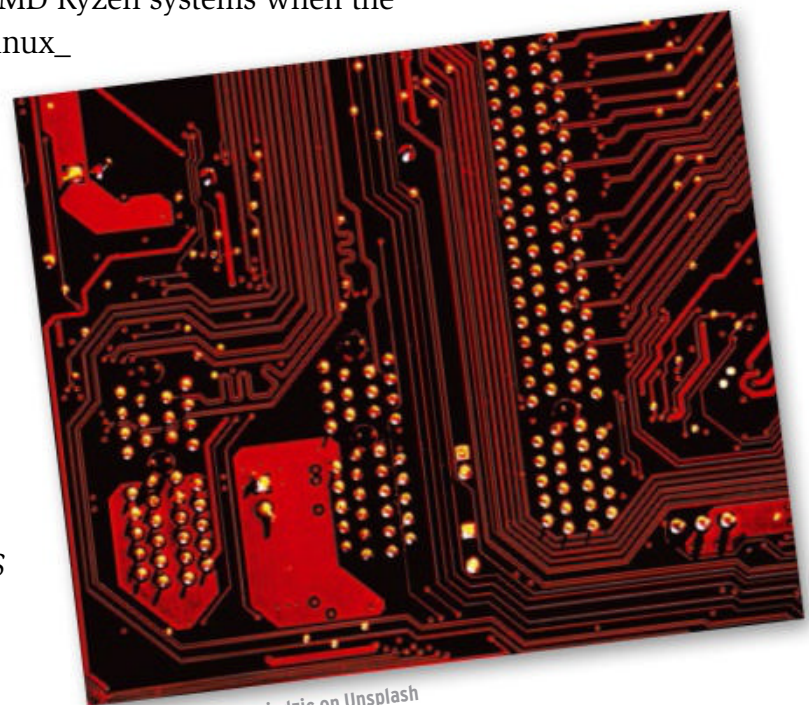


Photo by Michael Dziedzic on Unsplash

Bare metal deployment with OpenStack

Design Consultant

Automating processes in the age of the cloud is not just a preference, but a necessity, especially as it applies to the installation and initial setup of compute nodes. OpenStack helps with built-in resources.

By Bernd Müller and Steffen Schmalstieg

Once a private cloud with OpenStack is up and running, virtual machines can be created quickly, but you have to get there first. This process involves several steps, including installing hardware in the data center, wiring, and installing an operating system. The operating system installation is where the OpenStack component Ironic [1] enters the field. In this article, we introduce the Ironic

module, which specializes in bare metal installations. Ironic first appeared as an Incubator Project in the OpenStack Icehouse release of April 2017 [2] as an offshoot of the Nova bare metal driver. It was further integrated in the course of the Juno release and finally found its place as an integrated project in the Kilo release.

Bifrost

Installing an operating system raises the chicken and egg problem: The operating system is installed on disk by an installer, which in turn needs

something like an operating system under which it can run. The Bifrost subproject aims to solve this conundrum. Underneath, you have to imagine a standalone, running Ironic that OpenStack provides with all the components it requires.

This article assumes the following situation: A Bifrost instance and OSISM (Open Source Infrastructure and Service Manager) are used to deploy an operating system on a node. OSISM [3], the deployment and lifecycle management framework, installs and configures all the required OpenStack components on that node so that another node can subsequently be installed by Ironic (Figure 1).

Bifrost bridges the gap between “the hardware is ready and wired in the data center” and “now you can log on to the operating system.” The name comes from Norse mythology and

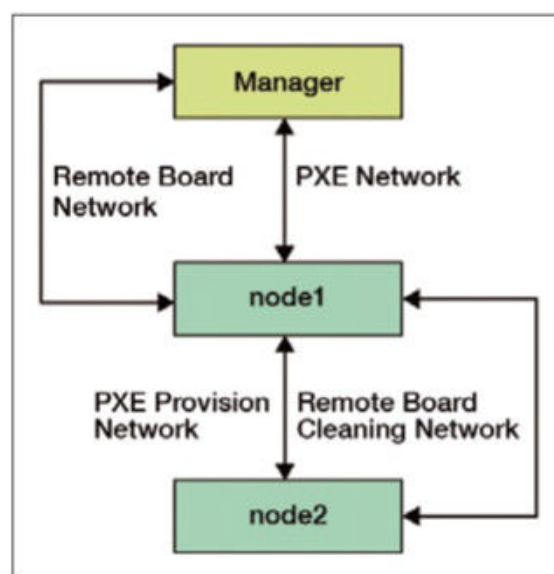


Figure 1: The sample setup for this article includes three servers with two networks each.

Listing 1: bifrost.yml

```
enabled_hardware_types: ipmi
download_ipa: true
use_tinyipa: false
ipa_upstream_release: stable-{{ openstack_release }}
cirros_deploy_image_upstream_url: https://share/ironic-ubuntu-osism-20.04.qcow2
dhcp_pool_start: 192.168.21.200
dhcp_pool_end: 192.168.21.250
dnsmasq_router: 192.168.21.254
domain: osism.test
```

Photo by Anatolii Nesterov on Unsplash

Listing 2: servers.yml

```
---
node1:
  uuid: "UUID"
  driver_info:
    power:
      ipmi_username: "<LOGIN>"
      ipmi_address: "<IP/Hostname>"
      ipmi_password: "<lessecret>"
  nics:
    -
      mac: "<MAC of NIC>"
  driver: "ipmi"
  ipv4_interface_mac: "<MAC of NIC>"
  ipv4_address: "192.168.21.21"
  ipv4_subnet_mask: "255.255.255.0"
  ipv4_gateway: "192.168.21.254"
  ipv4_nameserver:
    - "8.8.8.8"
    - "8.8.4.4"
  properties:
    cpu_arch: "x86_64"
    ram: "65536"
    disk_size: "1024"
    cpus: "16"
    name: "node1"
```

refers to the rainbow bridge between Midgard and Asgard (i.e., the connection between middle earth, the home of humans, and the kingdom of the gods and goddesses).

Bifrost requires two configuration files in the environments/kolla/files/overlays/ directory: one for itself and one on which the server is to be installed (node1 in this case). The most important parameters in the Bifrost configuration file (bifrost.yml, Listing 1) are enabled_hardware_types, use_tinyipa, cirros_deploy_image_upstream_url, and dhcp_pool*.

The enabled_hardware_types configuration option describes the interface for controlling the hardware [4], and the use_tinyipa variable gives you a choice between CoreOS (true) and CentOS (false); CoreOS is intended for continuous integration and test environments. The cirros_deploy_image_upstream_url parameter contains the link to the image to be installed, and dhcp_pool* contains the DHCP pool for the PXE environment.

Listing 2 shows the servers.yml configuration file for server(s) to be installed. It is important to use the MAC address of the deployment net-

Listing 3: 20-roles

```
[generic]
manager.osism.test
node1.osism.test

[manager]
manager.osism.test

[monitoring]
manager.osism.test

[control]
node1.osism.test

[compute]
node1.osism.test

[network]
node1.osism.test
```

work interface controller (NIC) here and not, say, that of the remote NIC interface. In the case of Supermicro, for example, you determine this with the command:

```
# ipmitool -H <IP address> \
-U <user> \
-P <password> raw 0x30 0x21 | \
tail -c 18 | sed 's/ /:/g'
```

To avoid the password specified here appearing in the process list and history, use the -f <path>/<to>/<passwordfile> parameter instead of -P <password>, if necessary, and enter the path to a file containing the password. Once you have committed both files to a Git repository, you can now turn to the OSISM Manager inventory. To begin, assign roles by modifying the inventory/20-roles file as shown in Listing 3; you need to define the host variables for node1 in in-

Listing 4: node1.osism.test.yml

```
--
ansible_host: 192.168.21.21
console_interface: eno1
management_interface: eno1
internal_address: 192.168.21.21
fluentd_host: 192.168.21.21
network_interfaces:
  - device: eno1
    auto: true
    family: inet
    method: static
    address: 192.168.21.21
    netmask: 255.255.255.0
    gateway: 192.168.21.254
    mtu: 1500
  - device: eno2
    auto: true
    family: inet
    method: manual
    mtu: 1500
network_interface: eno1
```

ventory/host_vars/node1.osism.test.yml (Listing 4).

These files also should be maintained in the Git repository and retrieved with the osism-generic configuration command in the OSISM Manager, which updates the repository contents in /opt/configuration/. This workflow always serves as the basis for an OSISM environment: maintaining the Git repository and retrieving with osism-generic configuration:

```
local: git add <path>/<to>/<file>
local: git commit -m "<Commit Message>"
```

Listing 5: Volumes, Images, CentOS

```
01 # docker volume ls
02 local    bifrost_httpboot
03 local    bifrost_ironic
04 local    bifrost_mariadb
05 local    bifrost_tftpboot
06
07 # ll -h /var/lib/docker/volumes/bifrost_httpboot/_data/
08 [...]
09 -rw-r--r-- 1 42422 42422 6.1G Mar  2 10:33 deployment_image.qcow2
10 -rw-r--r-- 1 root  root   57 Mar  2 10:21 deployment_image.qcow2.CHECKSUMS
11 -rw-r--r-- 1 42422 42422 318M Mar  2 10:13 ipa.initramfs
12 -rw-r--r-- 1 42422 42422 104 Mar  2 10:13 ipa.initramfs.sha256
13 -rw-r--r-- 1 42422 42422 9.1M Mar  2 10:13 ipa.kernel
14 -rw-r--r-- 1 42422 42422 101 Mar  2 10:12 ipa.kernel.sha256
15
16 # file /var/lib/docker/volumes/bifrost_httpboot/_data/ipa.kernel
17 /var/lib/docker/volumes/bifrost_httpboot/_data/ipa.kernel: Linux kernel x86
boot executable bzImage, version 4.18.0-240.10.1.el8_3.x86_64
(mockbuild@kbuilder.bsys.centos.org) #1 SMP Mon Jan 18 17:05:51 UT, R0-rootFS,
swap_dev 0x9, Normal VGA
```


manager: osism-generic configuration

Now Bifrost can be installed with the:

```
osism-kolla deploy bifrost
```

command to create the `bifrost_deploy` container and four volumes (Listing 5, lines 1-5). There are two operating system images: One is the Ironic Python Agent (IPA), and the other is the deployment image (lines 7-14). The IPA image is CentOS (lines 16 and 17).

The deployment image here is `ironic-ubuntu-osism-21.04.qcow2`, which was configured in Listing 1. The `osism-kolla deploy-servers bifrost` command triggers the installation of `node1`. Bifrost creates a PXE environment for the MAC address configured for `node1` – started by Intelligent Platform Management Interface (IPMI) boot into the PXE environment – and the IPA image is loaded and started. The Ironic Python Agent connects to Bifrost and writes the deployment image to disk. If this is successful, Bifrost dismantles the PXE environment, restarts `node1`, and boots from disk.

Up to Neutron with OSISM

Installing and setting up the OpenStack components requires an SSH connection between the Manager and `node1`. The private SSH key is available in the OSISM Manager in `/opt/ansible/secrets/id_rsa.deploy` and

also in the Ansible Vault in `environments/kolla/secrets.yml`; look for the `bifrost_ssh_key.private_key` variable. You can then connect:

```
$ ssh -i /opt/ansible/secrets/id_rsa.deploy ubuntu@192.168.21.21
```

After successfully opening a connection, the OpenStack components can be installed from the OSISM framework. The statement:

```
# osism-generic operator --limit node1.osism.test --user ubuntu --ask-become-pass --private-key /opt/ansible/secrets/id_rsa.deploy
```

sets up the operator user that you need for all further `osism-*` calls. The `osism-generic facts` command collects the facts, and the network configuration stored in `host_vars` is written with the `osism-generic network` command. The call to `osism-generic reboot` verifies the network configuration. After the configuration of the remaining OpenStack components in the Git repository, the commands in Listing 6 continue the installation up to the Nova phase.

OSISM relies on local name resolution and maintains the `/etc/hosts` file on manager and `node1` for this purpose. As part of the bootstrap, it sets kernel parameters, (de)installs packages, and hardens the SSH server. In this setup, the manager also takes on the controller role, which you would distribute across at least three nodes in a production setup (Figure 2).

HAProxy handles the distribution of queries to the controller nodes. The common, elasticsearch, and kibana components provide centralized logging that can be accessed from the Kibana web interface. Memcached accelerates the Apache server behind OpenStack's Horizon web interface. OpenStack requires a database (MariaDB in this case), a message queue (RabbitMQ), and a virtual switch (Open vSwitch). Now the Keystone authentication/authorization, Horizon, Glance image store, Placement resource management, Nova, and Hypervisor components are installed and set up. The following customizations for the OpenStack Neutron module allow Ironic to talk to the hardware:

```
neutron_type_drivers: "flat,vxlan"
neutron_tenant_network_types: "flat"
enable_ironic_neutron_agent: "yes"
```

The mandatory network type `flat` provides a way to communicate with the hardware at the network level. The `ironic_neutron_agent` component is responsible for communication between the Neutron network component and Ironic. According to the git commands entered earlier, `osism-kolla deploy neutron` installs and configures the Neutron component.

Listing 6: Installation Up to Nova

```
# osism-generic hosts
# osism-generic bootstrap
# osism-kolla deploy common
# osism-kolla deploy haproxy
# osism-kolla deploy elasticsearch
# osism-kolla deploy kibana
# osism-kolla deploy memcached
# osism-kolla deploy mariadb
# osism-kolla deploy rabbitmq
# osism-kolla deploy openvswitch
# osism-kolla deploy keystone
# osism-kolla deploy horizon
# osism-kolla deploy glance
# osism-kolla deploy placement
# osism-kolla deploy nov
```

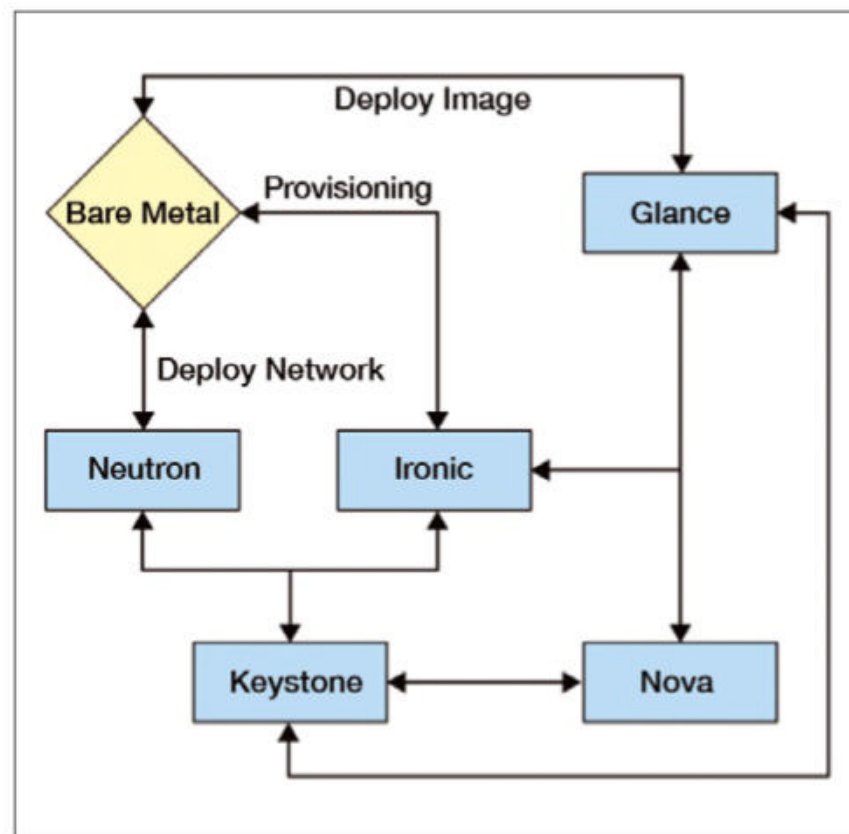


Figure 2: A rough overview of how the OpenStack components interact with Ironic.

En Route to IroniC

IroniC deployment requires a provision network and a cleaning network. The remote board network serves as the cleaning network, and the provision network handles the deployment (Listing 7). The UUIDs of the two networks can be used to populate IroniC's configuration (Listings 8 and 9 in the environments/kolla path). The command:

```
osism-kolla deploy ironic
```

installs IroniC, with three other steps in addition to adjusting the quota: Send the deployment kernel, including the ramdisk and the image to be written to Glance (Listing 10);

create a flavor with appropriate parameters [5];

```
# openstack flavor create 2
--ram 65536 2
--disk 500 2
--vcpus 16 2
```

Listing 7: Provision and Cleaning Networks

```
# openstack network create --provider-network-type flat --provider-physical-network physnet1 --share provisionnet
# openstack subnet create --network pxenet --subnet-range 192.168.21.0/24 --ip-version 4 --gateway 192.168.21.254
--allocation-pool start=192.168.21.22,end=192.168.21.28 --dhcp provisionsubnet
# openstack network create --provider-network-type flat --provider-physical-network physnet1 --share cleannet
# openstack subnet create --network pxenet --subnet-range 10.21.21.0/24 --ip-version 4 --gateway 10.21.21.254
--allocation-pool start=10.21.21.22,end=10.21.21.28 --dhcp cleansubnet
```

Listing 8: ... /configuration.yml

```
enable_ironic: "yes"
ironic_dnsmasq_dhcp_range: "192.168.21.150,192.168.21.199"
ironic_cleaning_network: "<cleannet-UUID>"
ironic_dnsmasq_default_gateway: "192.168.21.254"
ironic_dnsmasq_boot_file: pxelinux.0
```

```
--property resources:CUSTOM_BAREMETAL_2
RESOURCE_CLASS=1 baremetal-flavor
```

and create the node and port in IroniC (Listing 11).

Make sure you have the right IPMI data and that the resource class

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



matches that stored in the flavor. The flavor uses CUSTOM_BAREMETAL_RESOURCE_CLASS, which gives you a baremetal-resource-class. Other examples include CUSTOM_BAREMETAL_LARGE or CUSTOM_BAREMETAL_HANA, which resolve to baremetal-large and baremetal-hana.

As the driver-info, you need to specify the cleaning and provision networks in addition to the deploy kernel and the associated ramdisk. To enable deployment by iPXE, you need to specify direct as the deploy interface and UEFI as the boot_mode, and then create the Ironic port for node2 manually:

```
# openstack baremetal port 2
create <MAC-Address> 2
--node <Baremetal-Node-UUID> 2
--physical-network physnet1
```

or load it by in-band inspection.

Identifying the Hardware

The command:

Listing 9: ... /files/overlays/ironic

```
conf
[DEFAULT]
enabled_network_interfaces = noop,flat,neutron
default_network_interface = neutron
[neutron]
provisioning_network = <provisionnet-UUID>
```

Listing 10: Deployment Kernel, Ramdisk, and Image

```
# openstack image create --disk-format aki --container-format aki --file /data/ironic-agent.kernel
--public deploy-kernel
# openstack image create --disk-format ari --container-format ari --file /data/ironic-agent.
initramfs --public deploy-ramdisk
# openstack image create --disk-format qcow2 --container-format bare --file /data/osism-image.qcow2
--public deployment-image
```

Listing 11: Ironic Node and Port

```
# openstack baremetal node create --driver ipmi --name node2 --driver-info ipmi_username=ADMIN
--driver-info ipmi_password=<lesssecret> --driver-info ipmi_address=192.168.33.22 --resource-class
baremetal-resource-class --driver-info deploy_kernel=<Deploy-Kernel-UUID> --driver-info deploy_
ramdisk=<Deploy-Ramdisk-UUID> --driver-info cleaning_network=<cleannet-UUID> --driver-info
provisioning_network=<provisionnet-UUID> --property cpu_arch=x86_64 --property capabilities='boot_
mode=uefi' --inspect-interface inspector --deploy-interface direct --boot-interface ipxe
```

```
openstack baremetal node manage node2
```

switches node2 to the manageable state. You can now call:

```
openstack baremetal node power on node2
openstack baremetal node power off node2
```

to test the IPMI communication. Two configuration tweaks in the environments/kolla/files/overlays path, /ironic/ironic-conductor.conf and /ironic-inspector.conf, enable in-band inspection [6]:

```
[DEFAULT]
enabled_inspect_interfaces = 2
inspector,no-inspect

[processing]
add_ports = pxe
keep_ports = present
```

In addition to in-band inspection, out-of-band inspection uses, for example, the interfaces ilo, idrac, and irmc. The command:

```
# openstack baremetal node inspect node2
```

triggers the inspection. The procedure is similar to deployment: Start the PXE environment, initramfs, and the kernel, which is where Ironic gets the data for the nodes from the IPA. The command:

```
# openstack server create 2
--image deployment-image 2
--flavor baremetal-flavor node2
```

finally starts the deployment of node2.

Ironic Approach

Basically, Ironic and Nova use an image for the deployment that can be created in the same way as the images already in use. It is important to consider the underlying hardware (RAID controller, network cards, etc.); otherwise, the bridge is built with Ironic and Bifrost. Other interfaces besides IPMI provide additional options. For example, Ironic also lets you maintain the BIOS/UEFI or set up a hardware RAID.

In the form of Ironic and Bifrost, OpenStack brings well-functioning components to the table for configuring hardware with an operating system. The potential applications range from provisioning high-performance compute nodes to adding compute nodes to extend OpenStack. You have the option of customizing the image to be installed so that compute nodes can be deployed without additional manual steps.

Info

- [1] Ironic: [https://wiki.openstack.org/wiki/Ironic]
- [2] OpenStack releases: [https://releases.openstack.org]
- [3] Open Source Infrastructure and Service Manager: [https://www.osism.tech]
- [4] Ironic drivers: [https://docs.openstack.org/ironic/latest/admin/drivers]
- [5] Deploy ramdisk: [https://docs.openstack.org/ironic/latest/install/deploy-ramdisk.html]
- [6] Inspection: [https://docs.openstack.org/ironic/latest/admin/inspection.html]

Meet Me in St. Louis.

HPC is fueling breakthroughs across industries in science and beyond, from academia to commercial enterprises, by harnessing data to unlock insights faster and more accurately than ever before. HPC powers new capabilities in artificial intelligence and machine learning that, combined with modeling and simulation, accelerate discovery in solving our toughest challenges.

Join us in St. Louis to learn how HPC is empowering innovations that were never before possible to improve everyday life across the globe. Register early and save!

Program

14–19

NOVEMBER

Exhibits

15–18

NOVEMBER

The International Conference for High Performance
Computing, Networking, Storage, and Analysis



SC21

St. Louis, MO | science
& beyond.

Register Today!

sc21.supercomputing.org

Sponsored by:




sighpc



IEEE
COMPUTER
SOCIETY



TCHPC



Build system images with Kiwi

Conjurer

Configure, build, and deploy operating system images in a suitable format to facilitate and accelerate the installation of new systems. By Sebastian Meyer and Christian Schneemann

System images are installed and pre-configured operating systems, including application software, for immediate use. They come in various forms. The best known is probably the virtual system, which is used in the platform as a service (PaaS) cloud. Other uses include preloading desktop systems and laptops or SD card images for single-board computers.

You can create system images step by step by installing and configuring a system according to your wishes or existing standards and then creating a copy of the installation. However, don't forget to remove some of the configuration files when you're done (e.g., those containing the characteristics of the source system, such as MAC addresses or the hard drive IDs). Of course, this procedure is relatively time consuming and prone to error. If

you are looking for an easier way to build an image, try a tool like Kiwi. Kiwi NG [1] builds different types of ready-to-use system images (appliances), such as those for starting a virtual machine or for installing on a bare metal system. In the latter case, the tool copies the configured system image directly to the local storage medium so that it is immediately ready for use the next time the hardware is booted.

This type of installation can be extended to include configuration services, such as `yast-firstboot` on SUSE. This YaST module then queries user information such as name and password and even the time zone at first boot (Figure 1). Kiwi can do more than create images for SUSE: It works with all other RPM-based distributions, as well as for Debian/

Ubuntu and Arch Linux. The installation of Kiwi itself is explained in the box "Setting Up Kiwi NG."

Using Kiwi

Regardless of the type of system image to be created, you need a descrip-

Setting Up Kiwi NG

Depending on the distribution used, the maintainers offer ready-made installation packages from the openSUSE build service [5]. If a Python3 installation with `pip` is on the system, Kiwi NG can also be installed with the `pip install kiwi` command. After the setup, you call the binary `kiwi-ng`. The first version of Kiwi was written in Perl and progressed through version 7. Today it is known as Legacy Kiwi. A complete rewrite in Python was completed for version 9, and Kiwi NG (Next Generation) was born.

Lead image © chachar, 123RF.com

tion of the system in XML format that contains all the details of the future system (**Listing 1**), such as package lists and sources, users to be created with passwords and groups, and the partitioning or logical volume manager (LVM) layout.

After Installation

In most cases, it is not enough simply to install packages; you also need to adjust their configurations. The simplest approach for inserting your own files into the appliance is to create a directory named `root/` and rebuild the structure of the intended filesystem there with the configuration file location.

For example, if you want an `/etc/appliance_version` file on the later system, it must be stored in `root/etc/appliance_version` with the desired content. Kiwi copies the content of this directory to the filesystem after installing the packages named in the XML profile, which also allows files from the installation packages to be overwritten and thus customized.

After copying the packages to the future system and setting them up, you still might need to configure services for automatic startup. For this, Kiwi offers the possibility of storing scripts that call the tool at defined points of the build process.

Roughly speaking, Kiwi's work can be divided into two main processes: preparation and creation. The preparation part includes the steps discussed so far (i.e., installing the packages, creating users and groups, copying the files from the root directory and any additional archives specified, and finally, calling the `config.sh` script if it is present).

Finally, a Script

Kiwi runs `config.sh` in a chroot environment on the future filesystem. The script can contain arbitrary program calls that further customize the system, including the previously mentioned task of enabling services at startup (i.e., `systemctl enable <service>`), as well as downsizing

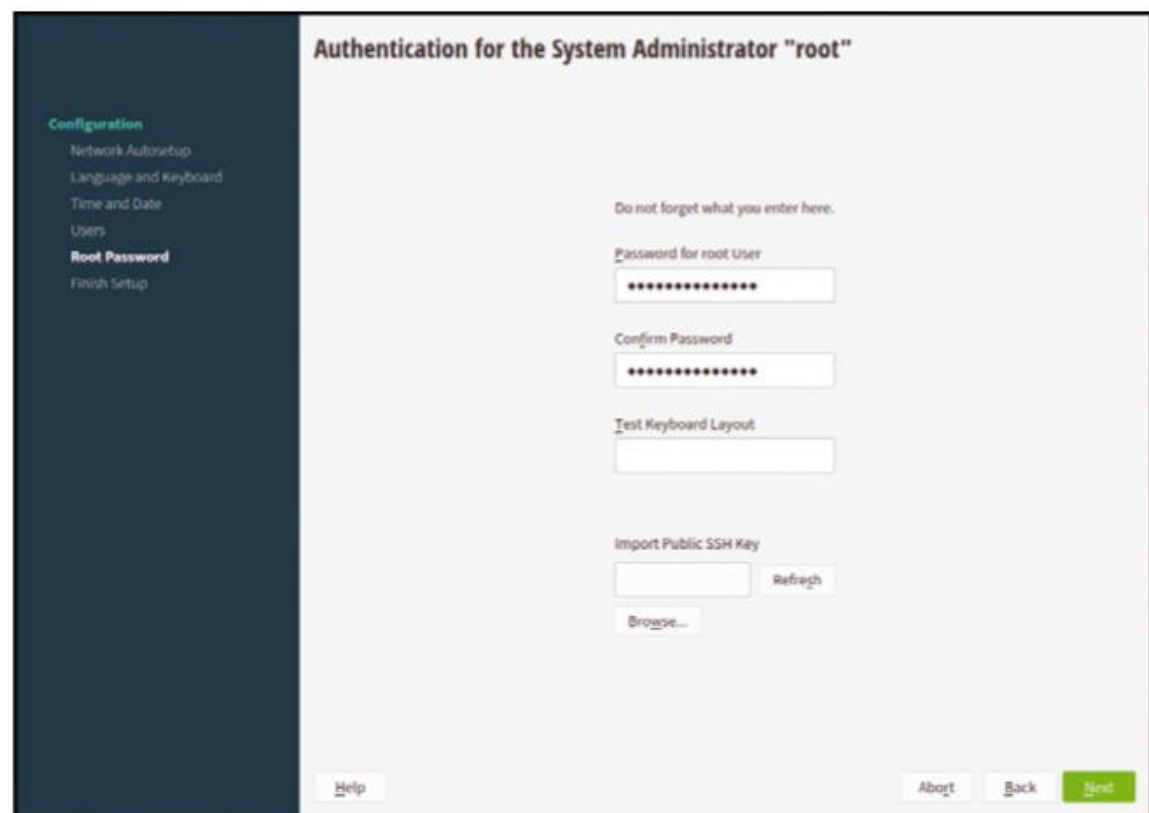


Figure 1: Prompting for the root password in YaST2 firstboot.

of the filesystem by removing unneeded files. The preparation part is now complete; Kiwi has installed the future appliance into a directory and customized it according to your specifications. In the next step, Kiwi now creates one or more image types from this directory. Before this happens, you will want to check the configuration files in the future filesystem again. This step is very helpful when you are working on a new image, especially if many adjustments need to be made to the filesystem. If you have no need for this step, the preparation and building process can be processed directly in a call to Kiwi and the desired system image created directly.

An image of type OEM from the directory in which

Listing 1: Kiwi XML Profile

```
<?xml version="1.0" encoding="utf-8"?>
<image schemaversion="6.8" name="Leap-15.2_example">
  <preferences>
    <version>1.15.2</version>
    <packagemanager>zypper</packagemanager>
    <locale>de_DE</locale>
    <keytable>de</keytable>
    <timezone>Europe/Berlin</timezone>
    <rpm-excludedocs>true</rpm-excludedocs>
    <rpm-check-signatures>false</rpm-check-signatures>
    <bootsplash-theme>bgrt</bootsplash-theme>
    <bootloader-theme>OpenSuse</bootloader-theme>
    <type image="vmx"
      filesystem="ext4"
      bootloader="grub2"
      kernelcmdline="splash"
      firmware="efi">
    </type>
  </preferences>
  <users>
    <user password="$1$wYJUgpm5$RXMMeASDc035eX.NbYWF10"
      home="/root"
      name="root"
      groups="root"/>
  </users>
  <repository type="rpm-md" alias="Leap_15_2" imageinclude="true">
    <source path="obs://OpenSuse:Leap:15.2/standard"/>
  </repository>
  <packages type="image">
    <package name="checkmedia"/>
    <package name="patterns-OpenSuse-base"/>
    <!-- .... -->
    <package name="openssh"/>
    <package name="kernel-default"/>
  </packages>
  <packages type="bootstrap">
    <package name="udev"/>
    <package name="filesystem"/>
    <package name="glibc-locale"/>
  </packages>
</image>
```


the necessary description files are located (the `--description` parameter) is created in the directory specified by `--target-dir` (e.g., `/tmp/Kiwi-ng-builddir`, in this case):

```
$ sudo kiwi-ng 2
--type oem system build 2
--description . 2
--target-dir /tmp/Kiwi-ng-builddir
```

where the files of the finished image are subsequently located.

Editing Examples

Examples of image descriptions (i.e., XML profiles with scripts and a root directory) are provided by the Kiwi maintainers in a repository on GitHub [2]. Local cloning of the repository, including the examples in this article, offers a quick-start option. The examples can then be tried out directly.

From the examples provided by the maintainers, you can quickly create your own image with individual customizations for a virtual machine. To

do this, clone the repository with `git`, change to the appropriate directory, and view the content of the example directory (Listing 2).

The `config.xml` file controls the build process by defining what should be installed and configured. After the build process, Kiwi copies the content of the root directory to the root filesystem. Finally, the `config.sh` script runs. In the example of Listing 3, the script enables the SSH daemon and `chronyd` and sets the `systemd` target to “graphical” (equivalent to the earlier `runlevel 5`). Often, time servers are useful (Figure 2) – either those belonging to the NTP pool project or your own. A few adjustments are all it takes. The `chrony` and `chrony-pool-empty` packages must be installed. Without speci-

fying the latter, the system installs a package preconfigured for openSUSE time servers. The time servers to be used are listed in the `root/etc/chrony.d/server.conf` file:

```
server 0.de.pool.ntp.org
server 1.de.pool.ntp.org
server 2.de.pool.ntp.org
server 3.de.pool.ntp.org
```

The entry `suseInsertService chronyd` in `config.sh` takes care of starting the `chronyd` service. Instead, the script can contain a `systemctl` call or any other commands. For some standard processes, Kiwi provides prepared functions for further simplification, as in the `suseInsertService` example. A list with explanations [3] of these

Listing 2: Cloning a Repository

```
$ git clone https://github.com/OSInside/Kiwi-descriptions
$ cd Kiwi-descriptions/suse/x86_64/suse-leap-15.2
$ ls -l
config.sh
config.xml
root
root/etc
root/etc/udev
root/etc/udev/rules.d
root/etc/udev/rules.d/70-persistent-net.rules
root/etc/motd
root/etc/sysconfig
root/etc/sysconfig/network
root/etc/sysconfig/network/ifcfg-lan0
```

Listing 3: config.sh

```
test -f /.kconfig && . /.kconfig
test -f /.profile && . /.profile

echo "Configure image: [$Kiwi_iname]..."

suseSetupProduct

suseInsertService sshd
suseInsertService chronyd

baseSetRunlevel 5
```

Listing 4: Querying System Configuration

```
<?xml version="1.0"?>
<productDefines xmlns="http://www.suse.com/1.0/yast2ns"
  xmlns:config="http://www.suse.com/1.0/configns">
  <textdomain>firstboot</textdomain>
  <globals>
    <enable_autologin config:type="boolean">false</enable_autologin>
  </globals>
  <workflows config:type="list">
    <workflow>
      <stage>firstboot</stage>
      <label>Configuration</label>
      <mode>installation</mode>
      <modules config:type="list">
        <module>
          <label>Language and Keyboard</label>
          <enabled config:type="boolean">true</enabled>
          <name>firstboot_language_keyboard</name>
        </module>
        <module>
          <label>Time and Date</label>
          <enabled config:type="boolean">true</enabled>
          <name>firstboot_timezone</name>
        </module>
        <module>
          <label>Users</label>
          <enabled config:type="boolean">true</enabled>
          <name>firstboot_user</name>
        </module>
        <module>
          <label>Root Password</label>
          <enabled config:type="boolean">true</enabled>
          <name>firstboot_root</name>
        </module>
      </modules>
    </workflow>
  </workflows>
</productDefines>
```

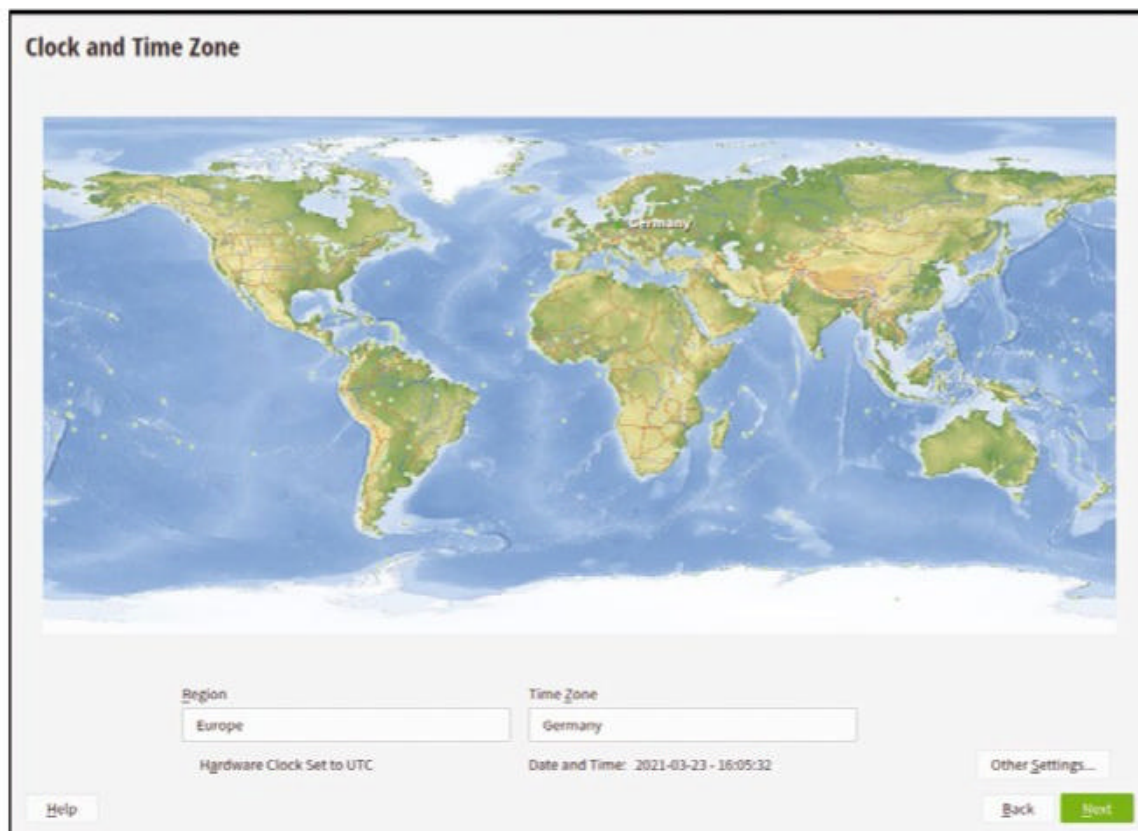



Figure 2: Configuring the time zone to be used with YaST2 firstboot.

functions can be found on the project's website.

An entry in the `root/etc/motd` file, such as:

```
Welcome to a modified openSUSE 2
Leap 15.2 appliance.
```

gives you a “message of the day.”

Final Configuration

If you want the user to set up the finished appliance on the first boot,

openSUSE provides the `yast2-firstboot` package for this purpose [4]. If it is installed, the system asks the user for the individual configuration at the time of first boot, which includes the username and password, hostname, network configuration, time zone, and more.

The modules to be called are in the `root/etc/YaST2/firstboot.xml` file on the later filesystem. Listing 4 shows a sample configuration that starts the modules for the system language, keyboard layout, time zone, and a

user to be created. The root password is also prompted for and set here, creating a customized image from a role model that can be installed and used immediately. ■

Info

- [1] Kiwi NG: [\[https://github.com/OSInside/kiwi\]](https://github.com/OSInside/kiwi)
- [2] Sample profiles: [\[https://github.com/OSInside/Kiwi-descriptions\]](https://github.com/OSInside/Kiwi-descriptions)
- [3] config.sh docs: [\[https://osinside.github.io/kiwi/concept_and_workflow/shell_scripts.html#script-template-for-config-sh-images-sh\]](https://osinside.github.io/kiwi/concept_and_workflow/shell_scripts.html#script-template-for-config-sh-images-sh)
- [4] Documentation for YaST firstboot: [\[https://en.opensuse.org/YaST_Firstboot\]](https://en.opensuse.org/YaST_Firstboot)
- [5] Kiwi package sources for various distributions: [\[https://download.opensuse.org/repositories/Virtualization/Appliances/Builder/\]](https://download.opensuse.org/repositories/Virtualization/Appliances/Builder/)

Authors

Christian Schneemann works at B1 Systems GmbH (Germany) as a Linux consultant and developer with a focus on system management and deployment. He participates in the development of custom Linux appliances and manages them on behalf of customers. Sebastian Meyer has been working for B1 Systems GmbH as a Linux consultant and trainer since 2013. His focus is on system and software management with a wide variety of configuration management and deployment tools. ■

Tinkerbell life-cycle management

Magical Management

Tinkerbell specializes in bare metal deployment and life-cycle management, allowing intervention in every phase of the setup. By Martin Loschwitz

The subject of bare metal life-cycle management is a huge topic for providers today (see the “Early Efforts” box). Red Hat, Canonical, and SUSE all have powerful tools on board for this task. Third-party vendors are also trying to grab a piece of the pie, one of them being Foreman, which enjoys huge popularity.

A vendor you might not expect is now also getting into the mix, with Equinix launching its Tinkerbell tool. Primarily a provider of data center and network infrastructure, Equinix is looking to manage a kind of balancing act with Tinkerbell. The tool is intended to enable customers to provision bare metal nodes in Equinix data centers just as easily as virtual instances in cloud environments.

Open Source Tinkerbell

With its Metal service, Equinix has been hunting for customers for several years. Customers used the company almost exclusively as a hoster. If you were looking for a collocation for your own setup, Equinix was the right choice. In this constellation, however, the customer has a number

of different tasks ahead of them: Procuring the hardware, mounting it in the rack, and cabling correctly are just a few.

Equinix Metal instead offers servers in the form of bare metal at the push of a button: Servers that Equinix keeps on hand are automatically configured to be available exclusively to a customer. Tinkerbell makes it possible

Early Efforts

Debian Installer has offered preseeding from the start for hardware management; that is, you could pass in a number of presets to the installer in the form of a text file. Configuration settings that exist in the preseeding file are then not requested by the installer. If you answer all of the installer’s questions by preseeding, Debian can be installed in a completely automated process. The free distribution is by no means the only one to support automation: Red Hat has Kickstart and Anaconda, and SUSE has AutoYaST2. Moreover, external projects like Fully Automatic Installation (FAI) can handle different distributions.

However, all these approaches are based on various assumptions about the existing infrastructure: One assumption is that the admin can find a way to start the setup routine of the respective system.

to provide the systems with exactly the kind of basic equipment admins needs for their environments. In the meantime, Equinix put Tinkerbell under a free license and published it on GitHub. The service can therefore also be used outside of Equinix Metal. In this article, I show in more detail what distinguishes this solution from other systems for bare metal management.

In the small, conventional environments of years past, this assumption was fine. Reinstalling hardware in the data center was not a recurring task. Once the admin was on site, they could quickly install systems one after another in an automatic process involving an appropriately prepared image.

Today, however, this approach no longer works. Today’s massively scalable environments (e.g., to operate Kubernetes fleets) frequently need to be expanded – with dozens or hundreds of systems being added. In recent years, therefore, the principle of the bare metal life cycle and management to match have emerged. The idea is that as soon as a machine is unpacked and wired up in the rack, it can be installed automatically and remotely at the push of a button. As part of this process, the machine is also equipped with the appropriate software; a short time later, it is up and ready for production.

Photo by Bee Felten-Leidel on Unsplash

They Already Have That?

Although bare metal life-cycle management sounds very much like marketing hype, in essence, it's all about the ability to (re)install automatically any infrastructure (e.g., servers) at any time. Moreover, the automatic removal of a machine from a setup, known as decommissioning, plays a role – albeit a noticeably subordinate one. A system that has to be reinstalled during operation because of a misconfiguration is a more common occurrence than the final shutdown of a component.

In fact, bare metal life-cycle management is a concise term for a principle that has been around for decades. The protocols that are still in use today – in Tinkerbell, too, by the way – can all look back on more than 30 years of existence. Combining them to achieve a fully automated installation environment is not new either. As an admin, you will always encounter the same old acquaintances: DHCP, PXE, TFTP, HTTP or FTP, NTP – that's it. This begs the question: What does Tinkerbell do differently than Foreman or an environment you create [1]?

A Bit of History

A detailed answer to this question can be found in a blog post by Nathan Goulding [2], who is part of the inner core of the Tinkerbell developer team and cofounder a few years ago of Packet, the company that launched Tinkerbell and now goes by the name Equinix Metal. Packet was originally independent and offered a kind of global service that could roll out systems to any location. After its acquisition by Equinix, the focus is now on Equinix's data centers, but Tinkerbell can be used entirely without an Equinix connection.

The developers' original motivation, according to Goulding, was to create a generic tool for bare metal deployments that would be as versatile as possible. However, it was by no means intended to mutate into a multifunctional juggernaut – unlike Foreman, for example, which

has long since ceased to be all about bare metal deployment and, instead, also integrates automators and performs various additional tasks. One of the motivations behind Tinkerbell, claimed Goulding, was that existing solutions had made too many compromises and were therefore unable to complete the task at hand in a satisfactory way.

Bogged Down in Detail

The basics were not the big problem, said Goulding. Taking a server out of the box, mounting it in a rack, and then booting it into an installer in a preboot execution environment (PXE) is not the challenge. In most cases, however, this is only a small part of the work that needs to be done. Saying that commodity hardware always behaves in the same way is simply not true. Anyone who has ever had to deal with different server models from the same manufacturer can confirm this. Bare metal life-cycle management therefore also includes updating the firmware, observing different hardware requirements for specific servers, and implementing specific features on specific systems – not to mention the special hardware that needs to be taken into account during deployment.

Imagine a scenario in which a provider uses special hardware such as network interface controllers (NICs) by Mellanox, for which the driver is also integrated into its own bare metal environment. If you need to buy a successor model for a batch of additional servers because the original model is no longer available, you face a problem that quite often requires a complete rebuild. Tinkerbell has looked to make precisely these tasks more manageable right from the outset.

The Tinkerbell community particularly sorely misses the ability to intervene flexibly with individual parts of the deployment process in other solutions. Indeed, Red Hat, Debian, and SUSE offer virtually no controls once the installer is running. Moreover, changing the installer with a view to extended functionality turns out to be very much nontrivial.

One Solution, Five Components

To achieve these goals, the Tinkerbell developers adhere to virtually all the specifications of a modern software architecture. Under the hood, Tinkerbell comprises five components that follow the principle of microarchitecture; it thus has a separate service on board for each specific task (Figure 1).

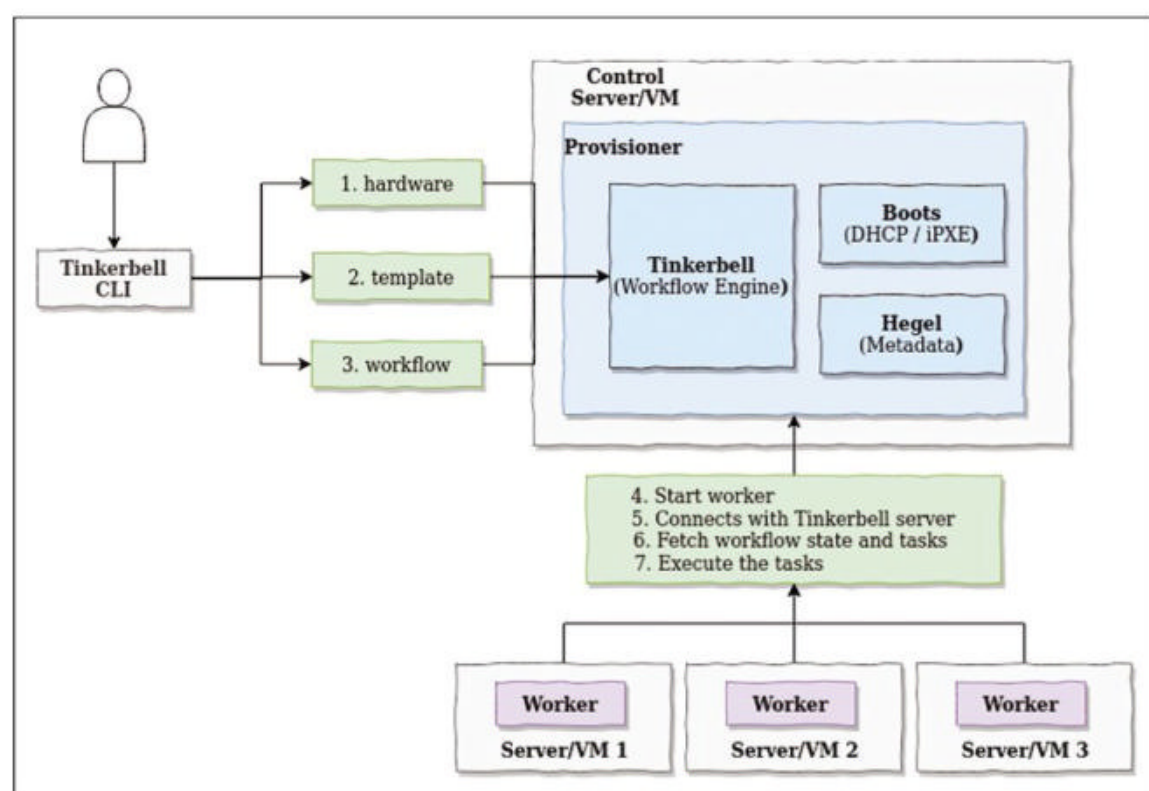


Figure 1: Tinkerbell comprises various individual components such as Boots and Hegel, each of which provides only one function. This architecture is typical of a microcomponent architecture (Tinkerbell docs [3]).

Tinkerbell does not rely on existing components; rather, it is a construct written from scratch. Consequently, the developers implemented the services for basic protocols such as DHCP or TFTP from scratch, too. Experienced administrators automatically react to this with some skepticism – after all, new wheels are rarely, if ever, rounder than their predecessors. Is Tinkerbell the big exception?

An answer to this question requires a closer examination of Tinkerbell's architecture. The authors of the solution distinguish between two instances: the Provisioner and the Worker. The Provisioner contains all the logic for controlling Tinkerbell. The Worker converts this logic in batches into logic tailored for individual machines, which it then executes locally.

Tink as a Centralized Tool

Anyone who has ever dealt with the approach of microarchitecture applications will most likely be familiar with what “workflow engine” means in this context. Many recent programs rely on workflows, which define individual work steps and specify the order in which the work steps need to be completed.

In a bare metal context, for example, a workflow might consist of a fresh server first booting into a PXE environment over DHCP, and then receiving the kernel and RAM disk for a system inventory and performing the installation. During the transition from one phase to the next (i.e., from one element of the workflow to the next), the server reports where it is in the process directly to the workflow engine, which enables it to take corrective action if necessary and to cancel or extend processes.

Tink, one of the five core components in Tinkerbell, follows exactly this approach. As Tinkerbell's workflow engine, it acts as the solution's control center. You communicate with Tink over the command-line interface (CLI) and inject templates into it in this way. A template contains the instructions to be applied to a specific

piece of hardware (e.g., a server) or the workflow that the server runs through in Tinkerbell, if you prefer. Moreover, Tink contains the database listing the machines Tinkerbell can handle.

Furthermore, Tink includes a container registry, which will become important later on. All the work that Tink does on the target systems takes the form of containers. On the one hand, this allows you to define your own work steps and store them in the form of generic containers. On the other hand, it makes the standard container images of the major distributors usable, even if Tink provides a small detour.

DHCP and iPXE with Boots

Tink is surrounded by various components that do the real work on the target systems. These include Boots,

a DHCP server, and an iPXE server written especially for Tinkerbell. As a reminder, the PXE extension iPXE offers various additional features such as chain loading (i.e., the ability to execute several boot commands one after the other).

The only task Boots has is to field the incoming DHCP requests from starting servers and to match the queried MAC address with the hardware stored in Tink. When it identifies a machine, it assigns it an IP address and then sends the machine an iPXE image, ensuring that the server boots into the third Tinkerbell component, the operating system installation environment (OSIE). This mini-distribution is based on Arch Linux, which processes the various steps defined in the template for the respective server, one after the other. OSIE uses Docker for this purpose, which lets you use simple containers of your own that OSIE calls

```

1 cat > hardware-data.json <<EOF
2 {
3   "id": "ce2e62ed-826f-4485-a39f-a82bb74338e2",
4   "metadata": {
5     "facility": {
6       "facility_code": "onprem"
7     },
8     "instance": {},
9     "state": ""
10  },
11  "network": {
12    "interfaces": [
13      {
14        "dhcp": {
15          "arch": "x86_64",
16          "ip": {
17            "address": "192.168.1.5",
18            "gateway": "192.168.1.1",
19            "netmask": "255.255.255.248"
20          },
21          "mac": "34:d0:b8:c1:11:c4",
22          "uefi": false
23        },
24        "netboot": {
25          "allow_pxe": true,
26          "allow_workflow": true
27        }
28      }
29    ]
30  }
31 }
32 EOF

```

Figure 2: Defining hardware in Tinkerbell with a JSON file; the service doesn't need to know that much about the particular bare metal.

in the sequence you define. Alternatively, you can rely on standard containers from the major vendors. OSIE is supported by a metadata service named Hegel, the fourth component, that stores the configuration parameters you specify in a template for the respective server so that they can be retrieved directly from OSIE. In principle, this process works like cloud-init in various cloud environments. The script talks to a defined API interface over HTTP at system startup; by doing so, it obtains all the parameters defined for the machine (e.g., a special script that calls the virtual instance or, in the Tinkerbell example, the physical server at system startup).

Hegel imposes virtually no limits on your imagination, although a reminder is in order: Especially in the context of virtual instances, many administrators get carried away with re-implementing half an Ansible setup in the boot script that the VM receives from its metadata. However, this is not exactly what the scripts are designed to do. In fact, they are only supposed to do the tasks that are immediately necessary. If any additional work is to be done later, you will want it done by

components that are made precisely for those purposes.

Bare Metal Is Not the Focus

The fifth and most recent Tinkerbell component is the power and boot service, which can control the machine by out-of-band management. This function was not originally part of the Tinkerbell design; however, it quickly became clear that reinstalling a system on the fly is also part of life-cycle management and only works conveniently if the life-cycle manager controls the hardware. Otherwise, you would have to use IPMITool or the respective management tool for the out-of-band interface to change the boot order on the system to PXE first and then trigger a reboot locally.

However, you can only get this to work if the affected system still lets you log in over SSH. If this does not work, you have to use the BMC interface. Tinkerbell's power and boot service allows all of this to be done automatically and conveniently from the existing management interface. The developers clearly focus on IPMI. All BMC vendor implementations offer IPMI support, even though you might have to enable it separately.

Listing 1: Tinkerbell Template

```
cat > hello-world.yml <<EOF
version: "0.1"
name: hello_world_workflow
global_timeout: 600
tasks:
  - name: "hello world"
    worker: "{{.device_1}}"
    actions:
      - name: "hello_world"
        image: hello-world
        timeout: 60
EOF
```

In any case, the only alternative to IPMI would be to fall back on Dell's remote access admin tool, Racadm. Tinkerbell does not implement these protocols itself but relies on existing tools in the background, which undoubtedly saves a lot of work.

Communication by gRPC

By now, you know the components of the Tinkerbell stack. As befits a solution in the year 2021, the individual services do not communicate with each other in any way; rather, they use the gRPC protocol that was originally developed by Google. Therefore Tink, as the central component, ultimately controls the other services

remotely in a certain way. gRPC is designed to be both robust and stable, so the developers' decision in favor of the protocol is understandable.

Practical Tinkerbell

If you want to try out Tinkerbell, you can access the developers' Vagrant containers. The objective of this exercise is not to install an operating system on a server. Instead, the Vagrant

```
1 version: "0.1"
2 name: Ubuntu_Focal
3 global_timeout: 1800
4 tasks:
5   - name: "os-installation"
6     worker: "{{.device_1}}"
7     volumes:
8       - /dev:/dev
9       - /dev/console:/dev/console
10      - /lib/firmware:/lib/firmware:ro
11     actions:
12       - name: "stream ubuntu image"
13         image: quay.io/tinkerbell-actions/image2disk:v1.0.0
14         timeout: 600
15         environment:
16           DEST_DISK: /dev/sda
17           IMG_URL: "http://192.168.1.2/focal-server-cloudimg-amd64.raw.gz"
18           COMPRESSED: true
19       - name: "kexec ubuntu"
20         image: quay.io/tinkerbell-actions/kexec:v1.0.0
21         timeout: 90
22         pid: host
23         environment:
24           BLOCK_DEVICE: /dev/sda1
25           FS_TYPE: ext4
```

Figure 3: Bootstrapping in Tinkerbell relies on templates and Docker containers. In this example, Tinkerbell writes an Ubuntu image to disk.


```

# docker exec -i deploy_tink-cli_1 tink workflow events a8984b09-566d-47ba-b6c5-fbe482d8ad7f
>
+-----+-----+-----+-----+-----+-----+
| WORKER ID | TASK NAME | ACTION NAME | EXECUTION TIME | MESSAGE | ACTION STATUS |
+-----+-----+-----+-----+-----+-----+
| ce2e62ed-826f-4485-a39f-a82bb74338e2 | hello world | hello_world | 0 | Started execution | ACTION_IN_PROGRESS |
| ce2e62ed-826f-4485-a39f-a82bb74338e2 | hello world | hello_world | 0 | Finished Execution Successfully | ACTION_SUCCESS |
+-----+-----+-----+-----+-----+-----+

```

Figure 4: The individual work steps within a template can be traced from the outside.

environment is intended to show the basic workings of Tinkerbell and the opportunities it offers.

The Vagrant files and detailed instructions [4] supplied build a Tinkerbell environment in a very short time without disabling any features. The essence of the process is not complicated. First, you roll out the Provisioner with Vagrant that contains all the services described above: Tink, Boots, and Hegel, including their sub-services and everything else an admin needs. From Vagrant, Tinkerbell then runs in Ubuntu containers, which you roll out with a Docker Compose file. Therefore, you also have the Tink CLI. You also need an image to get things started: In this example, I used the `hello-world` image provided by the Docker developers themselves. Tinkerbell runs a local container registry. You first need to run the `pull` command to retrieve the image from Docker Hub and then push it into the local registry with a `push` command.

Next, you inject the physical server configuration into Tinkerbell with the help of a file in JSON format (**Figure 2**) that gives the machine an ID, a few configuration details, and – very importantly – its network settings. This is where you specify which IP address the system will get from Boots later on and whether it uses UEFI or a conventional BIOS.

Server Template

The Tinkerbell services are running and the server configuration is where it belongs. However, Tinkerbell would not yet know what to do with the server when it phones home. It would start OSIE on the machine, but it would not install an operating system for a lack of instructions. Therefore,

the template in **Listing 1** is used in the next step.

In Tinkerbell, a template associates a particular machine with a set of instructions (**Figure 3**), which is strongly reminiscent of Kubernetes pod descriptions in style and form, even though it does not ultimately implement the Kubernetes standard. The most important thing in the template is the `tasks` section, which ensures that Tinkerbell executes the named image on the system after the OSIE startup.

In the next step, if you start a Tinkerbell Worker and give it the ID of the template and the MAC address of the target machine, the process starts up. As soon as the corresponding server boots into a PXE environment, Tink kits it out with the OSIE image and then calls the container in the image, which outputs *Hello World*.

The individual steps of the process can be displayed directly in Tinkerbell by the workflow engine (**Figure 4**). To help with debugging, you can also see which parts of the workflow were executed and whether the execution worked.

A Practical Example

In the daily grind, the rudimentary Vagrant example is admittedly unlikely to make an administrator happy. However, it shows basically what is possible. Tinkerbell's documentation also comes with concrete examples of templates that ultimately install Ubuntu or RHEL from official container images.

If you're now thinking, "Wait a minute, they don't have kernels and they don't install GRUB out of the box, either," you're absolutely right: You have to specify these steps individually in the respective templates – and

in the way you want them to be executed. As with other solutions, the greater flexibility in Tinkerbelle comes at a price of additional overhead in terms of needing to specify what you want to accomplish.

Conclusions

Tinkerbell may seem very hip to some die-hard admins – after all, it has been possible to provide servers with an operating system automatically without gRPC, Docker, and all the other stuff for quite some time now. But if you dismiss Tinkerbell because of this, you are doing the solution an injustice. It shows not only its strengths in its interaction with Equinix Metal, but also in real-world setups. A deployment system that allows such elementary intervention with the process in every phase of the setup is unparalleled on the market. It can't hurt to take a closer look at Tinkerbell and try it out. ■

Info

- [1] “The Versatile Network Bootloader iPXE” by Schlomo Schapiro, *Linux Magazine*, issue 167, October 2014; pg. 56, [<https://www.linuxpromagazine.com/Issues/2014/167/iPXE>]
- [2] Blog post by Nathan Goulding: [<https://metal.equinix.com/blog/open-sourcing-tinkerbell/>]
- [3] Tinkerbell docs: [<https://docs.tinkerbell.org/architecture/>]
- [4] Vagrant files for Tinkerbell: [<https://docs.tinkerbell.org/setup/local-vagrant/>]

The Author

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





SeaGL 2021 JOIN US!

Seattle GNU/Linux Conference
returns for the ninth year!

We're an annual community-focused free/libre/open source software, hardware, and culture event in Seattle, and since last year, *virtually* all over the world! Join us for free, no registration required!

This year, alongside the keynotes and many presentations, will see the return of the **Career Expo** - providing resume reviews and career guidance, **TeaGL** - our virtual tea time and online tea swap, and many more socialization opportunities!

Interested in helping make SeaGL happen? Lend a wing to our *all volunteer effort*! Connected to a company who wants to sponsor? Contributions will have a *real and lasting impact* on the FLOSS community!

November 5th & 6th, 2021
Virtual Conference

Visit SeaGL.org for more information!





Six emergency CDs from antivirus manufacturers

Life Jacket

Linux-based rescue systems, such as those offered by most antivirus software manufacturers, can repair and recover data from Windows computers compromised by malware. By Erik Bärwaldt

Because of its many security weaknesses, Microsoft Windows is considered particularly vulnerable to malware such as viruses, worms, and Trojans. However, because the operating system has more or less a monopoly on the desktop, administrators in the enterprise in particular are repeatedly confronted with malware damaging Windows workstations or even rendering them unusable.

In such cases, it usually makes sense to boot the damaged computers from a removable disk and then examine the mass storage devices with an external rescue system. For this purpose, many manufacturers of proprietary antivirus and anti-malware software provide free emergency CDs that use the manufacturer's own scanners to detect and remove malware on Windows systems. In many cases, they can also be used to repair damaged system files so that the Windows system can be used again.

The rescue CDs, available for download as ISO images on the websites of the respective manufacturers, are based on Linux derivatives and, in addition to the antivirus scanners, often contain free Linux tools for creating a backup of the target systems or managing mass storage devices.

Avira Rescue System

Avira [1], a company based in Tettwang on Lake Constance, Germany, has developed over the years into a global player in the field of system security, primarily for Windows computers. Avira offers various antivirus applications for products from Redmond but has continuously expanded its portfolio in recent years to include virtual private network (VPN) services, security solutions for mobile systems such as Android and iOS, and protection software for dealing with email.

The company also offers a freely available rescue system based on Linux called Avira Rescue System [2]. The system, offered as an ISO image, focuses on damaged Windows computers. It scans target computers for malware, can repair the operating system, and supports editing the registry file that acts as the central configuration database on Windows computers. The tool does not let you repair boot sectors or modify partitions. Additionally, Avira Rescue System also scans Linux systems for malware, which it also can remove. The hybrid image, which weighs in at around 1.2GB, can be booted both from an optical data carrier or from a USB memory stick.

Start

The system, based on Ubuntu 20.04.1 and equipped with a fairly up to date 5.8 kernel, first boots into a graphical GRUB screen that



Figure 1: The Avira Rescue System has an up-to-date look.

offers various startup options and settings. You can also test the RAM or check the mass storage available on the system from the corresponding menu entries. The system then boots into a modified Unity desktop, which has just a few applications in the vertical panel on the left (**Figure 1**). For an overview of all the installed applications, click the tile icon at the bottom of the vertical panel. The start buttons of the existing applications now appear on the desktop.

Features

The panel bar hosts a very fast web browser borrowed from the Gnome treasure trove, as well as the Déjà Dup backup program, also courtesy of Gnome, and the forensic Windows registry editor Fred. The central element of the distribution, however, is the Avira Rescue System, which you launch by pressing the umbrella icon in the top left corner of the desktop. After confirming the license, the tool performs some basic configuration steps before you press the *Quick scan* button at bottom center to run a quick system scan. The strong focus on Windows is already noticeable in this dialog. If you launch the Avira DVD on a computer whose mass storage consists entirely of Linux partitions, the rescue system stops without any further activity in the system scan (**Figure 2**). With multiboot

installations, the scan software fails to deliver satisfactory results. The *Update* dialog is also non-functional on Linux.

In the Tools dialog, in addition to the Déjà Dup backup tool, you will also find a button for calling the TeamViewer remote maintenance software. Both can also be used on Linux, although the individual versions of TeamViewer are not compatible. TeamViewer v15.15.3 included in the Avira package was the current version of the remote maintenance solution at press time.

Outside of Avira's own rescue tool, you will find the GParted graphical front end in the tile overview of the applications, which you can use to manage mass storage devices. This

universal tool is useful for creating and editing partitions and drives. A terminal for executing Linux commands is also available.

On Windows

The Avira Rescue System also exhibits some minor weaknesses on Windows. After starting the tool, you must update the virus signature database. The software displays the message *Detection database is outdated* to let you know. After clicking *Check update*, it first checks whether Internet access is available. If so, a click on the *Start update* button updates the database and displays a message to that effect. Next, press *Start Scan* to start the system scan and choose between a quick and a full scan of the Windows system. Even a quick scan of a current NVMe SSD was quite slow in the test. You will need to allow several hours for a scan, even with a freshly installed Windows on a larger partition. Afterward, any malware identified in the scan can be removed.

Comodo Rescue Disk

Comodo Rescue Disk is a rescue system for Windows computers that uses Linux tools developed in-house. The US manufacturer [3] uses the lean SliTaz Linux from Switzerland as its basis. The operating system in Comodo, developed independently from sources, comes with kernel 2.6.37 and uses the Openbox window

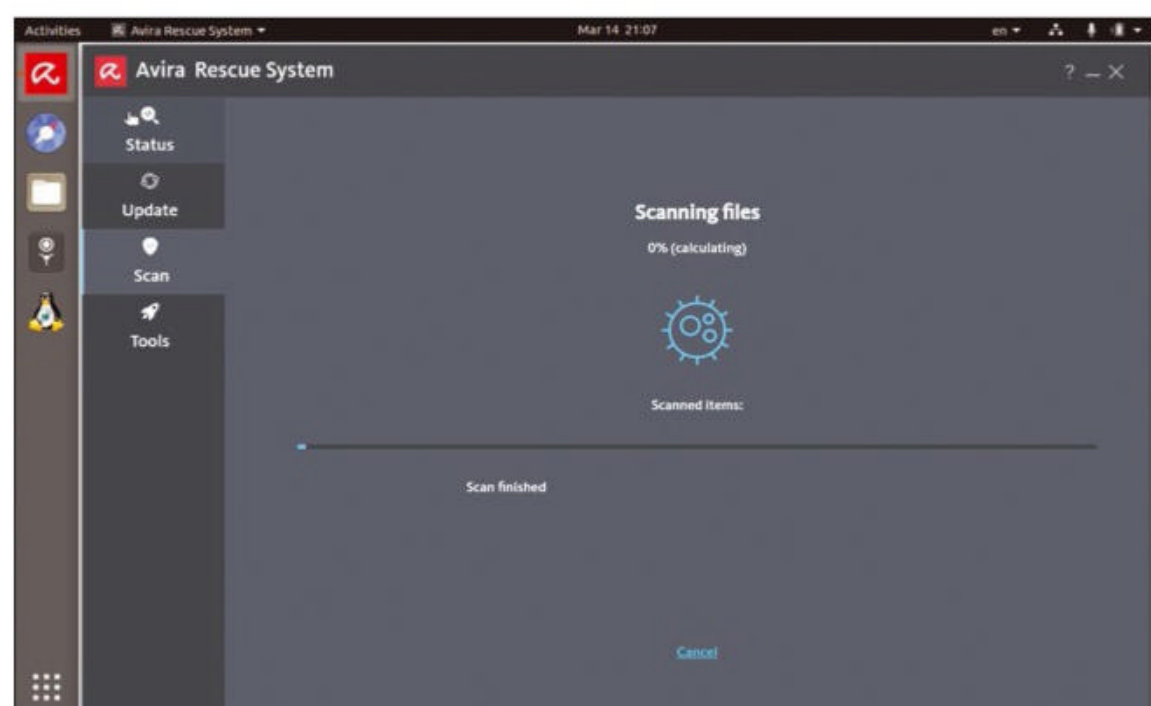


Figure 2: Avira Rescue System is for Windows only, despite the penguin in the sidebar.



Figure 3: The Comodo Rescue Disk provides various scan modes from which to choose.

manager. It will also run on 32-bit hardware and works frugally and quickly on all platforms. You can pick up the Comodo Rescue Disk system in the form of a hybrid ISO image [4] of only about 55MB, which then runs from a USB flash stick or CD-ROM.

Software

Comodo Rescue Disk only includes a small software inventory. The central element that stands out is Comodo Cleaning Essentials (CCE), a collection

of GUI tools for removing malware from Windows systems. CCE loads automatically after booting the system and first updates the signatures; then, a wizard opens to scan the Windows system.

To begin, you define whether you want to perform a *Smart Scan*, a *Full Scan*, or a *Custom Scan* (Figure 3). For the *Custom Scan*, you specify in a separate dialog the directories to scan and whether CCE should also check the system's boot

sector. The *Full Scan* and the *Smart Scan* have no further settings. CCE first updates the malware signature database and then starts the system scan (Figure 4). In the process, a progress bar, updated almost in real time, provides information about the problematic files found.

Before selecting a scan option, you should click the *Options* link at the top of the program window. Specify whether CCE should also check the disk's master boot record (MBR) for modifications and whether it should include file archives in the scan.

Also, you can specify the maximum size of the content to be scanned; by default, the scan leaves out files larger than 40MB.

The *Tools* link, also at top right, lets you view the files removed to a quarantine directory after the scan has been completed and evaluate the logfiles generated by default for each run. You can save the plain text logfiles to an external data medium for later documentation.

Additional Tools

The Comodo Rescue Disk also has a small program menu in the upper panel where you will find the SliTaz Netbox Manager for configuring the LAN and WiFi interfaces, the lean Midori web browser, the PCManFM file manager, a small screenshot tool, and the Xterm terminal. Installing other applications on the system is not possible.

Text Mode

As a special feature, the Comodo Rescue Disk also features a text mode,



Figure 4: Comodo Rescue Disk comes with its own tool for rescuing Windows installations.

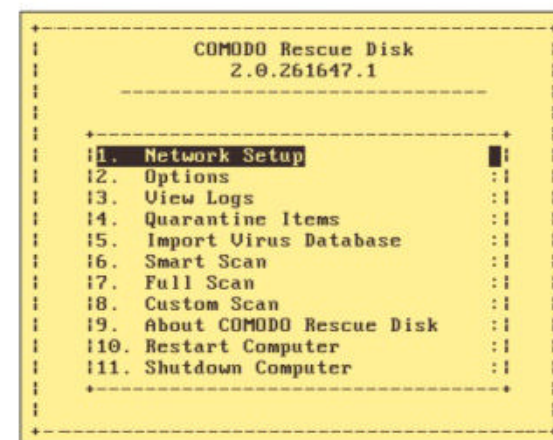


Figure 5: Friends of text mode will also get their money's worth from Comodo Rescue Disk.

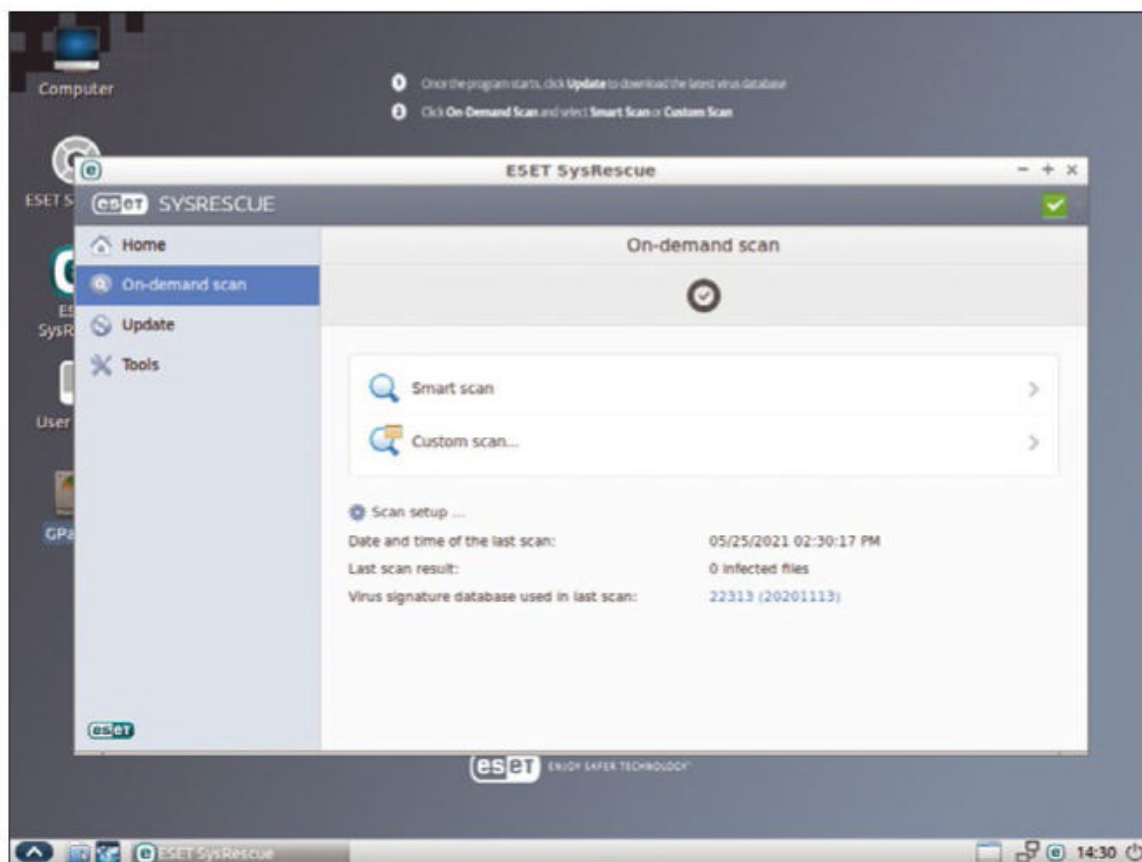


Figure 6: The ESET SysRescue program takes care of scanning and repairing Windows partitions.

which you enable in the GRUB boot menu from the *Enter Text Mode* option. In this mode, CCE starts as a text-only application that integrates some additional options that run separately in graphical mode (Figure 5), including, for example, a tool for configuring the network. Operation is entirely from the keyboard, and output is partly by means of semi-graphical elements, such as a progress bar when scanning the mass storage devices.

ESET SysRescue Live

For almost 30 years, Slovakia's ESET [5], based in Bratislava, has been developing various security solutions for Microsoft operating systems that can also be used with very old Windows installations. The free ESET SysRescue Live [6] distribution is available from the provider as an ISO image for optical media and as an IMG image for USB removable media. The USB IMG, at around 740MB, will also fit on smaller storage media.

Tools

The central tool for repairing a compromised Windows system is the ESET SysRescue tool, which launches automatically after booting the Ubuntu-based Linux derivative.

After confirming the license, the main SysRescue tool window opens (Figure 6). Start by updating the signatures for the virus databases by clicking *Update* in the left vertical bar. Next, press *On-demand scan* to start scanning the Windows system (Figure 7). A settings dialog opens on the right of the window and you can decide whether you want to run a *Smart Scan* or a *Custom Scan* of the system. If you select the user-defined variant, you can then modify numerous settings in

another window to influence the scope of the check.

During the subsequent scan, the software continuously states the number of scanned objects and displays a progress bar. The scan window also tells you how many threats have been detected. The routine distinguishes between infected and cleaned objects after the scan is complete.

ThreatSense

ESET SysRescue Live uses what is referred to as ThreatSense parameterization to detect infected files and objects. You can set the parameters in several groups in the *ThreatSense parameters* section of the *Advanced setup* dialog.

The ThreatSense checking module bundles various threat detection methods. You define both the objects the tool includes in each scan and whether you want to detect potentially insecure or undesirable applications. You can then specify how the routine should deal with this content. Optionally, the software offers to repair infected objects (i.e., rid them of dangerous code or remove them from the system). You can use a slider to specify the sensitivity of the program.

In another dialog, you can specify the maximum size of individual objects.

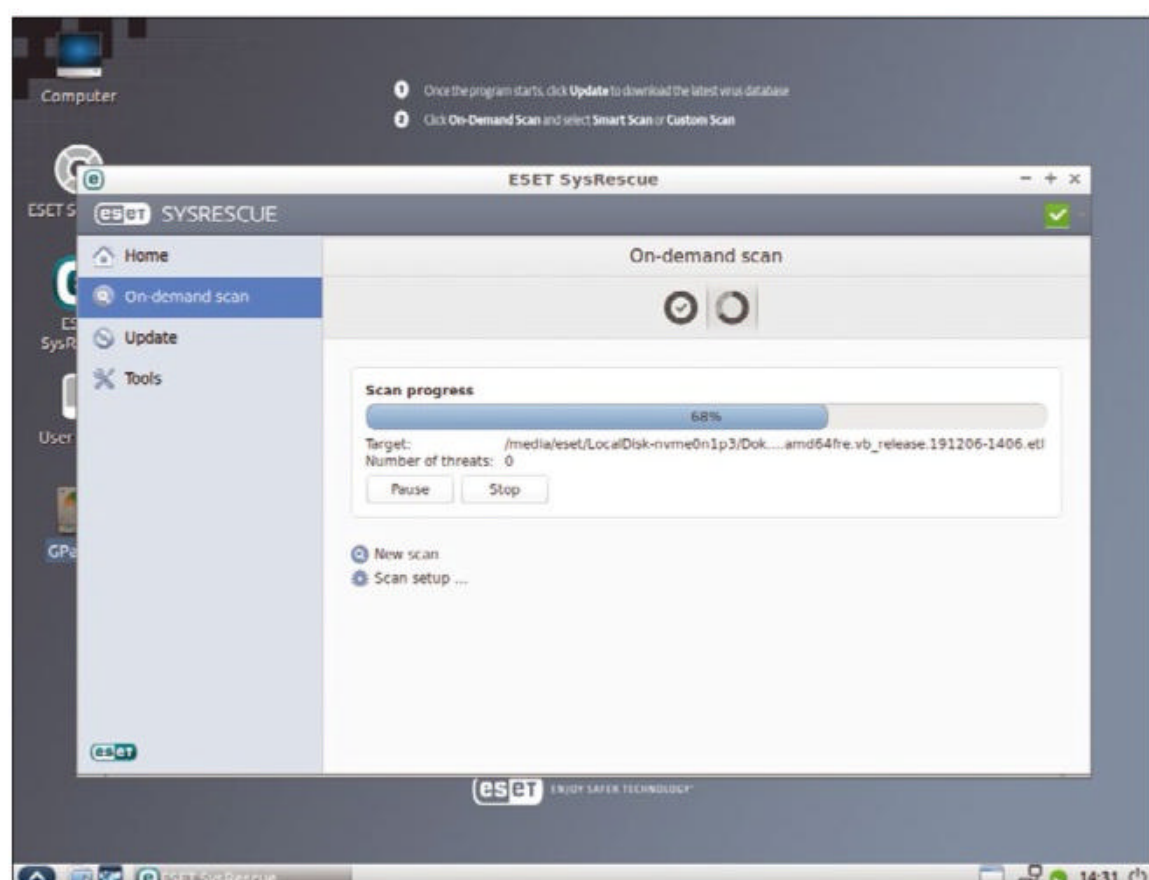


Figure 7: ESET SysRescue visualizes the progress of the scan in a progress bar.

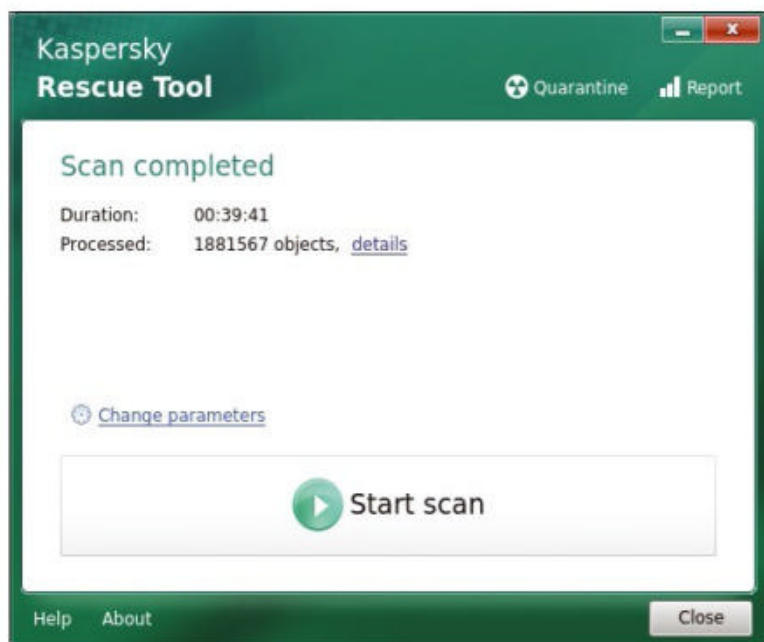


Figure 8: The Kaspersky Rescue Tool comes with a plain interface.

In the same dialog, you can also specify the nesting depth up to which the check scan archives, with 10 levels as the default. Once you have completed the settings, you can initiate another run by pressing the *New scan* link.

Additional Options

In the main window of the application, you are taken to additional options after selecting *Tools* on the left. The *Log files* option lets you view the logfiles created during the test run and filter them by various categories (e.g., information, errors, and warnings). The *Quarantine* option lets you restore objects from or move objects into quarantine. For this purpose, the software displays a corresponding file selection dialog. In *Security report*, you can trace the different threat scenarios on your computer system in a chart. The *Submit sample for analysis* option gives you the option of submitting a suspicious file to ESET for evaluation.



Figure 9: Kaspersky offers some additional tools on its Rescue Disk for Windows.

On Linux

The check routine in the ThreatSense module can also check Linux systems for various threats. Because the corresponding dialogs explicitly allow scanning email files for malware, the application is suitable for use on Linux-based mail servers. However, the malware hidden in mail files does not typically unleash its effect on the mail server itself, but on the mail

clients running on Windows.

The GParted graphical partitioning tool is also available for use with Linux clients and can be used to fix mass storage problems. It supports a variety of filesystems and is thus also suitable for use in multiplatform environments. For efficient work with files and directories, the rescue distribution also has Midnight Commander and its graphical counterpart PCManFM on board.

Kaspersky Rescue Disk

The Rescue Disk [7] released by the Russian security specialist Kaspersky Lab [8] is available for download free of charge from the company's website. The hybrid ISO image with

a size of around 610MB is based on Gentoo Linux. After downloading and transferring it to a bootable medium, the operating system first opens the GRUB graphical boot manager, where you can choose between the English and Russian localizations. After that, you will find various boot options in another dialog. Because of its lean design, the system boots very quickly to a visually unobtrusive Xfce 4.12.2 desktop. The Kaspersky Rescue Disk feature set primarily focuses on repairing Windows systems, but it is much more broadly positioned than many other solutions: Linux systems can also be checked with the built-in Rescue Tool, and additional software such as Midnight Commander, htop, and a hardware detection tool support flexible use of the suite.

After booting, the Rescue Tool automatically comes up to perform a system scan. If you don't have a network connection, the system first pauses the initialization routine to give you the opportunity to configure WiFi access. Once the network connection is established, the software displays the dialog for scanning the system after initializing the scan engine. If necessary, you can include boot sectors and the EFI system partition in the scan. The Rescue Tool quarantines any malware it finds to render it harmless. For documentation purposes, you can also generate

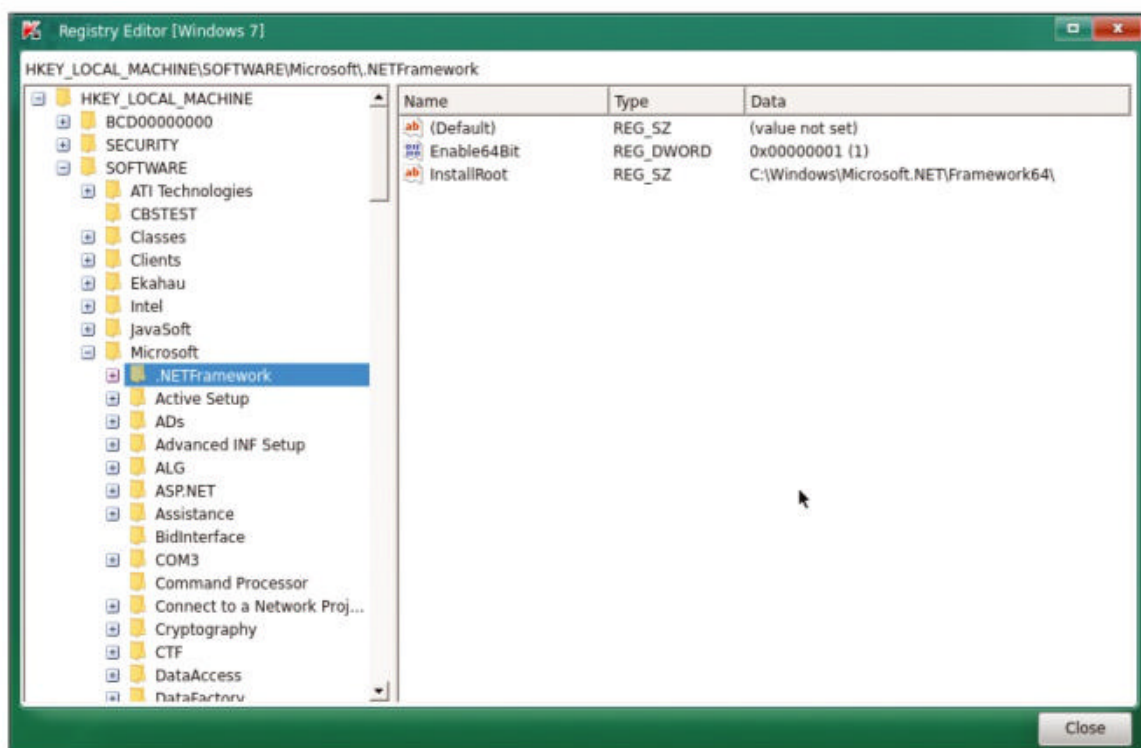


Figure 10: Kaspersky's rescue system also lets you edit the infamous Windows registry.

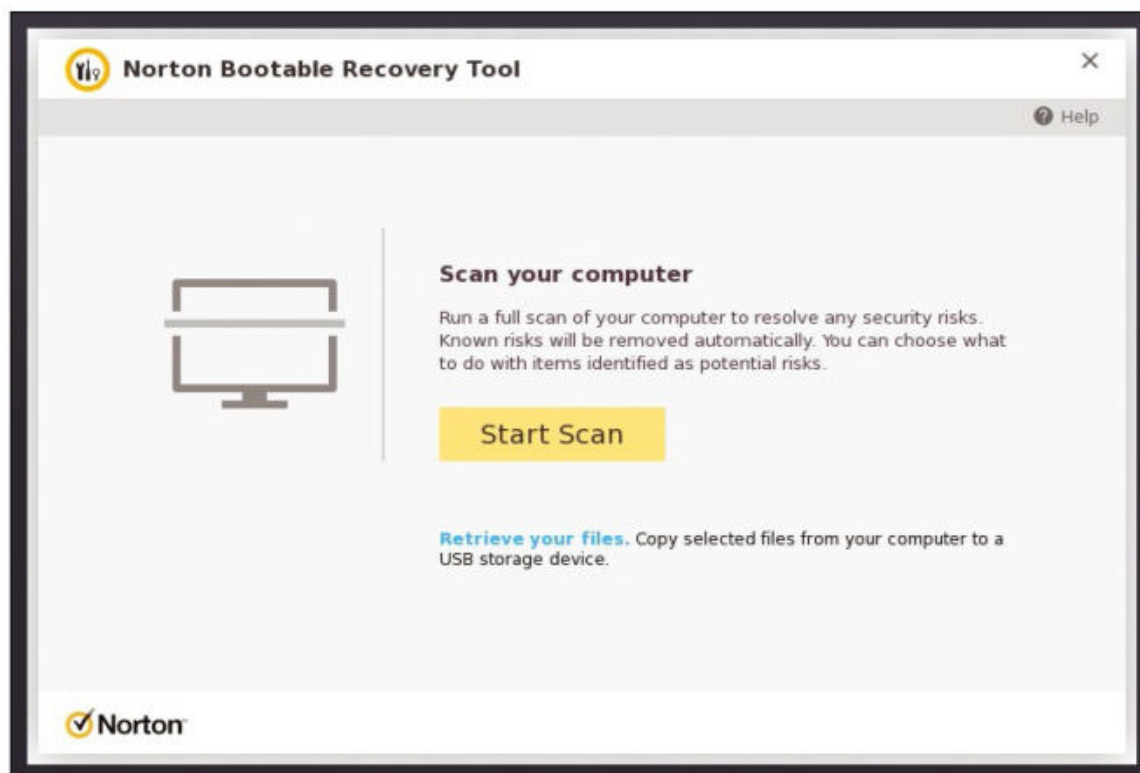


Figure 11: Scan complete Windows systems with the NBRT tool.

a report of the scan after it is complete (Figure 8).

On Windows

For Windows systems, the scan runs as for Linux systems, but the Rescue Tool only performs a quick scan by default. To scan the Windows system completely, you need to check the boxes for the drives in the dialog that comes up after choosing the *Change parameters* link. In the scan window, you will notice the *Tools* link that is not present when scanning Linux systems. Clicking on it opens a dialog window that offers a *Registry Editor*, a *Windows Unlocker*, and the *USB Recover* icons (Figure 9).

Whereas the Unlocker combats ransomware, the USB Recover tool restores the function of USB devices that were accidentally removed from Windows by mistake. For this purpose, it is also necessary to edit corresponding keys in the Windows registry (Figure 10).

Norton Bootable Recovery Tool

The Norton Bootable Recovery Tool (NBRT [9]) offered by US vendor Symantec is also a Linux-based Live system, which the manufacturer offers for download as a hybrid ISO image. At 850MB or so, the recovery

distribution looks reasonably up to date after the first boot (Figure 11), but a look under the hood reveals plenty of antiquated components: NBRT is based on Red Hat Enterprise Linux 6, which was released at the end of 2010, although still supported. The 32-bit system therefore still uses kernel 2.6.32.

When booting, NBRT enables a wizard as the central element of the rescue system; the Start menu has been banished from the panel. The only other utilities available are the Opera web browser and a terminal, which can be launched directly from the panel. Opera is also anything but new: The

12.16 version integrated into NBRT was released on July 4, 2013.

Toolbox

The wizard, which launches automatically after booting the system from a removable disk, can only be used to repair Windows systems. After selecting the language (if needed) and confirming the license, NBRT updates all the malware definition files. This process can take a long time and is accompanied by the message *Updating definitions*.

If the software finds an installed Windows system, it automatically starts a scan of the entire mass storage, otherwise it aborts with an error message. NBRT cannot be used for safeguarding Linux systems. The tool lists problematic files found during the scan in a table, indicating a threat level in addition to the file name. You can then define in the *Action* column what to do with the respective file. NBRT automatically enables a delete function for dangerous files; you might have to disable this function explicitly by unchecking the box to the left of the respective entry. If you do not make any changes to the entries in the table, the listed files are either deleted or repaired after clicking *Fix* top right in the window after a further prompt. This process may make a blocked Windows system workable again.



Figure 12: NBRT also lets you back up individual folders and files.



Figure 13: The REVE Rescue Disk is based on Puppy Linux and is useful for both backups and as a rescue system.

Data Backup

After updating the signature data, NBRT can save important files from the corrupted Windows installation, even before starting the system scan. For this purpose, NBRT has a special save dialog that you enable with *Retrieve Files*. A file manager opens with the system partitions from the mass storage (**Figure 12**), in which you select the desired files for backup by checking the boxes, before backing them up to a removable USB drive.

If your machine does not have a USB storage device attached to the system, the application points out this shortcoming with a corresponding error message and lets you connect a device to the Windows computer before backing up the files. If possible, you should not save any Windows system files, only personal files, to the removable medium because it is not possible to rule out the possibility of malware infecting system files.

REVE Rescue Disk

In 2014, REVE Group first launched REVE Antivirus **[10]**. The Singapore-based company is one of the few to offer an antivirus solution specifically for Linux. The software, which is also intended for desktop systems, is available in variants for 32- and 64-bit systems.

The vendor also provides the REVE Rescue Disk **[11]**, which can be downloaded free of charge after registration. The approximately 240MB ISO is based on the lean Puppy Linux 5.2.8 and offers a graphical user interface as well as numerous graphical tools. After starting the system, an illustrated quick start guide opens first. The Puppy configuration dialog then lets you localize the operating system and adjust the screen resolution. After restarting the X server, a fully-fledged Puppy Linux with a customized background comes up (**Figure 13**).

Concept

Unlike other manufacturers' rescue systems, REVE Rescue Disk does not

focus on a proprietary tool for restoring a Windows system crippled by malware; instead, it puts data backup in the foreground. Puppy is excellent for this, because it allows the individual partitions to be mounted and opened at the push of a button by displaying the existing drives on the desktop. Folders and files can then be transferred by drag and drop between the active file manager instances (**Figure 14**).

The REVE Rescue Disk stands out as a practically complete Puppy Linux system that includes all preinstalled system tools. You can install any software you like with the Puppy Package Manager, and you can even install on a mass storage device (although this doesn't normally make much sense). The REVE Rescue Disk is therefore also excellently suited for working with damaged Linux systems. To repair compromised Windows systems, on the other hand, you have to use the rescue disk to transfer the affected partitions to a removable drive and process them on another system with a commercial scanner solution from the REVE Group.

Only a few applications are pre-installed on the disk, including the remote maintenance solution TeamViewer, although in a fairly antiquated version 7 and with a complete Wine runtime environment. Besides this, Google's Chrome – again this is an outdated version 21.0 – and the lean Dillo web browsers are available.

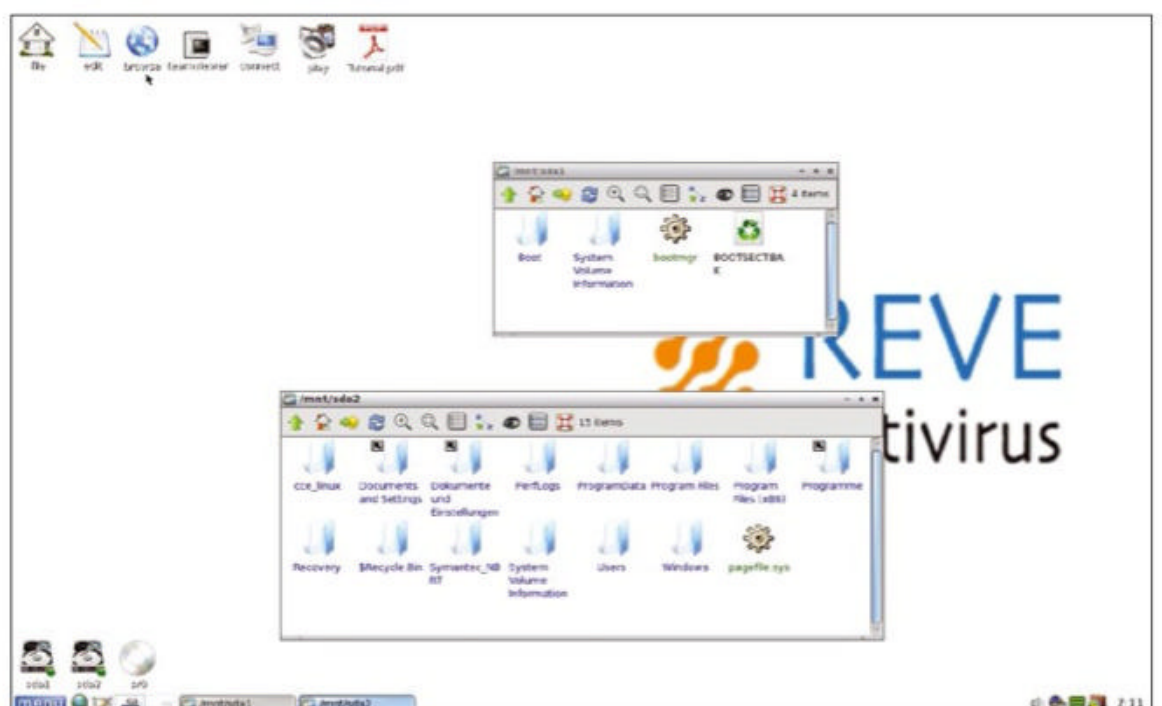


Figure 14: The REVE package is mainly suitable for backups.

Table 1: Live Rescue Systems

	Avira Rescue System	Comodo Rescue Disk	ESET SysRescue Live	Kaspersky Rescue Disk	Norton Bootable Recovery Tool	REVE Rescue Disk
System Linux	Ubuntu	Slitaz	Ubuntu	Gentoo	RHEL 6	Puppy Linux
Windows Rescue						
Own repair tool	Yes	Yes	Yes	Yes	No	-
Registry editor	Yes	No	Yes	No	No	-
Unlocker	No	No	Yes	No	No	-
USB recovery	No	No	Yes	No	No	-
Features						
Text mode	No	Yes	No	No	No	-
Installation possible	No	No	No	No	Yes	Yes
Linux Tools						
Terminal	Yes	Yes	Yes	Yes	Yes	Yes
Web browser	Yes	Yes	Yes	Yes	Yes	Yes
Graphical partitioning	Yes	No	Yes	No	No	Yes

When you launch a web browser from the panel bar at the bottom of the screen, a selection dialog opens where you choose the one you want to use.

Because the REVE package completely takes over the hardware detection and system maintenance tools usually included in Puppy Linux, including the backup routines, it is also suitable for locating hardware problems. However, it remains incomprehensible why the developers have also left numerous multimedia applications and games in the distribution – this does not sit well with the idea of a cross-platform rescue system.

Conclusions

The various rescue systems impressively prove that Linux is brilliantly suited for rescuing and recovering third-party operating systems (Table 1). Most of the distributions discussed

here use proprietary vendor tools to remove malware from a compromised Windows system.


The tools that step slightly out of line are REVE Rescue Disk, which can genuinely back up data and even fix hardware problems on all operating systems with a slightly modified Puppy Linux, and Avira Rescue System and Norton Bootable Recovery Tool, whose repair tools only work with Windows systems.

ESET SysRescue Live scans email files for malware and is thus useful for scanning Linux mail servers. As a unique feature, the Comodo Rescue Disk offers a text mode in which the most important functions can be used without a graphical desktop.

For administrators of heterogeneous environments, this selection leaves virtually nothing to desire: Simply choose the rescue distribution that is the best fit for your requirements. ■

Info

- [1] Avira: [<https://www.avira.com>]
- [2] Avira Rescue System: [<https://www.avira.com/us/start-download/product/2275/-l12e2mvtstLKUkGnG-U3eKJ0u0Ef8c>]
- [3] Comodo: [<https://www.comodo.com/>]
- [4] Comodo Rescue Disk: [<https://www.comodo.com/business-security/network-protection/rescue-disk.php>]
- [5] ESET: [<https://www.eset.com/us/?intcmp=intrw>]
- [6] ESET SysRescue Live: [<https://www.eset.com/int/support/sysrescue/>]
- [7] Kaspersky Lab Rescue Disk: [<https://www.kaspersky.com/downloads/thank-you/free-rescue-disk>]
- [8] Kaspersky Lab: [<https://www.kaspersky.com/>]
- [9] Norton Bootable Recovery Tool: [<https://support.norton.com/sp/static/external/tools/nbrt.html>]
- [10] REVE Antivirus: [<https://www.reveantivirus.com/en>]
- [11] REVE Rescue Disk: [<https://www.reveantivirus.com/en/download>]

The background of the entire page is a photograph of a modern, multi-level library. The library features white bookshelves filled with books, a wide staircase with a white railing, and a bright, open space with a blue armchair. The image is overlaid with a large, semi-transparent blue wave graphic that curves across the middle of the page.

Hone your skills with special editions!

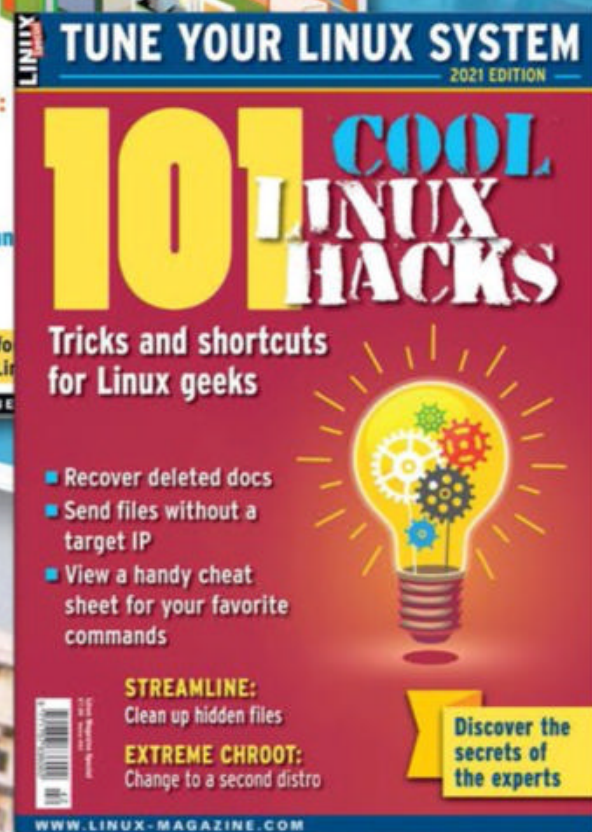
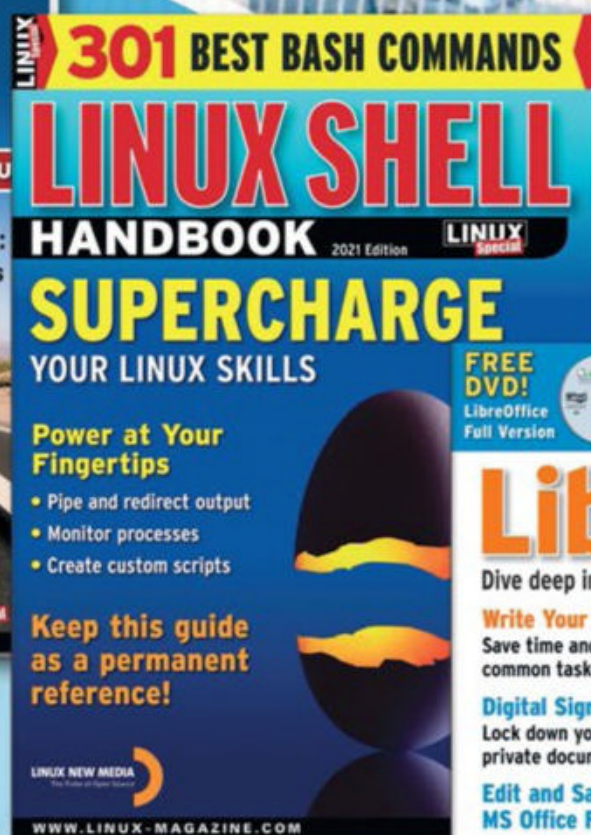
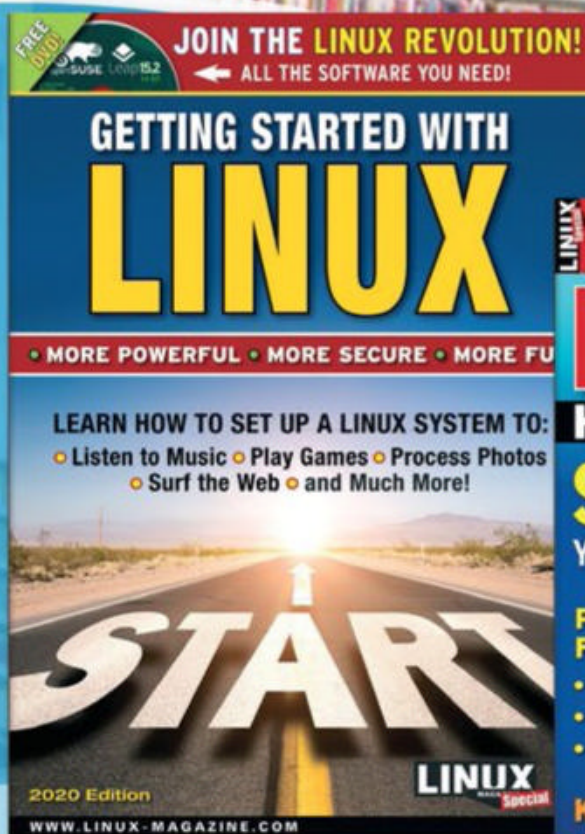
Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

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





SALE

Fast and scalable ownCloud Infinite Scale

Everything Must Go

A complete rebuild of ownCloud delivers the speedy and scalable ownCloud Infinite Scale file platform. By Martin Loschwitz and Markus Feilner

The ownCloud Infinite Scale (oCIS) enterprise file-sharing and syncing software stack is a fast and scalable yet complete rewrite of the classic ownCloud. The current version of oCIS is the technology preview 1.8.0 (as of July 2021). Go, Vue.js, and a modular approach replace the outdated PHP, database, and POSIX file-system the older versions needed. In internal documents, ownCloud promises up to 10 times better performance than ownCloud 10.6.0 published in December 2020 and, so far, the last PHP-based ownCloud variant.

Some History

A shooting star in 2010, ownCloud hasn't make it into the news that much recently, other than with its fork Nextcloud [1]. Whoever thinks that might be attributed to Nextcloud's media success and big community misses the point, though. Over the years, the two companies that once were one have developed in completely different directions. Nextcloud focused on its community and web front end, a web app store with many community apps targeting the growing market of web desktops like Microsoft 365, whereas

ownCloud quietly improved the back end and dealt with enterprise needs of (not just) their customers by refactoring the software and making it faster and more robust.

In hindsight, it almost looks like shared work, but it never was. After all, some of these different expectations inside management were why the company broke up in 2016 – but that is another story. Today both companies claim to be very successful with their approaches, and jokes have it that both are convinced they are light years ahead of their competitor.

PHP vs. Performance

Now in mid 2021, a big change is about to come. ownCloud is diverging from the common path and presents a complete rework of the back end. Over the years, the ownCloud developers had to realize the limits of the PHP architecture, especially in large setups with the need for massive scalability (e.g., CERN). Version 1 of oCIS [2] is the result of years of work: a complete rewrite of the software, a new back end in Go, a front end in Vue.js, and other changes (e.g., removing the need for a database).

When ownCloud was founded, PHP seemed to be the natural choice because Ruby, Go, Vue.js, and reactive design were not around then. PHP dates back to 1995, and even the fifth major release, PHP 5, was eight years old when ownCloud finally left stealth mode. Developers have invested a lot of work to adapt the dinosaur to modern software principles and to improve its performance, as have admins. The basic design and architecture of PHP, however, has remained the same throughout the years, which effectively means that a number of almost mandatory modern day software principles are not honored.

Scalability Without PHP and POSIX

A further ownCloud requirement was the need for a separate database in the background, a web server hosting PHP content, and a filesystem for storage. To be more precise, ownCloud has been centered around the idea of having a POSIX-compatible filesystem to store data uploaded by users, different versions of these files and the trash files, as well as configuration files and logs. By default, all files belonging to users can be found in a path like `/var/www` or `/srv/www` on the ownCloud instance – a web server's document root.

Photo by Markus Spiske on Unsplash

Admins who have ever called an ownCloud or Nextcloud their own know that ownCloud setups grow massively in size. Today, most of them even begin at much bigger sizes than ownCloud was originally planned to serve. One of the largest ownCloud users today, an Australian university is storing the data of more than 100,000 users.

oCIS is the company's attempt to break free of all the barriers and limitations of the PHP past. The paradigm shift goes deep: The last years have been spent fixing and adapting the existing codebase to squeeze a bit more speed out of the system. All suffering admins, users, and developers may now see a light at the end of the tunnel: oCIS does not recycle any code from ownCloud; it's a complete rewrite.

Let's Go

The programming language Go brings in a large number of advantages (e.g., concurrency). Most computer programs, even if they appear to be one monolithic piece of code from the outside, are split into different components internally. Web servers (e.g., the Apache web server, which is usually deployed with ownCloud) are excellent examples. Internally, they usually have a function handling TCP/IP connections; another function might be responsible for handling SSL, and yet another piece of code might execute the requested PHP files and deliver the results to the end user. All of those events happen, however, in a certain order. PHP and Apache simply can't handle concurrency: The individual steps of handling requests are done one after another, not at the same time, even if it was logically possible. For a client request to be served, multiple steps must be performed. Any environment with concurrency will allow all those events to happen at the same time instead of serially. Servers that are capable of handling requests in parallel will have to wait less for processes to finish their work, thus offering faster results to the user. By the way, this feature is one of the reasons why Go has become one of the de

facto standards for containerized microarchitecture applications.

A More Modular ownCloud

ownCloud is adapting to an architecture centered around the principle of microservices. oCIS splits into three tiers: storage, core, and front end. In this article, we look at each of these tiers separately. Even though for clients who upload and download files all that counts is the overall performance (including bandwidth), choosing the right underlying storage is crucial. In its three-tier model, ownCloud considers the storage available to the system to be the lowest tier. Along with performance goes scalability: Large ownCloud instances must be able to cope with the load of thousands of clients and must allow additional disk space if the existing space fills up. Like so many concepts, object stores and scalable storage wasn't as available in the past when ownCloud was initially designed as it is today, which is yet another good argument to review oCIS. Administrators today are used to having more choices, and ownCloud permits outsourcing the handling of physical storage devices to an external solution. Although S3-based object storage, storage based on the Samba service, or POSIX-compatible filesystem options continue to be supported, the preferred way to deploy oCIS is the way of Earth Observing System (EOS) Storage [3].

EOS to the Rescue

Originally provided by CERN in Geneva for its Large Hadron Collider experiment, EOS is optimized to feature very low latency when accessing files. It provides disk-based storage to clients by the XRootD [4] framework but also permits other protocols to access files. ownCloud uses the EOS HTTP protocol extension to talk to the storage solution (over HTTPS, of course). Far more interesting, though, is that it allows almost infinite scalability. For instance, at the time of writing this article, the CERN setup

featured more than 200PB of disk storage and continues to grow. By turning to EOS, ownCloud got rid of a number of shortcomings: EOS does not have a typical single point of failure, and all relevant services are run redundantly, including its ability to scale out and add additional instances of all existing services. EOS promises never to run out of disk space and comes with built-in redundancy for the data stored therein.

Other Options

Consequently, at least for large environments, ownCloud will expect the administrator to deploy an instance of EOS along with oCIS. The trade-off is clear: In exchange for the burden of having to maintain a separate storage system, you get the benefit of not having to worry about scalability and performance of the oCIS instance. This assumption also provides a hint about the assumed use case for oCIS and ownCloud: ownCloud's strategy with oCIS targets big data centers. For SoHo and end-user setups, EOS will, however, likely be over the top and much more complex than what the average end user is willing to maintain, let alone a lack of appropriate hardware. Smaller setups are served by oCIS through the framework by enabling the aforementioned support for Amazon S3, Samba, and POSIX-compatible filesystems, which is possible because EOS is not hard-coded into oCIS. Reva [5] on the one hand can't provide the same feature set as EOS but will accomplish most of the needs of end users and non-EOS-sized setups. Its strategic importance if EOS, on the other hand, shows when considering that only the combination of EOS and oCIS permits the full feature set of ownCloud.

Microservice Core

The second tier of oCIS is (thanks to Go) more of a collection of microservices than a core. Each and every microservice is responsible for a single task in the background (e.g., scanning for viruses). Basically all functionality

provided by oCIS is the direct result of the work of a specific microservice, like authenticating requests with OpenID Connect against an identity provider. In the end, that makes connecting existing user directories (e.g., ADFS, Azure AD, or LDAP) to ownCloud a simple task. For those who do not have an existing identity provider, ownCloud ships its own instance, effectively making ownCloud maintain its own user database.

Brand New User Interface

The third tier of oCIS is what the vendor calls Phoenix or ownCloud Web [6]. The completely rewritten user interface is based on the Vue.js JavaScript framework. Just like the oCIS core itself, Phoenix is written according to microservice principles and hence allows for better performance and scalability. The developers also used the opportunity to give the web interface a makeover; compared with the previous ownCloud versions, the oCIS web interface looks reduced and slick.

The oCIS developers did an impressive job regarding compliance with modern software design principles. The fundamental problem in building applications while following the microservice approach is usually how to make the individual components of the environment communicate with each other. APIs come to the rescue, but then every microservice component must have its own, well-defined API interface.

Luckily, a number of tools are available in the wild taking that burden off the shoulders of developers, most notably Google remote procedure calls (gRPC). The basic idea behind gRPC is to have a set of predefined APIs that allow actions in one component to be triggered from within another.

Traefik

The application design brings about some challenges to the underlying network, and the oCIS developers have chosen the Traefik framework [7] to tackle those. Traefik automatically load balances between the different

instances of microservices, takes care of automated SSL encryption, and allows for the additional deployment of firewall rules.

The split between the back end and the front end add additional advantages to oCIS. In fact, actions triggered by the user by means of ownCloud Web are completely decoupled from the ownCloud engine performing the respective task in the back end. If a user manually starts a virus check on their file(s) stored in ownCloud, they do not have to wait for the check to be finished. Instead, the check itself happens in the background, and the user only ever sees the result once the check is completed – the principle of concurrency at work.

Extensions as Microservices

Like other web services, ownCloud also supports extending its capabilities with extensions. oCIS doesn't change that, but it is promising to tackle a well-known problem, especially with community apps. Apps of unknown origin can cause trouble in the server and hamper updates and often have negative effects on the server's overall performance. The new oCIS gRPC-based architecture makes it much easier to create additional extensions spawned alongside the already existing microservices. Because the API is predefined by gRPC, all developers really have to do is create a microservice featuring the desired functionality that can be controlled by gRPC. Traefik will, on a per-case basis, ensure that newly deployed add-ons are automatically added to the existing communication mesh. For users and third parties interested in ownCloud, this means that a flood of additional ownCloud functionality will likely start to pour in for oCIS.

Arbitrary Extensions

The oCIS developers make it very clear that they want to support such extensions, and they are even making it very easy for developers to integrate custom modules. Their docu-

mentation [8], for instance, contains a detailed explanation of how to write oCIS extensions, along with code for a real-world example.

Said documentation suggests that writing an oCIS extension is not super-complex for experienced Go jockeys. Three major components are required: the server providing the extension's functionality through a gRPC API, a so-called "greeter API" to serve requests to the extension from the outside, and, if desired, a separate component integrating the extension into ownCloud Web. The greeter API integrates with the oCIS server itself to allow commands to the server to be caught on the API of the oCIS instance. It communicates with the server through gRPC. The Hello World example [9] in the oCIS documentation illustrates this clearly: For instance, the command received through the oCIS API could be for *Hello World* to be returned as the result of an oCIS API request. The greeter API would catch that request, trigger the corresponding action over gRPC, and deliver the return value. The web interface can be used to send the *Hello World* request to the greeter API. Although a simple Hello World is an oversimplified example, the flow of commands brought up by oCIS developers also allows for highly complex operations to be performed by oCIS extensions. Of course, all oCIS extensions will still benefit from Go's concurrency functionality and its default optimization for performance purposes.

Goodbye MySQL!

ownCloud's switch to gRPC and microservices eliminates the need for a relational database. Instead, components in oCIS required to store metadata will do that on their own. With Reva and no MySQL dependencies, the complexity of running ownCloud in small environments is reduced considerably – a point interesting not just to large-scale data center admins.

Up and Running

ownCloud published a technical preview of oCIS 1.0 at the end of 2020,

shipping it as a Docker container and binaries. More examples on getting it running are linked in the deployment section of the GitHub repository.

The vendor's claim that getting the oCIS Docker container up and running is easy, is true, although things get complicated with EOS (see below). Docker images for oCIS are available on Docker Hub [10]; their latest tag always reflects the current master branch.

Any standard virtual machine with one of the big cloud providers or any entry server in the data center with a standard Linux distribution should be sufficient, provided the system has a container runtime installed. On openSUSE, for example,

```
zypper in docker
```

provides the needed container. One or both commands

```
service docker enable
```

```
service docker start
```

create the standard Docker environment.

Pulling the oCIS container and starting it is all that is required to get the server and ownCloud Web up and running with the default usernames (Figure 1):

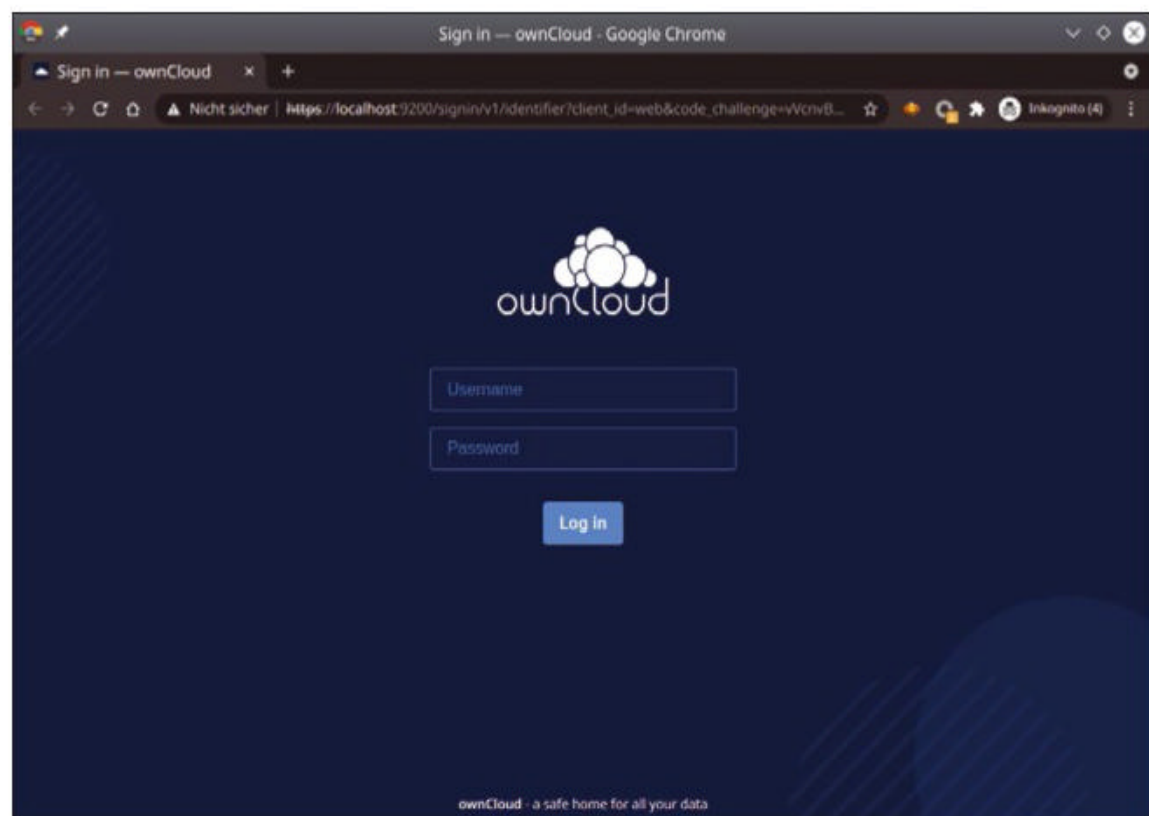
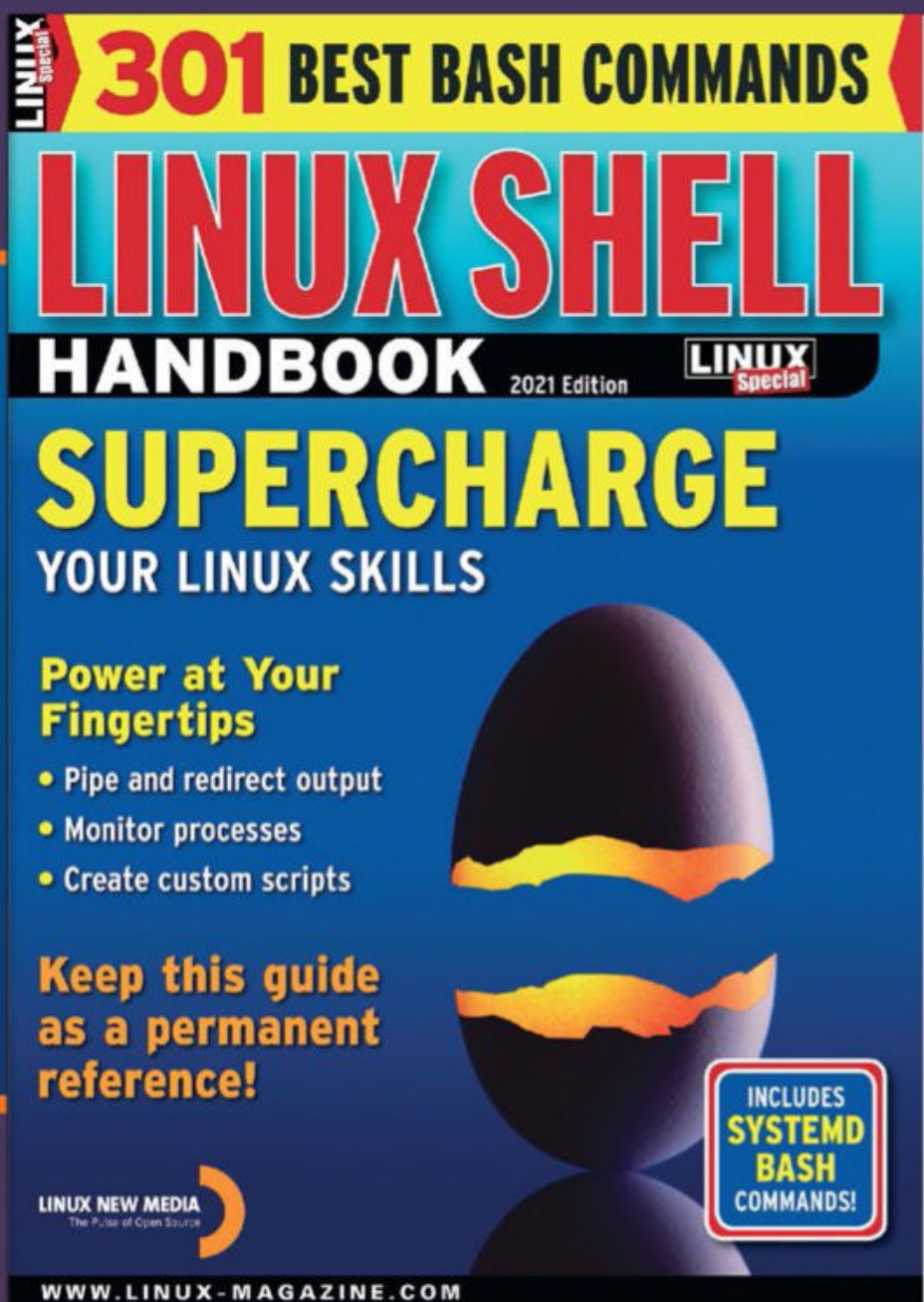


Figure 1: The oCIS login window after downloading and starting the Docker image.



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
- Bash Scripting
- Regular Expressions
- Networking Tools
- Systemd
- And much more!

ORDER ONLINE:

shop.linuxnewmedia.com/specials

Listing 1: Getting oCIS Binaries

```
# for Mac
curl https://download.owncloud.com/ocis/ocis/1.0.0/ocis-1.0.0-darwin-amd64 --output ocis
# for Linux
curl https://download.owncloud.com/ocis/ocis/1.0.0/ocis-1.0.0-linux-amd64 --output ocis
chmod +x ocis
```

```
# zypper in docker
# service docker start
# docker pull owncloud/ocis
# docker run --rm -ti \
    -p 9200:9200 owncloud/ocis
```

That’s it. oCIS is now at your service on localhost, port 9200 (*http://localhost:9200*). The demo accounts and passwords are *einstein:relativity*, *marie:radioactivity*, and *richard:superfluidity*. Admin accounts are *moss:vista* and *admin:admin*. If oCIS is run on a server with a resolvable hostname,

it can request an SSL certificate from Let’s Encrypt via Traefik.

oCIS Binary

An alternative to Docker is a pre-compiled binary that users – thanks to Go – can simply download from GitHub. The latest binaries from the master branch are in the *testing* section. Listing 1 downloads oCIS from the ownCloud download repository. The binary edition of oCIS expects */var/tmp/ocis* as the default storage location, but that can be changed in its configuration (Figure 2).

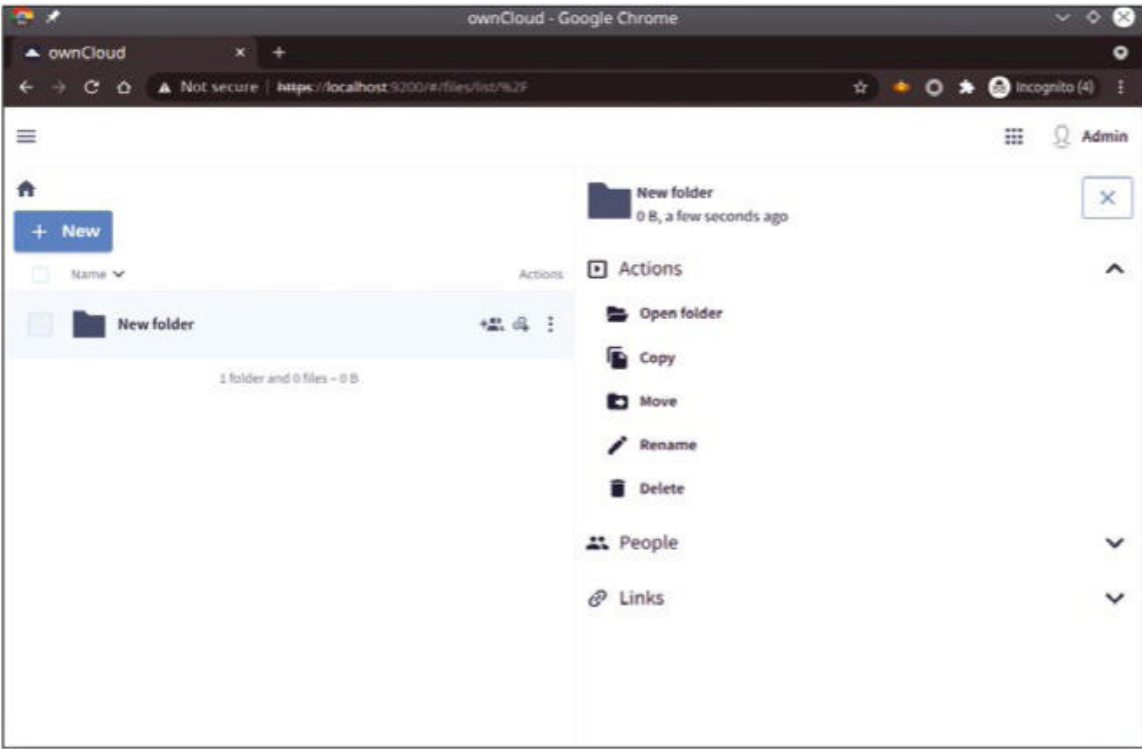


Figure 2: User management interface in the oCIS test machine.

Table 1: oCIS Commands

Command	Function
Management	
ocis server	Start the server
ocis --version	Print the version of the installed oCIS
ocis health --help	Execute a health check
Extensions	
ocis list	Print all running oCIS extensions
ocis kill web	Stop a particular extension
ocis run web	Start a particular extension
ocis --help	More information (e.g., the ownCloud repository) (Figure 3)

The server can be started with *./ocis server*. Basic management and extension commands are shown in Table 1. If the exit code of the health check equals zero, the service should be up and running. If the exit code is greater than zero, the service is not in a healthy state. The oCIS binary already contains some extensions that can be managed by oCIS commands.

EOS Is Complicated

If you want to follow ownCloud’s recommendations to deploy oCIS along with EOS for large environments, you have to take some more steps. EOS not only adds to the amount of required hardware and increases the complexity of the whole environment, it’s also a slightly bigger task to set up. To help, you will find separate and concise EOS documentation from CERN [3] and a step-by-step guide from ownCloud [11]. In a nutshell, users have to get and start EOS and oCIS containers; configure LDAP support; and kill home, user, and metadata storage before starting the EOS configuration. The Accounts service needs to be set up to work with EOS, too. All of these steps are docker compose commands, which are covered in the documentation. The *Storage backends | EOS* page in the oCIS documentation also provides information on verification and troubleshooting and has a command reference for the built-in EOS shell. Said documentation also gives an impression into why dealing with oCIS is likely interesting only for large oCIS environments.

Data Paths

Although it is relatively easy to imagine the flow of data through ownCloud 10 with Apache and PHP, the great number of components in oCIS makes imagining the path of data more difficult. ownCloud therefore provides a diagram that explains the flow of data in detail [12]. Before working with oCIS for the first time,

you should have a look at that document for a better understanding of data flow.

Initially, a file coming from a client will hit the oCIS proxy component, which is part of Traefik and ensures smooth load balancing across the different instances of oCIS components. The incoming request – in this example, a file upload – is then forwarded to one of the available Reva storage front ends.

From the administrator's point of view, it will look as if data disappears at this point. Reva comprises a number of microservices itself: a gateway for incoming requests, a DAV-speaking component that initially deals with incoming requests, and its own registry. The DAV component supplies the oCIS proxy with all relevant information for placing a file in the environment. Once Reva has successfully determined where the file needs to go, it reports that information back to the oCIS proxy, which finally puts the file into the desired location.

What looks like a side note here is in fact something very important to note: Just like ownCloud 10, oCIS continues to use the DAV interface for external clients. Therefore, any existing clients for ownCloud, including the project's very own iOS and Android applications, will continue to work with oCIS. It also allows

ownCloud to create a bridge mode between ownCloud 10 and oCIS, making migrations from one system to the other much easier.

oCIS Configuration

Administrators familiar with ownCloud 10 or earlier are used to the tedious configuration process, because it is an application running inside a web server with PHP. Parts of the settings required affect PHP directly and must go into the PHP configuration file. Other settings are Apache-related and must be in the web server's configuration file (even there, you have different options, e.g., the VirtualHost or httpd configuration). Moreover, some ownCloud-specific configuration files define parameters such as the database connection to use. Tuning ownCloud 10 to the maximum accordingly requires a number of configuration file changes. Because these settings can be different on different Linux distributions, automating ownCloud has been a tedious task in the past.

oCIS changes the scene. Because ownCloud no longer requires Apache or PHP, their configuration files are absent. Instead, you are expected to use the `ocis` binary as the central command to control behavior. Understanding the power of that tool

is tricky at first glance, though, so a closer look is necessary.

One Binary to Rule Them All

In oCIS, `ocis` is the universal binary. It starts the microcomponents that make up the oCIS core and the ownCloud Web interface. Additionally, it runs the components that allow oCIS to connect to EOS, if EOS is part of the setup. The use of microcomponents would be violated if oCIS were a huge binary incorporating all that functionality. Therefore, `ocis` is a skeleton that loads most of the functionality by starting the respective extensions. For instance, the ownCloud Web interface for the Vue.js-based GUI is such an extension. Consequently, for every extension oCIS can handle, you control the component's behavior either directly from the command line or by means of a configuration file.

If you decide to deploy configuration files, oCIS lets you choose between the YAML and JSON formats. A single file in `/etc/ocis` controls the behavior of the oCIS core and the `ocis` command itself. The individual extensions typically use their own file for determining configuration settings, but the details are left for the authors of the extensions to define. What is elegant though is that oCIS supports sharing configuration files between the host and the individual oCIS containers. Files in `/etc/ocis` will be present in any oCIS-related container, allowing for a seamless automation of oCIS setups.

A Massive Change in Handling

Overall, the introduction of the `ocis` CLI binary to control any and all aspects of ownCloud's behavior is by far the most invasive change for experienced ownCloud administrators and will make adapting to oCIS a bit more challenging – particularly for those who have to maintain ownCloud 10 and oCIS setups in parallel (for compatibility reasons or different use cases

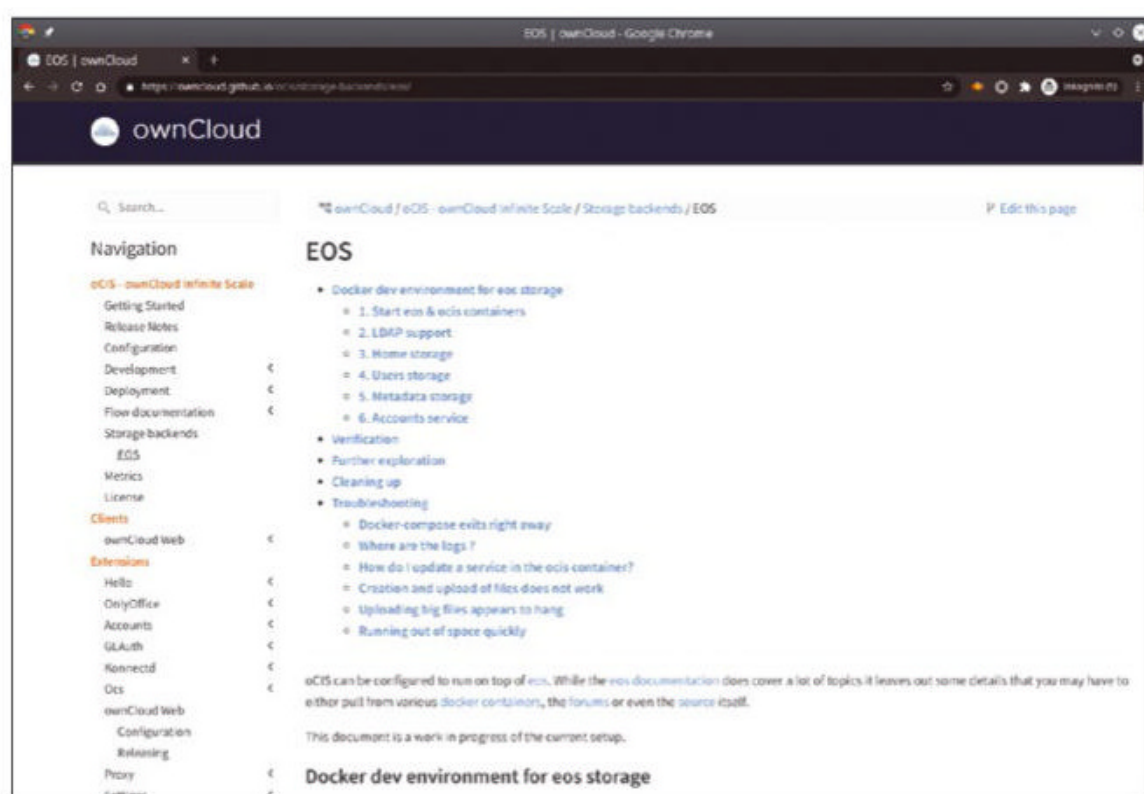


Figure 3: ownCloud supplies an overwhelming amount of documentation, especially when considering it is still a technology preview.

in the same company). If you are in this camp, you will have to cope with two very different configuration approaches until ownCloud 10 is retired completely.

Monitoring

Massively scalable environments like ownCloud bring new challenges in terms of monitoring. In conventional systems, monitoring usually refers to event monitoring only. Accordingly, the question a typical event monitoring environment will try to answer is whether a certain service at a certain point in time performs according to the desired state. For scale-out solutions such as ownCloud, that is simply not enough. In fact, the status of an individual component of the setup doesn't matter that much. Microservice approaches are inherently redundant – if one service of a certain type fails, the other services automatically take over that component's workload (gRPC as well as Traefik allow oCIS to follow this principle easily). What is much more relevant to the administrator is knowing ahead of time when additional storage space must be added and when additional servers for oCIS core components need to be commissioned. This challenge is not exactly new, environments such as Kubernetes or OpenStack have paved the way already and forced developers to come up with mighty solutions. Hence, monitoring usually is just one aspect of what administrators call monitoring, alerting, and trending (MAT). Solutions for MAT in scalable environments exist; Prometheus is one of the most prominent alongside InfluxDB. In oCIS, ownCloud developers add an interface for Prometheus to read out the most important metric data during normal operations.

Prometheus to the Rescue

Prometheus itself is a well-established standard by now. By following the pull principle, other components must make their metric data available through a pre-defined API interface to be fetched by Prometheus. If you enable the Prometheus interface in oCIS, every oCIS instance will allow the Prometheus instances of a setup to fetch metric data from port 8001 through the oCIS debug endpoint. Available metrics include a number of different values related to Go, different memory statistics, threads in use, and more of the like. Additionally, ownCloud recommends deploying Telegraf on oCIS servers for general-purpose system metrics, such as disk space in use, CPU usage, or other vital parameters. Combined with the Jaeger client, it even becomes possible to do in-depth performance debugging of individual oCIS components in case some parts of the environment do not live up to expectations. Long story short, MAT functionality is now at the core of oCIS and much more easily available than it was in previous ownCloud versions. Admins should absolutely leverage that new functionality.

Tough Call and a Risk to Take

The technical aspects of oCIS that are visible are already nothing short of spectacular. The software is very easily installed, and its completely redesigned components are faster than its predecessor and offer much better scalability. The modular design with microservices and APIs, even for extensions, looks promising. The biggest risk for ownCloud appears to be that, while taking big technological steps, they might lose a significant part of their

PHP-based community. Community management in terms of Go and Vue.js (and EOS) is surely a big task for the company now and in the years to come. ■

Info

- [1] Nextcloud: [\[https://github.com/nextcloud\]](https://github.com/nextcloud)
- [2] oCIS: [\[https://github.com/owncloud/ocis\]](https://github.com/owncloud/ocis)
- [3] EOS: [\[https://information-technology.web.cern.ch/services/eos-service\]](https://information-technology.web.cern.ch/services/eos-service)
- [4] XRootD: [\[https://xrootd.slac.stanford.edu/\]](https://xrootd.slac.stanford.edu/)
- [5] Reva storage engine: [\[https://github.com/cs3org/reva\]](https://github.com/cs3org/reva)
- [6] ownCloud Web: [\[https://owncloud.dev/clients/web/\]](https://owncloud.dev/clients/web/)
- [7] Traefik: [\[https://doc.traefik.io/traefik/\]](https://doc.traefik.io/traefik/)
- [8] oCIS documentation: [\[https://owncloud.dev/ocis/\]](https://owncloud.dev/ocis/)
- [9] Example extension: [\[https://owncloud.github.io/extensions/ocis_hello/\]](https://owncloud.github.io/extensions/ocis_hello/)
- [10] Docker Hub: [\[https://hub.docker.com/u/owncloud/\]](https://hub.docker.com/u/owncloud/)
- [11] ownCloud EOS docs: [\[https://owncloud.dev/ocis/storage-backends/eos/\]](https://owncloud.dev/ocis/storage-backends/eos/)
- [12] oCIS data flow: [\[https://owncloud.github.io/ocis/flow-docs/public-upload-flow/\]](https://owncloud.github.io/ocis/flow-docs/public-upload-flow/)

The Authors

Markus Feilner is a Linux specialist from Regensburg, Germany. He has worked with Linux as an author, trainer, consultant, and journalist since 1994. For many years, he has worked as deputy editor-in-chief of

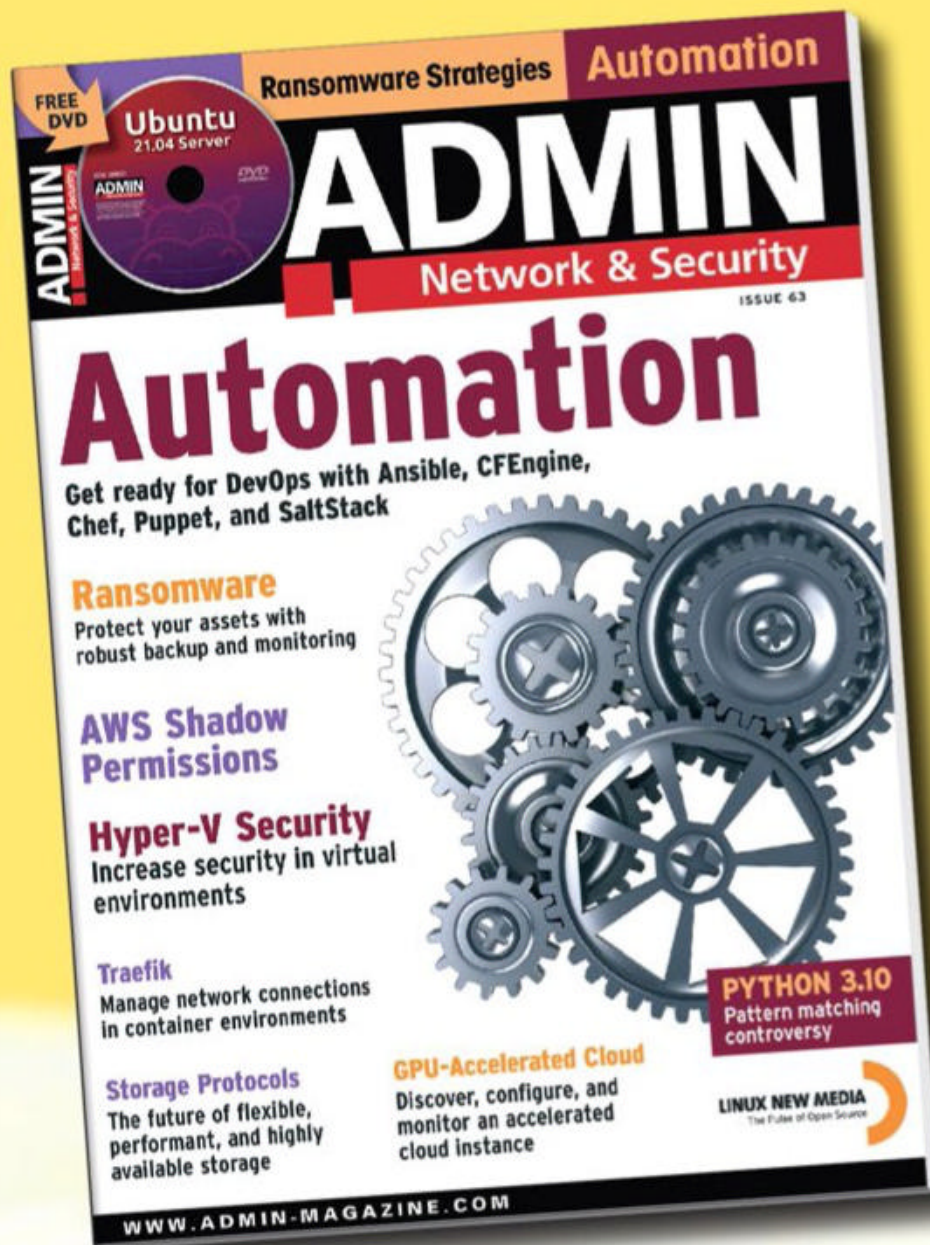
German Linux Magazine and Heise iX. His company Feilner IT helps companies and administrations with their OSS strategy, documentation and leadership consulting.



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



REAL SOLUTIONS *for* REAL NETWORKS



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

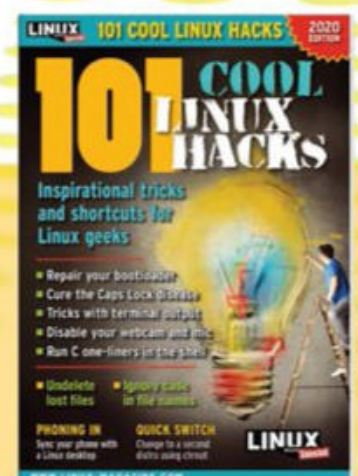
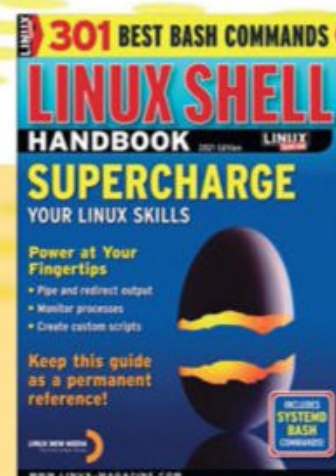
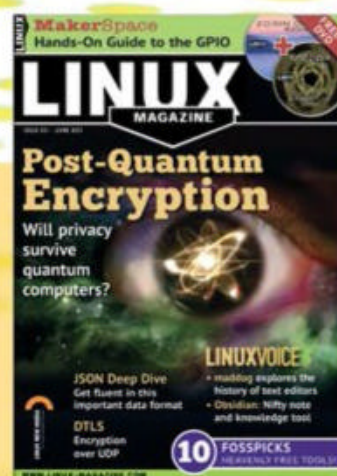
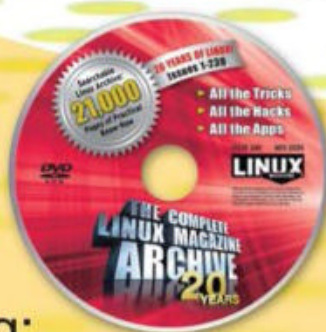
Improve your admin skills with practical articles on:

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



SUBSCRIBE NOW!
shop.linuxnewmedia.com

Check out
our full catalog:
shop.linuxnewmedia.com





Back up virtual machines and clusters

Flexible Security

Vembu BDR Suite provides comprehensive software that supports flexible configuration when backing up virtual production operations. By Thomas Joos

Vembu BDR Suite [1] helps you back up Hyper-V and VMware ESXi environments with up to 10 virtual machines (VMs) free of charge. In this article, I back up a Hyper-V host and start by installing an agent for the connection – the Vembu integration service (VIS) – that is also capable of backing up server-based services such as Exchange or SQL Server on the VMs.

You do not need to install any additional software on the virtual machines themselves. In addition to standalone Hyper-V hosts, you can also back up clusters with Vembu. For the backup, the vendor draws on the Hyper-V volume shadow copy service (VSS). The VMs are therefore included in the backup, instead of just copying the virtual hard drives and their data.

Vembu BDR Suite also relies on resilient change tracking (RCT) in Windows Server 2016 and 2019, a feature that allows backup software to detect which data has changed on the virtual disks since the last full backup so that only the changed data is saved. In this case, the virtual hard drive consists of blocks. If you perform a full backup, all blocks are backed up to a repository.

However, for the next backup, you will only want to back up the changed blocks. The change block tracker creates a bitmap of all the blocks on the virtual hard drive. As soon as the data in a block changes, it is marked accordingly. At the time of a backup, the backup software only needs to check the bitmap for all blocks that have changed since the last backup and then simply backs up the changed blocks.

This method has been possible without additional functions since Windows Server 2016 with RCT. RCT creates three bitmaps: one in memory and two on the hard disk, which ensures that the bitmaps are not lost. As long as the VM is running, the backup software checks the bitmap and only backs up the changed blocks when an incremental backup is requested. If a VM is migrated or a power failure occurs, the backup software still has two bitmap files from which to choose. The RCT file is used during normal operation. In case of a sudden power failure or a similarly serious event, the modified region table (MRT) file can also help. This table is maintained in write-through mode and has coarser tracking granularity. In the event of a sudden power failure,

the MRT file still provides a record of what has changed on the disk and guides the back up of data on recovery. The technology thus accelerates and simplifies incremental backups.

Functions and Limitations

Backups are easy to perform with the features included in the free edition of Vembu BDR Suite. As mentioned, the free version supports VMware ESXi and Hyper-V; you can also connect to vCenter with the free version. Up to 10 VMs are supported, and 10 EC2 instances can be backed up in Amazon Web Services (AWS). The same applies to services in Microsoft 365 and Google G Suite. The limit is 10 user accounts and their data. Backups of conventional Windows servers are also possible, as well as up to 10 workstations. You can also include Mac computers in addition to your Windows computers.

You can back up the VMs on Hyper-V hosts to local disks; CSV files, SMB shares, and Storage Spaces Direct are also supported. The free version is already capable of backing up applications in the VMs, including Exchange Server and SQL Server, among others, and can perform the backups online and delete transaction logs.

Lead image © ximagination, fotolia.com

If you are satisfied with the product overall but need to back up more than 10 VMs, you can opt for one of the commercial editions. The costs are from \$108 per CPU socket for the Standard Essentials edition to \$360 for the Enterprise edition. The VMs to be backed up

also need to be licensed. The prices are \$18 for the Standard Essentials edition and \$60 for the Enterprise edition.

Installing Vembu BDR Suite

The suite does not need to be installed on the Hyper-V host itself; instead, the connection and backup take place over the network. After the install, the functions of the Enterprise Edition can initially be used for 30 days, after which the software switches to Free mode. Free mode can also be enabled at any time beforehand; the settings for the editions can be found under *Management | BDR Edition*, which is where you can activate the functions of the Free Edition immediately after the install, as well the functions of the other editions, provided you have licensed the product.

The Vembu BDR Suite management data end up in a PostgreSQL database that's installed on a backup server when the suite is installed and does not contain the data of the virtual hard drives themselves, only the information of the backup for recovery and management. During the install, you specify the web console port and the back support. The installation wizard displays the default data in the window. You can also specify the username for accessing the administration interface and the corresponding password during the installation. After you select from the various options, the wizard starts the installation. Afterward, you need to check in

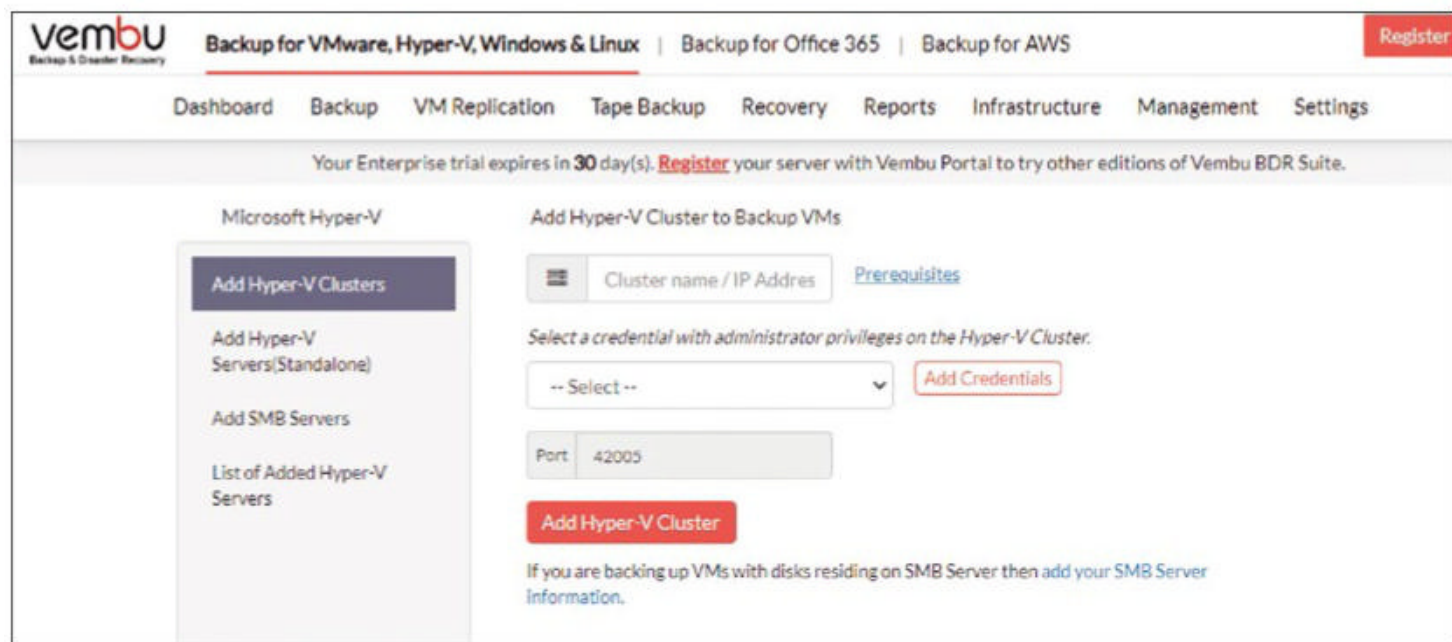


Figure 1: A Hyper-V backup in BDR starts with adding the virtual servers.

system services management whether the *VembuBDR* system service is running and whether the user account is correct. Parallel to the web interface, BDR also offers the Vembu agent interface on the server. From the context menu of the icon, you can restart the system service if necessary. The agent also displays messages telling you whether backups completed successfully or terminated with errors. These messages can also be disabled in the context menu.

To connect Hyper-V hosts, you need to start the *VembuBDR* system service with a domain account that is authorized to install an agent and perform tasks on the Hyper-V hosts. If the authorization is not sufficient, the wizard for connecting a new Hyper-V host will report an error to match.

Connecting the Hyper-V Host

Once the system service is started, you can open the web interface and log in with the user account you specified during installation. To back up Hyper-V hosts, first go to *Backup | Configure Backups | Microsoft Hyper-V*, which is where you add the servers that have the VMs you want to back up.

When adding Hyper-V hosts, use the *Add Hyper-V Servers* or *Add-Hyper-V Clusters* side tab to select the type of host you want to connect to (Figure 1). In the following, I assume you are setting up a standalone Hyper-V host because in most cases clusters have more than 10 VMs.

In the upper field, enter the name of the Hyper-V host. For *Add Credentials*, enter the credentials for accessing the Hyper-V host. For Hyper-V backup, an agent must be installed on the host (see the “Access Authorizations as a Source of Error” box). The various logins can be managed at any time under *Infrastructure | Credentials*. The item *Management | BDR Clients | List Client Agents* lets you view the currently connected servers, including the version of the client, in the course of connecting the host. You have to confirm the install agent prompt to connect the Hyper-V host.

Backing up Hyper-V

Once the host is connected, you can configure the backup with the *Configure Backup* button. However, you first need to define a repository to which the VM data will be backed up by se-

Access Authorizations as a Source of Error

To back up VMs on a Hyper-V host, communication between the Vembu BDR server and the Hyper-V host must be working. To install the agent, copy the installation file to the *admin\$* share of the host. If the Hyper-V host connection does not work, you need to check whether you are allowed to access the share from the Vembu server. If this does not work, you can use a checklist from Vembu [2] to track down the problem.

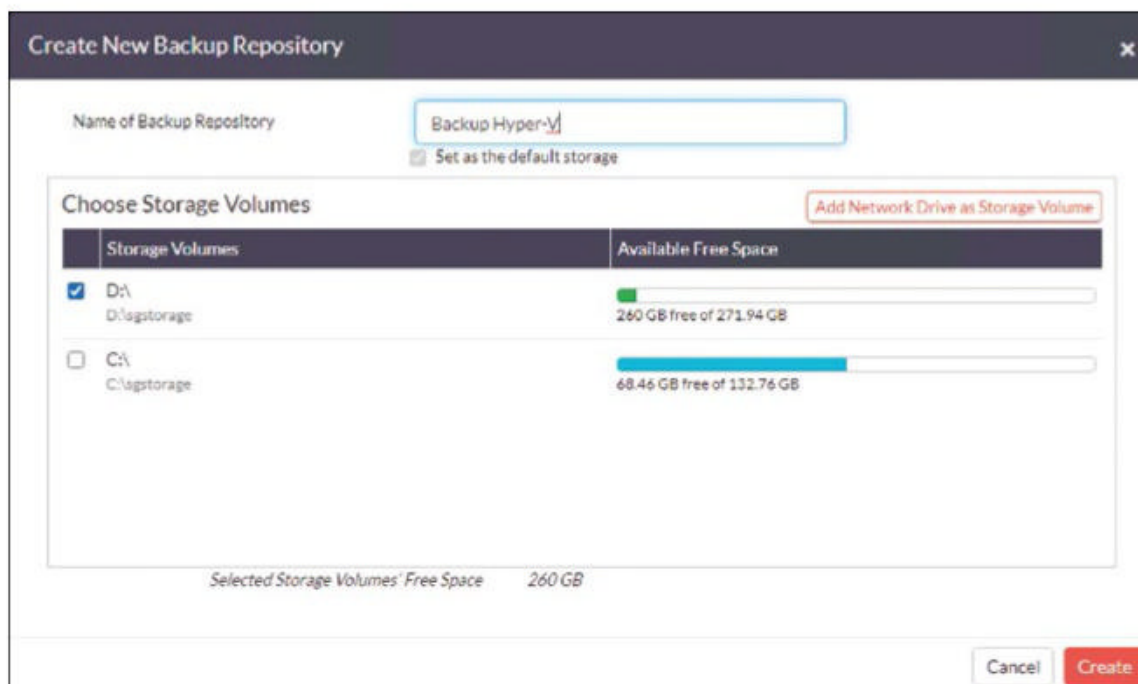


Figure 2: Setting up a backup repository for backing up VMs. Network drives are a good choice.

lecting *Infrastructure | Backup Repository | Create New Backup Repository* (Figure 2). Besides local data carriers, network shares can be used.

Of course, you can also create multiple backup targets, which you define later in the backup jobs. Under *Infrastructure | Backup Repository* you can view the individual targets, as well as the free storage space. Once you have created at least one backup repository, use *Configure Backup* to create a backup job for the individual Hyper-V hosts that are connected to Vembu. When using the free version, you can choose whether you want to use all backup features for a maximum of 10 VMs or whether limited features are sufficient for more VMs. With the limited features, only full backups are created, and you have to delete the backups manually. Retention is disabled in the free version when you back up VMs with the standard features.

On the first page of the wizard (VM Selection), you select the VMs to be backed up. You can see which VMs are running or currently switched off. Use the *VM | Disk Exclusion* button to remove individual virtual disks from the backup jobs. By default, Vembu BDR backs up all virtual disks from all selected VMs.

Under *Guest Processing*, you can also back up the applications on the VMs by activating *Guest Processing Settings*. In this way, you can, for example, create an online backup of

Exchange Server and your SQL databases. The software can delete the transaction logs after the backup if required, but first you need to enter the credentials for accessing the virtual server applications on the configuration page. If you already used these credentials when connecting the Hyper-V hosts and the user account is the same, you can select it directly from the menu.

Next, you need to decide when the servers will be backed up from the numerous options. Daily and weekly backups are possible, as are multiple daily backups. You can also specify

that full backups always take place at certain times and how many full backups Vembu will keep in the repository at the same time.

The Settings item lets you define when the backups will expire and be deleted from the repository. Additionally, multiple incremental backups can be grouped by specific time periods, which will significantly reduce the number of restore points. Finally, you can select the backup repository and enable encryption. The last step is to display a summary of the individual entries and give the backup job a name. By default, the backup is started immediately after saving, but this can also be disabled. If you click on the status of a backup, you will see the details of the current operation.

Managing and Monitoring Backups

In the web interface, you can see a summary of the currently active and processed jobs on the dashboard (Figure 3). *Backup | List Backup Jobs* shows all the defined backup jobs and the current status of the backup. If a backup is running, the list of backup jobs displays here. This is also where you suspend and resume jobs. If necessary, you can cancel running backup jobs from the Status item.

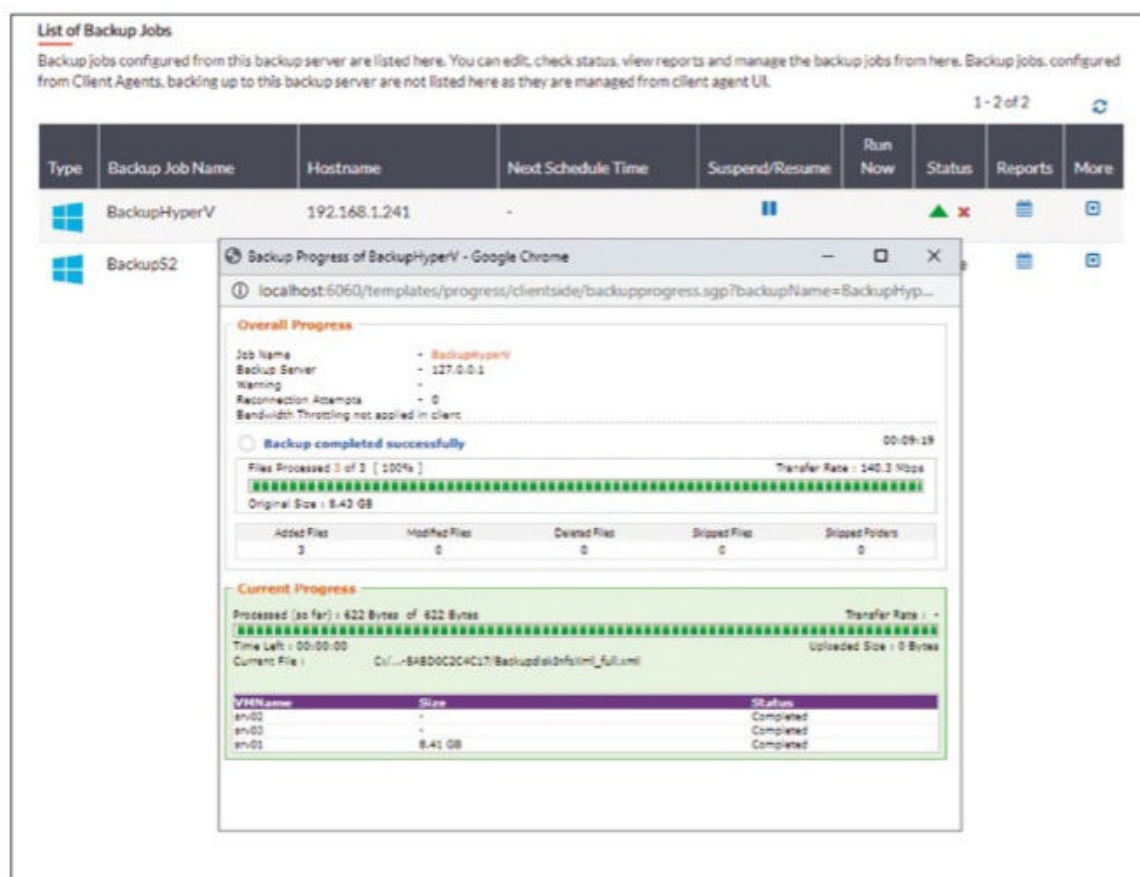


Figure 3: The backup status gives you information on its progress.

Under Reports, you can see descriptions of the latest job runs and whether the backups were successful. The volume of data for the job and possible errors, warnings, and other information can also be found here. Click on a job to view more detailed information, including the VMs to be backed up, the excluded disks, the schedule, the sequence of backups, and the disk encryption configuration.

If data is lost, you can restore VMs and individual data from the VMs from the Recovery menu item. To do this, select the icon for recovery and then specify whether you want to recover a complete VM or individual servers. When you start the recovery process, you have numerous options to choose from, including an instant recovery process, a full recovery, individual data, and even an image. Once the wizard is done, follow the prompt, and recovery will begin with

the specified options. Again, clicking on *Status* will give you more detailed information.

Conclusions

Vembu BDR Suite is quite an interesting piece of software for companies that want to back up VMs – up to 10 for free. Because the free version also supports backing up conventional servers, it lends itself to flexible use. Vembu BDR Suite is easy to manage in the web interface, and anyone who needs more than the features of the free edition can easily switch to a larger, commercial version by purchasing a license, with no need to reinstall; the backup data and settings are also transferred.

If you do not use Hyper-V on your network, you can also back up VMware ESXi. The procedure is identical, and the interface is the same. Only the approach for connecting

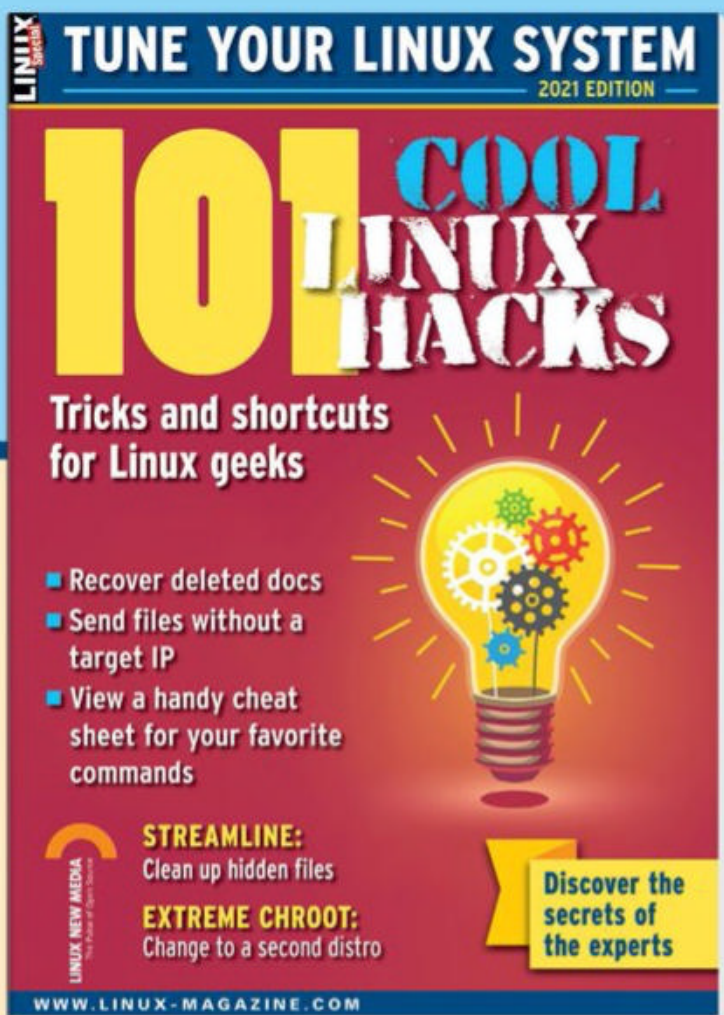
the hosts is slightly different. In the same interface, you can also back up conventional Windows servers, as well as data in the cloud – also free of charge. ■

Info

- [1] Vembu BDR Suite: [<https://www.vembu.com/free-hyper-v-backup>]
- [2] Checklist from Vembu: [<https://www.vembu.com/support/knowledge-base/question/117008/unable-to-connect-to-the-vembu-integration-service-in-the-target-host-check-network-connection-status-between-the-vembu-bdr-backup-server-and-the/>]

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>].



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

Kubernetes clusters within AWS EKS

Cloud Clusters

Automated deployment of the AWS-managed Kubernetes service EKS helps you run a production Kubernetes cluster in the cloud with ease.

By Chris Binnie

When a Kubernetes laboratory environment is required, the excellent localized Minikube and the tiny production-ready k3s distributions are fantastic. They permit a level of interaction with Kubernetes that lets developers prove workloads will act as expected in staging and production environments.

However, as cloud-native technology stacks lean more on the managed service offerings of Kubernetes, courtesy of the popular cloud platforms, it has become easier to test more thoroughly on a managed Kubernetes service. These managed services include the Google Kubernetes Engine (GKE) [1], Azure Kubernetes Service (AKS) [2], and Amazon Elastic Kubernetes Service (EKS) [3].

In this article, I walk through the automated deployment of the Amazon Elastic Kubernetes Service (EKS), a mature and battle-hardened offering that is widely trusted for production workloads all over the world by household names. For accessibility, suitable for those who do not want to run their own clusters fully, EKS abstracts the Kubernetes control plane away from users completely and provides access directly to the worker nodes that run on AWS Elastic Compute Cloud (EC2) [4] server instances.

In the Box

By concealing the Kubernetes control plane, EKS users are given the luxury of leaving the complexity of running a cluster almost entirely to AWS. You still need to understand the key concepts, however.

When it comes to redundancy, the EKS control plane provides a unique set of resources that are not shared with other clients – referred to as a single tenant – and the control plane also runs on EC2 instances that are abstracted from the user. The control plane of Kubernetes typically includes the API server etcd [5] for configuration storage, the scheduler (which figures out which nodes aren't too busy and can accept pods), and the Controller Manager (which keeps an eye on misbehaving worker nodes). For more information about these components, along with a graphical representation, visit the official website [6].

In addition to hosting worker nodes on EC2 across multiple availability zones for resilience (which are distinctly different data centers run by AWS at least a few miles away from their other sites), the same applies to the abstracted EKS cluster's control plane. Positioned in front of the worker nodes, which will serve your applications from

containers, is an Elastic Network Load Balancer to ensure that traffic is evenly distributed among the worker nodes in their Node group.

In basic terms, EKS provides an endpoint over which the API server can communicate with kubectl, in the same way that other Kubernetes distributions work. Also, a unique certificate verifies that the endpoint is who it claims to be and, of course, encrypts communications traffic. If required, it is possible to expose the API server publicly, too. AWS documentation [7] explains that, by default, role-based access control (RBAC) and identity and access management (IAM) permissions control access to the API server, which by default is made public. If you opt to make the API endpoint private, you restrict traffic to resources that reside inside the virtual private cloud (VPC) into which you have installed your EKS cluster. However, you can also leave the API server accessible to the Internet and restrict access to a specific set of IP addresses with a slightly more complex configuration.

Although not the norm, you can also have transparent visibility of the cluster's worker nodes. With the correct settings, you can log in over SSH to tweak the Linux configuration, if necessary. However, such changes should really be made programmatically; therefore, the use of SSH is usually just for testing. In this cluster creation example, I'll show the setting to get

Lead image © Zlatko Guzmic, 123RF.com

you started, plus how to create a key pair, and you can experiment with security groups and other configurations if you want to pursue the SSH scenario.

The marginal downside of using a platform as a service (PaaS) Kubernetes offering is that, every now and again, a subtle version upgrade by the service provider might affect the cluster's compatibility with your workloads. In reality, such issues are relatively uncommon, but bear in mind that by relying on a third party for production service provision you are subject to upgrades with much less control. Far from just applying to EKS, where you are encouraged to keep up with the latest Kubernetes versions as best as you can, many of today's software applications are subject to the chase-the-leader syndrome. From a thousand-foot view, EKS is designed to integrate easily with other AWS services such as the Elastic container registry (ECR), which can store all of your container images. As you'd expect, access to such services can be granted in a granular way to bolster security.

Now that the core components of the Kubernetes cluster aren't visible to users in EKS, take a look at what you can control through configuration.

Automation, Circulation

Although you could use a language like Terraform to set up EKS with infrastructure as code (IaC), I will illustrate the simplicity of using the native AWS command-line interface (CLI). Note that it will take anything up to 20 minutes to create the cluster fully. Also, be aware that some costs are associated with running a fully blown EKS cluster. You can minimize the costs by decreasing the number of worker nodes. The billing documentation states (at the time of writing, at least, so your mileage might vary): "You pay \$0.10 per hour for each Amazon EKS cluster that you create. You can use a single Amazon EKS cluster to run multiple applications by taking advantage of Kubernetes namespaces and IAM security policies."

I use version 2 of the AWS CLI to remain current with the latest CLI, as you can see by the *v2* path in the EKS reference URL [8].

From the parent page of the EKS instructions, I focus on the page that deals with the `create cluster` command [9]. From that page, you can become familiar with the options available when creating a cluster.

The `eksctl` tool [10] from Weaveworks has been named the official AWS EKS tool for "creating and managing clusters on EKS – Amazon's managed Kubernetes service for EC2. It is written in

Go [and] uses CloudFormation" Although you could set up a Kubernetes-friendly YAML to configure a cluster, for the purposes here, you can use a set of commands to create your EKS cluster instead. Start by downloading `eksctl` from the website and then move it into your path:

```
$ curl --silent 2
--location "https://github.com/2
weaveworks/eksctl/releases/2
latest/download/eksctl_2
$(uname -s)_amd64.tar.gz" | 2
tar xz -C /tmp
$ ls /tmp/eksctl
$ mv /tmp/eksctl /usr/local/bin
$ eksctl
```

The final command produces lots of help output if you have the tool in your path.

As mentioned, SSHing into your worker nodes takes a little bit more effort to permit access by security groups, but Listing 1 has the setting in the cluster creation; then, you can create a key pair that you can use for SSH.

When prompted for a name, don't save the key in `~/.ssh`; instead, type `eks-ssh` (line 4) and then save the key in the `/root` directory for now. The path to the public key is shown to be `/root/eks-ssh.pub` (line 8), which is then used in the `--ssh-public-key`

Listing 1: Creating a Key Pair for SSH

```
01 $ ssh-keygen -b4096
02
03 Generating public/private rsa key pair.
04 Enter file in which to save the key (/root/.ssh/id_rsa): eks-ssh
05 Enter passphrase (empty for no passphrase):
06 Enter same passphrase again:
07 Your identification has been saved in eks-ssh.
08 Your public key has been saved in eks-ssh.pub.
09 The key fingerprint is:
10 SHA256:Pidrw9+MRSPqU7vvIB7Ed6A11U1Hts1u7xjVEfiM1uI
11 The key's randomart image is:
12 +---[RSA 4096]-----+
13 |                .. 000+|
14 |                .  ...++|
15 |                . 0   =00|
16 |                . + . + =0|
17 |                S 0 * . =|
18 |                o o.+ E o.|
19 |                .B.o.. . .|
20 |                o=B.*  + |
21 |                .++++= . .|
22 +-----[SHA256]-----+
```

command in Listing 2 (line 11).

Now you're ready to create a cluster with the content of Listing 2. At this stage, I'm assuming that you have already set up an AWS access key ID and secret access key [11] for the API to use with the CLI.

Put the Kettle On

The command in Listing 2 will take a while, so water the plants, wash the car, and put the kettle on while you patiently wait for the cluster to be created. A word to the wise would be to make certain that you are in the correct region (*eu-west-1* is Dublin, Ireland, in my example). In the AWS Management Console, as you wait for your cluster,

Listing 2: Creating a Cluster

```
01 eksctl create cluster
02 --version 1.18
03 --name chrisbinnie-eks-cluster
04 --region eu-west-1
05 --nodegroup-name chrisbinnie-eks-nodes
06 --node-type t3.medium
07 --nodes 2
08 --nodes-min 1
09 --nodes-max 2
10 --ssh-access
11 --ssh-public-key /root/eks-ssh.pub
12 --managed
13 --auto-kubeconfig
14 --node-private-networking
15 --verbose 3
```

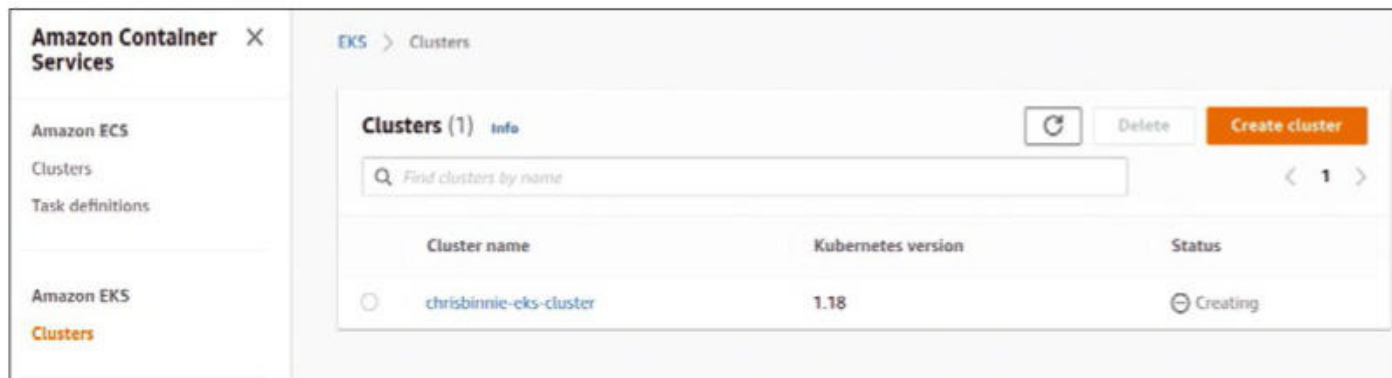



Figure 1: Happiness is a working Kubernetes cluster, when it is finally created.

look in the correct region for progress. In [Figure 1](#) you can see the first piece of good news: the creation of your control plane in the AWS Console (which is still in the *Creating* phase).

Note as the text in your CLI output increases that AWS is using CloudFormation (IaC, native to AWS) to create some of the services in the background, courtesy of `eksctl`. Incidentally, EKS v1.19 is offered as the cluster comes online, even though v1.18 is marked as the default version at the time of writing.

In [Figure 2](#), having waited patiently after the creation of the control plane, you can see the Node group has been set up, and the worker nodes are up and running within their Node group; you can scrutinize their running configuration, as well.

All Areas Access

Now, you need to authenticate with your cluster with AWS IAM authenticator:

```
$ curl -o aws-iam-authenticator \
  https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/aws-iam-authenticator
```

```
$ mv aws-iam-authenticator /usr/local/bin
$ chmod +x \
  /usr/local/bin/aws-iam-authenticator
```

Now run the command to save the local cluster configuration, replacing the cluster name after `--name`:

```
$ aws eks update-kubeconfig \
  --name <chrisbinnie-eks-cluster> \
  --region eu-west-1
Updated context arn:aws:eks:eu-west-1:XXXX:cluster/chrisbinnie-eks-cluster \
in /root/.kube/config
```

Now you can authenticate with the cluster. Download `kubectl` to speak to the API server in Kubernetes (if you don't already have it) and then save it to your path:

```
$ curl -LO "https://dl.k8s.io/release/$(\
  curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
$ mv kubectl /usr/local/bin
$ chmod +x /usr/local/bin/kubectl
```

This command allows you to connect to the cluster with a local Kubernetes config file. Now for the moment of truth: Use `kubectl` to connect to the cluster to see what pods are running across all

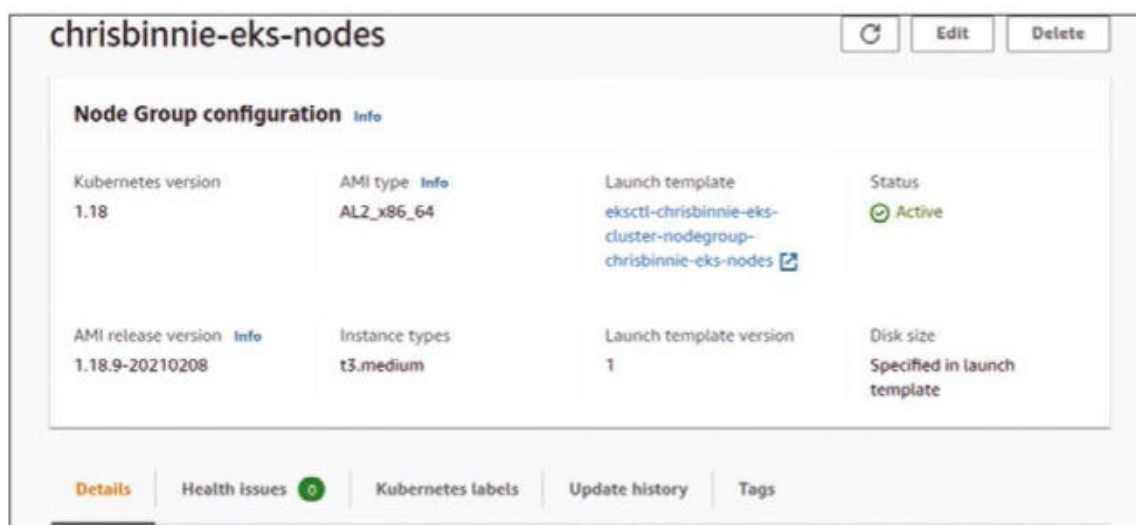


Figure 2: Some worker nodes are getting ready to serve workloads.

namespaces. [Listing 3](#) shows a fully functional Kubernetes cluster, and in [Figure 3](#) you can see the worker nodes showing up in AWS EC2. Finally, the EC2 section of the AWS Management Con-

sole ([Figure 4](#); bottom-left, under *Elastic IPs*) displays the Elastic IP address of the cluster.

Destruction

To destroy your cluster, you first need to wait for the Node group to drain its worker nodes by deleting the Node group in the AWS Console (see the “Node groups” box); then, you can also click the *Delete Cluster* button; finally, make sure you delete the Elastic IP address afterward. Be warned that the deletion of both of these components takes a little while, so patience is required. (See the “Word to the Wise” box.)

The End Is Nigh

As you can see, the process of standing up a Kubernetes cluster within AWS EKS is slick and easily automated. If you can afford to pay some fees for a few days, the process covered here can be repeated, and destroying a cluster each time is an

Word to the Wise

When you tear down the cluster (which you should get working programmatically), the Elastic IP address will continue to be billed to your account. That can total up to more than a few bucks in a calendar month if you forget, which might be a costly mistake. You should definitely check out the documentation [\[12\]](#), especially if you are new to AWS. Incidentally, deleting a cluster involves running a command similar to the `create cluster` seen in [Listing 2](#) [\[13\]](#). If you don't use the “`eksctl delete cluster`” option, to manually clear up all the billable AWS components, you should consider deleting the NAT gateway, network interface, elastic IP Address, the VPC (carefully, only if you are sure) and the cluster CloudFormation stacks.

Node groups

Although I've mentioned the worker nodes a few times, I haven't really looked at how EKS presents them for you to configure. EKS runs off vetted Linux 2 Amazon machine images (AMIs). It creates the maximum and minimum number of workers that you define dutifully yourself and then fronts them via an Amazon EC2 Auto Scaling group, so they can scale as you require automatically.

Note that not all Nodegroup settings let you SSH into the nodes for extended access. This access can be useful for troubleshooting, as mentioned, but also if the Docker Engine (potentially the CRI-O runtime instead) needs to have self-signed certificates ratified for tasks (e.g., connecting to image registries). You are advised to experiment with the levels of access your application requires. More official information is found on the [create-nodegroup](#) page [14].

Listing 3: Running Pods

```
$ kubectl get pods -A
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	aws-node-96zqh	1/1	Running	0	19m
kube-system	aws-node-dzshl	1/1	Running	0	20m
kube-system	coredns-6d97dc4b59-hrpjp	1/1	Running	0	25m
kube-system	coredns-6d97dc4b59-mdd8x	1/1	Running	0	25m
kube-system	kube-proxy-bkjbw	1/1	Running	0	20m
kube-system	kube-proxy-ctc6l	1/1	Running	0	19m

affordable way of testing your applications in a reliable, production-like environment.

A reminder that once you have mastered the process, it is quite possible to autoscale your cluster to exceptionally busy production workloads, thanks to the flexibility made possible by the Amazon cloud. If you work on the programmatic deletion of clusters, you can start and stop them in multiple AWS regions to your heart's content. ■

Info

- [1]

GKE: [\[https://cloud.google.com/kubernetes-engine\]](https://cloud.google.com/kubernetes-engine)
- [2]

AKS: [\[https://azure.microsoft.com/en-gb/services/kubernetes-service/\]](https://azure.microsoft.com/en-gb/services/kubernetes-service/)
- [3]

EKS: [\[https://aws.amazon.com/eks\]](https://aws.amazon.com/eks)
- [4]

Amazon EC2: [\[https://aws.amazon.com/ec2\]](https://aws.amazon.com/ec2)
- [5]

etcd: [\[https://www.etcd.io\]](https://www.etcd.io)
- [6]

Kubernetes components: [\[https://kubernetes.io/docs/concepts/overview/components\]](https://kubernetes.io/docs/concepts/overview/components)
- [7]

Amazon EKS cluster endpoint access control: [\[https://docs.aws.amazon.com/eks/latest/userguide/cluster-endpoint.html\]](https://docs.aws.amazon.com/eks/latest/userguide/cluster-endpoint.html)
- [8]

eks: [\[https://awscli.amazonaws.com/v2/documentation/api/latest/reference/eks/index.html\]](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/eks/index.html)
- [9]

create-cluster: [\[https://docs.aws.amazon.com/eks/latest/userguide/delete-cluster.html\]](https://docs.aws.amazon.com/eks/latest/userguide/delete-cluster.html)
- [10]

eksctl: [\[https://eksctl.io\]](https://eksctl.io)
- [11]

Setting up AWS keys: [\[https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html\]](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-quickstart.html)
- [12]

Amazon EC2 on-demand pricing: [\[https://aws.amazon.com/ec2/pricing/on-demand\]](https://aws.amazon.com/ec2/pricing/on-demand)

- [13]

delete-cluster: [\[https://awscli.amazonaws.com/v2/documentation/api/latest/reference/eks/delete-cluster.html\]](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/eks/delete-cluster.html)
- [14]

create-nodegroup: [\[https://awscli.amazonaws.com/v2/documentation/api/latest/reference/eks/create-nodegroup.html\]](https://awscli.amazonaws.com/v2/documentation/api/latest/reference/eks/create-nodegroup.html)

Author

Chris Binnie's new book, *Cloud Native Security* ([\[https://cloudnativesecurity.cc\]](https://cloudnativesecurity.cc)), teaches you how to minimize attack surfaces across all of the key components used in modern cloud-native infrastructure. Learn with hands-on examples about container security, DevSecOps tooling, advanced Kubernetes security, and Cloud Security Posture Management.

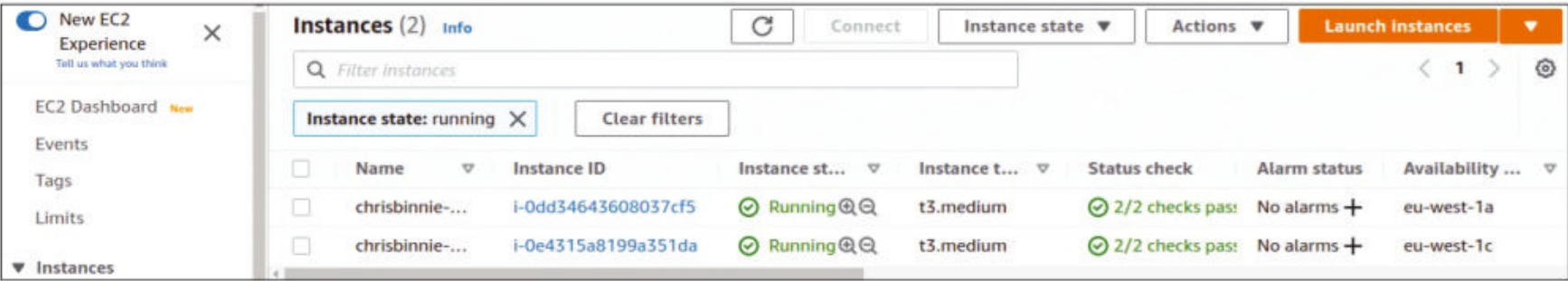


Figure 3: The worker nodes are seen in the EC2 section of the AWS Management Console.

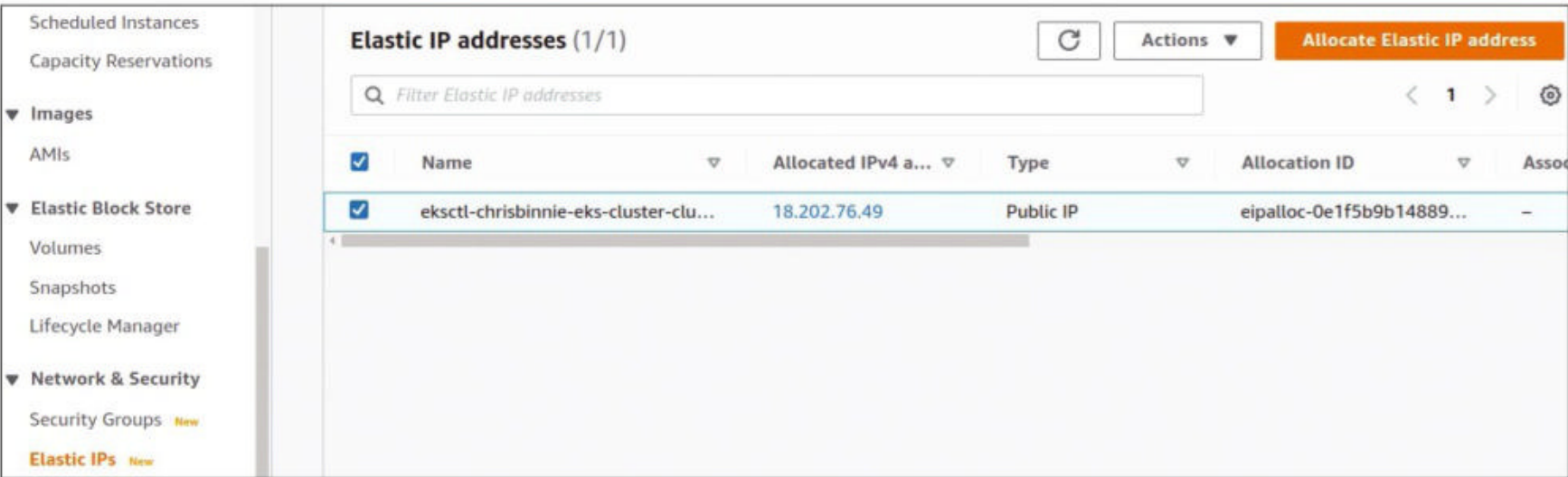


Figure 4: The Elastic IP address associated with the cluster is revealed in the AWS Console.



Run Kubernetes in a container with Kind

One of a Kind

Create a full-blown Kubernetes cluster in a Docker container with just one command. By Chris Binnie

Not often are you compelled to test unfamiliar software in the time it takes to read just a few lines on its website. Of course, the software has to be extremely accessible to allow you to get started quickly. An interesting application that recently fell into this category is Kind [1], a multinode Kubernetes distribution that is quick to install and configure for testing. The reason Kind was so intriguing is that it uses Docker containers for its nodes. Think about that for a minute and consider how versatile such an approach would be for quick testing. The clever Kind manages to squeeze a Kubernetes cluster inside containers.

In this article, I look at creating a freshly built Kind cluster installation to show off the infrastructure

created by Kind. You will see how one container runs an entire node – in this case, the control plane and master node.

Thank You, Kindly

Some really well constructed documentation about the Kind installation can be found as a Quick Start page online [2]. You should be the root user or prepend your commands with `sudo` for most of the commands that follow. If needed, install Docker CE first [3] (choosing the correct Operating System for your needs).

To begin, install `kubectl` so you can interact with your Kubernetes cluster in a coherent way. The Kind documentation points you to a page online

if you get stuck [4]. In my case, I just needed to run the following commands to download the binary file, check that it was safe, and download and check the checksum file:

```
$ curl -LO "https://dl.k8s.io/release/$(\
  curl -L -s https://dl.k8s.io/release/2\
  stable.txt)/bin/linux/amd64/kubectl"
$ curl -LO "https://dl.k8s.io/$(curl -L 2\
  -s https://dl.k8s.io/release/stable.\
  txt)/2\
  bin/linux/amd64/kubectl.sha256"
$ echo "$(<kubectl.sha256) kubectl" | 2\
  sha256sum --check
```

If the output from the `kubectl` command is *kubectl: OK*, move the binary into a directory in your path, set the permissions on the binary if you only

want root user access (skip this command if you've downloaded it into the /root directory), and allow other non-root users to run the binary:

```
$ mv kubect1 /usr/local/bin
$ chown root:root /usr/local/bin/kubect1
$ chmod 0755 /usr/local/bin/kubect1
```

If you are using a Debian derivative and want to use the package manager to install kubect1, start with the commands:

```
$ apt-get update
$ apt-get install -y apt-transport-https
ca-certificates curl
$ curl -fsSL /usr/share/keyrings/
kubernetes-archive-keyring.gpg \
https://packages.cloud.google.com/apt/
doc/apt-key.gpg
```

which will also add the repository key (ideally as one line). Check the documentation for Red Hat Enterprise Linux derivatives. In the /etc/apt/sources.list.d/kubernetes.list file, add the single line without splitting it up:

```
deb [signed-by=/usr/share/keyrings/
kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/
kubernetes-xenial main
```

Note that you might need to change xenial to match your Linux distribution name. Finally, install kubect1 after refreshing the sources:

```
$ apt-get update
$ apt-get install -y kubect1
```

Whichever route you take, once installed, test that you can use the command:

```
$ kubect1 version
...
Client Version: version.Info{
Major:"1", Minor:"21",
GitVersion:"v1.21.1"
...

```

The quick output is welcome. You can proceed if your result looks like the (abbreviated) output shown.

Be Kind To One Another

The next stage is of course installing Kind. Options include a Go module or downloading the source and running make build. You're advised to use the latest Go version if you do so. Check the documentation for more pointers about how to get started.

In this example, I download a binary, move it to the correct path, and check that it works from that path:

```
$ curl -Lo ./kind
https://kind.sigs.k8s.io/dl/v0.10.0/
kind-linux-amd64
$ chmod +x ./kind
$ mv ./kind /usr/local/bin/kind
$ kind --version
kind version 0.10.0
```

Get an incredibly quick start with the sophisticated functionality of Kind by creating a cluster ([Listing 1](#)). As soon as you run the command, you should see some clever whirring in the background with animated output as Kind gets a cluster ready.

Listing 1: Creating a Cluster

```
$ kind create cluster
Creating cluster "kind" ...
- Ensuring node image (kindest/node:v1.20.2)
- Preparing nodes
- Writing configuration
- Starting control-plane
- Installing CNI
- Installing StorageClass
Set kubect1 context to "kind-kind"
You can now use your cluster with:

kubect1 cluster-info --context kind-kind

Have a nice day!
```

Take the advice of the output in [Listing 1](#), and run the suggested command ([Listing 2](#)). The output is good news. If you run the command suggested at the end of the output, you receive a massive list of configuration data in JSON format that, as the output suggests, is excellent for debugging issues should you have any with your new Kubernetes installation.

As mentioned earlier, Kind is using Docker in a very clever way, as the

Listing 2: Starting Up the Cluster

```
$ kubect1 cluster-info --context kind-kind

Kubernetes control plane is running at https://127.0.0.1:34369
KubeDNS is running at https://127.0.0.1:34369/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
To further debug and diagnose cluster problems, use 'kubect1 cluster-info dump'.
```

Listing 3: docker ps (excerpt)

```
$ docker ps

IMAGE                STATUS    PORTS                               NAMES
kindest/node:v1.20.2 Up 6 minutes 127.0.0.1:34369->6443/tcp  kind-control-plane
```

Listing 4: kubect1 get nodes

```
$ kubect1 get nodes

NAME                STATUS    ROLES                  AGE    VERSION
kind-control-plane Ready    control-plane,master  21m    v1.20.2
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-74ff55c5b-jplvb	1/1	Running	0	14m
kube-system	coredns-74ff55c5b-r9hjm	1/1	Running	0	14m
kube-system	etcd-kind-control-plane	1/1	Running	0	15m
kube-system	kindnet-24lml	1/1	Running	0	15m
kube-system	kube-apiserver-kind-control-plane	1/1	Running	0	15m
kube-system	kube-controller-manager-kind-control-plane	1/1	Running	0	15m
kube-system	kube-proxy-46kfs	1/1	Running	0	15m
kube-system	kube-scheduler-kind-control-plane	1/1	Running	0	15m
local-path-storage	local-path-provisioner-78776bfc44-88bf9	1/1	Running	0	14m

Figure 1: A sentence that I never thought I'd write: Look at the Kubernetes cluster running in a container.

heavily abbreviated output of the `docker ps` command shows (Listing 3). What the example has done so far is bootstrap a Kubernetes cluster (with one node) with the node image supplied directly by Kind. You can look deeper at images online [5], and other images (check the release notes for the correct versions) let you choose the specific Kubernetes version you want to use. Although difficult to believe, the single container above is running a fully fledged Kubernetes cluster. You can check easily by running the command:

```
$ kubectl get pods -A
```

The `-A` is shorthand for `--all-namespaces`. Figure 1 shows the output is as expected and very welcome. The magic of Kind is clearly visible: A Kubernetes cluster running in a container! The output of the command in Listing 4 reveals that you have a control plane coupled tightly with a node acting as the master node in the cluster. The documentation continues with instructions on how to create multiple clusters with different names with `--name another-cluster` switch. Deleting a cluster is as simple as entering the delete command:

Listing 5: nginx Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
          ports:
            - containerPort: 80
```

```
$ kind delete cluster --name another-cluster
```

Make sure you use the `--name` flag if you don't want to delete the cluster named *kind*, which is the default name.

Kindly Advance

For more advanced configurations, see the kind documentation [6]. That section provides insight into setting up multinode clusters and opening host machine network ports so the cluster will allow applications to be accessed from outside of Docker. Additionally, you'll find a section in the documentation on "ingress" controllers [7] to get traffic into the cluster, and there's even a section about the use of Kind when Docker runs in rootless mode, without being the root user [8], which is the dream for Docker security and helps your container's security posture in numerous ways. The documentation is clearly constructed, so the pages referenced here and others are well worth a look.

So Kind

To prove that you have a Kubernetes build on which you can run workloads, install an nginx deployment. Listing 5 is the YAML configuration file. To ingest the YAML into Kuber-

netes, simply save the content in a file called `nginx.yaml` and run the command:

```
$ kubectl create -f nginx.yaml
deployment.apps/nginx-deployment created
```

The output looks successful. To see whether pods were created, as hoped, check the *default* namespace (Listing 6). As requested in the YAML file, two pods are running for extra resilience.

If you want to expose a container's service to a port on your host machine (although I haven't experimented with this yet), the process would involve something like:

- Delete the cluster you have running with the command `kind delete cluster`.
- Create a configuration file like that in Listing 7.
- Create a new cluster with the command:

```
kind create cluster --config cluster-config.yaml
```

If you then run a tool like `lsof` or `netstat` to show open ports, the output would show that the host machine's port was opened:

```
docker-pr  535    root    3u  IPv4  14678    0t0  TCP *:80 (LISTEN)
```

If you get stuck setting that up, I'd suggest disabling iptables and then

Listing 6: Check for Pods

```
$ kubectl get pods -n default
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-deployment-585449566-99qk6	1/1	Running	0	44s
nginx-deployment-585449566-pbzg2	1/1	Running	0	44s

Listing 7: Host Port Configuration File

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  extraPortMappings:
    - containerPort: 80
      hostPort: 80
      listenAddress: "0.0.0.0"
```


restarting Docker. I look forward to experimenting with ingress controllers and host machine ports when I get a chance. By doing so, you can then connect to your nginx pods in a meaningful way to test applications being exposed outside of the Kubernetes cluster.

The End Is Nigh

To create a full-blown Kubernetes cluster with just one command – and rapidly, at that – is a sight to behold. That the standard `kubectl` commands work seamlessly is just a bonus. Remember to run the

```
kubectl cluster-info --context kind-kind
```

command after building your cluster. For proof of concept deployments, compatibility, testing, and indeed other development activities, Kind is an excellent place to start with Kubernetes. I trust you will enjoy employing the excellent kind. I intend to use it as much as possible when running quick tests. ■

Info

- [1] Kind: [\[https://kind.sigs.k8s.io\]](https://kind.sigs.k8s.io)
- [2] Quick start: [\[https://kind.sigs.k8s.io/docs/user/quick-start\]](https://kind.sigs.k8s.io/docs/user/quick-start)
- [3] Docker engine: [\[https://docs.docker.com/engine/install/ubuntu\]](https://docs.docker.com/engine/install/ubuntu)
- [4] kubectl setup: [\[https://kubernetes.io/docs/tasks/tools/install-kubectl-linux\]](https://kubernetes.io/docs/tasks/tools/install-kubectl-linux)

[5] kindest/node:

[\[https://hub.docker.com/r/kindest/node\]](https://hub.docker.com/r/kindest/node)

[6] Configuring a Kind cluster: [\[https://kind.sigs.k8s.io/docs/user/quick-start/#configuring-your-kind-cluster\]](https://kind.sigs.k8s.io/docs/user/quick-start/#configuring-your-kind-cluster)

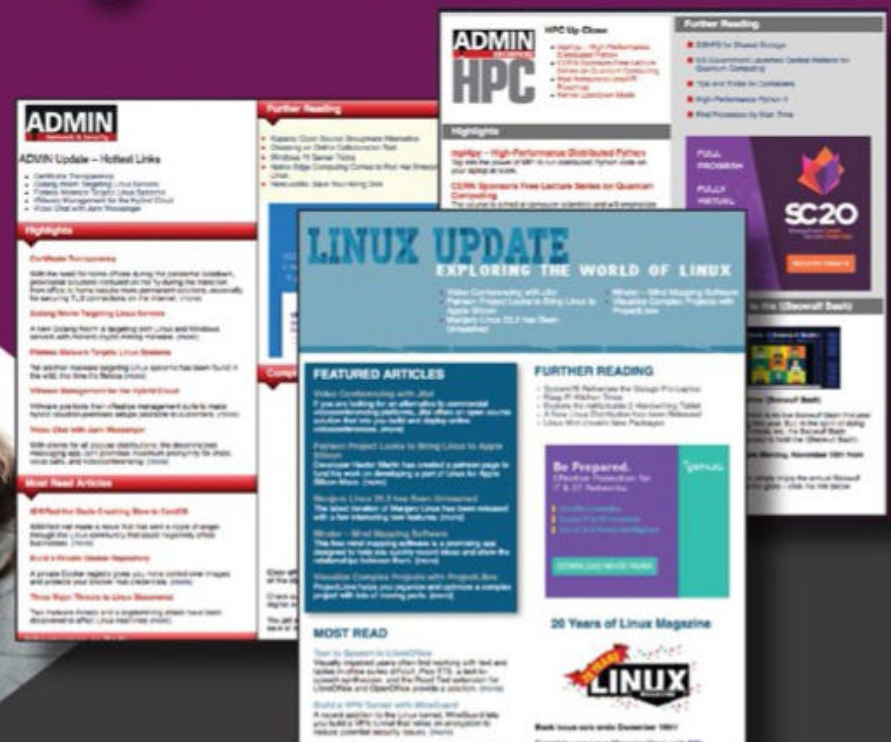
[7] Ingress controller: [\[https://kind.sigs.k8s.io/docs/user/ingress\]](https://kind.sigs.k8s.io/docs/user/ingress)

[8] Running Kind with rootless Docker: [\[https://kind.sigs.k8s.io/docs/user/rootless\]](https://kind.sigs.k8s.io/docs/user/rootless)

Author

Chris Binnie's new book, *Cloud Native Security*, teaches you how to minimize attack surfaces across all of the key components used in modern Cloud Native infrastructure. Learn with hands-on examples about container security, DevSecOps tooling, advanced Kubernetes security, and Cloud Security Posture Management: [\[https://www.cloudnativesecurity.cc\]](https://www.cloudnativesecurity.cc).

IT Highlights at a Glance



Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox.

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

ADMIN and HPC: bit.ly/HPC-ADMIN-Update

Linux Update: bit.ly/Linux-Update

Forensic analysis with Autopsy and Sleuth Kit

Game of Clue

Forensic admins can use the Autopsy digital forensics platform to perform an initial analysis of a failed system, looking for traces of a potential attack. By Matthias Wübbeling

Analyzing computer systems after a total failure (e.g., after an attack with malware) is the task of forensic specialists. With the appropriate tools, they can reconstruct log data, web history, or image data and detect so-called indicators of compromise. In this article, I introduce you to the Autopsy Sleuth Kit tool and show you how to use it for forensic analyses. After immediately provisioning alternative systems to secure business operations, one important task after a cyber incident in the enterprise is to process the incident and analyze the affected systems. In addition to countless commercial tools for the analysis and reconstruction of logs and data, you can also find very good, freely available, open source tools – such as the Sleuth Kit tool collection and its associated graphical user interface, Autopsy [1].

Images Only

Before you start analyzing the content of a hard drive, you first need to create a complete image of the disk. In

principle, you can also work with Autopsy directly on the running system or only analyze individual folders, but to be on the safe side and not destroy important data by accidentally writing to the drive, first connect to another system and create a corresponding image (e.g., with `dd` on Linux).

Whether you continue working with the hard disk or the image afterward depends a bit on the circumstances. Autopsy itself makes no distinctions and supports not only classic `dd` images, but also those in the expert witness format (EWF), a proprietary format belonging to EnCase software [2] from the software vendor OpenText, or virtual machine VMDK and VHD images.

If you are on Windows, you can download the latest version with all dependencies directly from the Sleuth Kit website and install it from the wizard. On Linux, use the package manager of the distribution you are using to install Sleuth Kit. After that, choose to download the Autopsy ZIP file from the website. After unpacking, run the `autopsy`

program in the `bin/` folder. If you receive the message that the Java JDK directory was not found, adjust the `etc/autopsy.conf` file. Remove the comment that assigns the directory to the `jdkhome` variable. After that, you will be able to launch Autopsy without any problems.

Source Analysis First

After starting Autopsy, use the *New Case* button to create a new case for processing. To begin, select a name and the working directory, and then enter your contact data. In the next step, select the first data source. If you want to work on a locally mounted hard drive on Windows, you will need to start Autopsy with administrative authorization.

Do so now, create a new processing case as before, then select *Local Disk* as the data source, and click *Next*. If you have selected an active hard drive, you have the option at this point to have Autopsy create an image of it in the form of a VHD file. After the first round, where it reads the

data, the tool then works with this image, and you can remove the hard disk to be examined from the system again and store it safely.

In the next step, from the available modules, select the file types you want to consider for the analysis and start the analysis run. This step will take some time, so get yourself some drinks and snacks and wait until Autopsy shows you the content of the hard disk or image. During the search, you can observe how data is found in the different categories in the menu on the left. For example, if you select *Web Cookies*, you will see a list of cookies found for the different installed browsers (**Figure 1**).

Targeted Knowledge Gain

If you press the *Timeline* link in the icon bar at top, you will see an overview of messages according to parameterizations. For older systems with many entries in the system log, creating a timeline of events takes a little time. In the upper area of the result window, you can change the display to show additional details about events besides the number of events. In the left pane, under *Filters*, you can reduce the number of events displayed. You can start the timeline while disk analysis is in progress. Autopsy then alerts you

when new data is found and offers to refresh the display.

Indicators of compromise (IoCs), for example, can be modeled and exchanged in structured threat information expression (STIX) format [3]. Analysts create STIX documents according to the malware they analyze and describe the IoCs. If you receive STIX documents from partners or service providers (e.g., from platforms such as the threat sharing platform MISP [4]), you can test them immediately with Autopsy by pressing the *Generate Report* button, selecting the STIX reporting module, and clicking *Choose file* to select the STIX document. You can also select a whole folder of STIX documents; Autopsy will then check all the files it contains and take them into account during reporting. After the analysis, you will see all found items under the *Interesting Items* menu.

Rescuing Files

Autopsy's retrospective analysis functionality also allows you to recover deleted files. Autopsy provides valuable services, especially in the field of image and video reconstruction. You can also reassemble image fragments – even if parts of the image have already been overwritten. To do this, the tools search the harddisk image block by block for potentially related

image data, so you can also recover images accidentally deleted from SD cards from digital cameras or from smartphones.

Additional modules let you expand functionality. For Android devices, the Android Analyzer Module supports the analysis of smartphone storage media. The module adds specific entries, such as call logs, contacts, or messages to the results menu. In this way, you can also back up conversations and contacts.

Conclusions

The Autopsy forensics tool lets you perform an initial analysis of the information and discover traces of a potential attack that are still present on the data medium after a system failure. However, if you don't have your own forensics department with the appropriate resources for such analyses, you should use external specialists for critical incidents.

Info

- [1] Autopsy: [<https://github.com/sleuthkit/autopsy>]
- [2] EnCase: [<https://security.opentext.com/encase-forensic>]
- [3] STIX: [<https://oasis-open.github.io/cti-documentation/stix/intro>]
- [4] MISP: [<https://www.misp-project.org>]

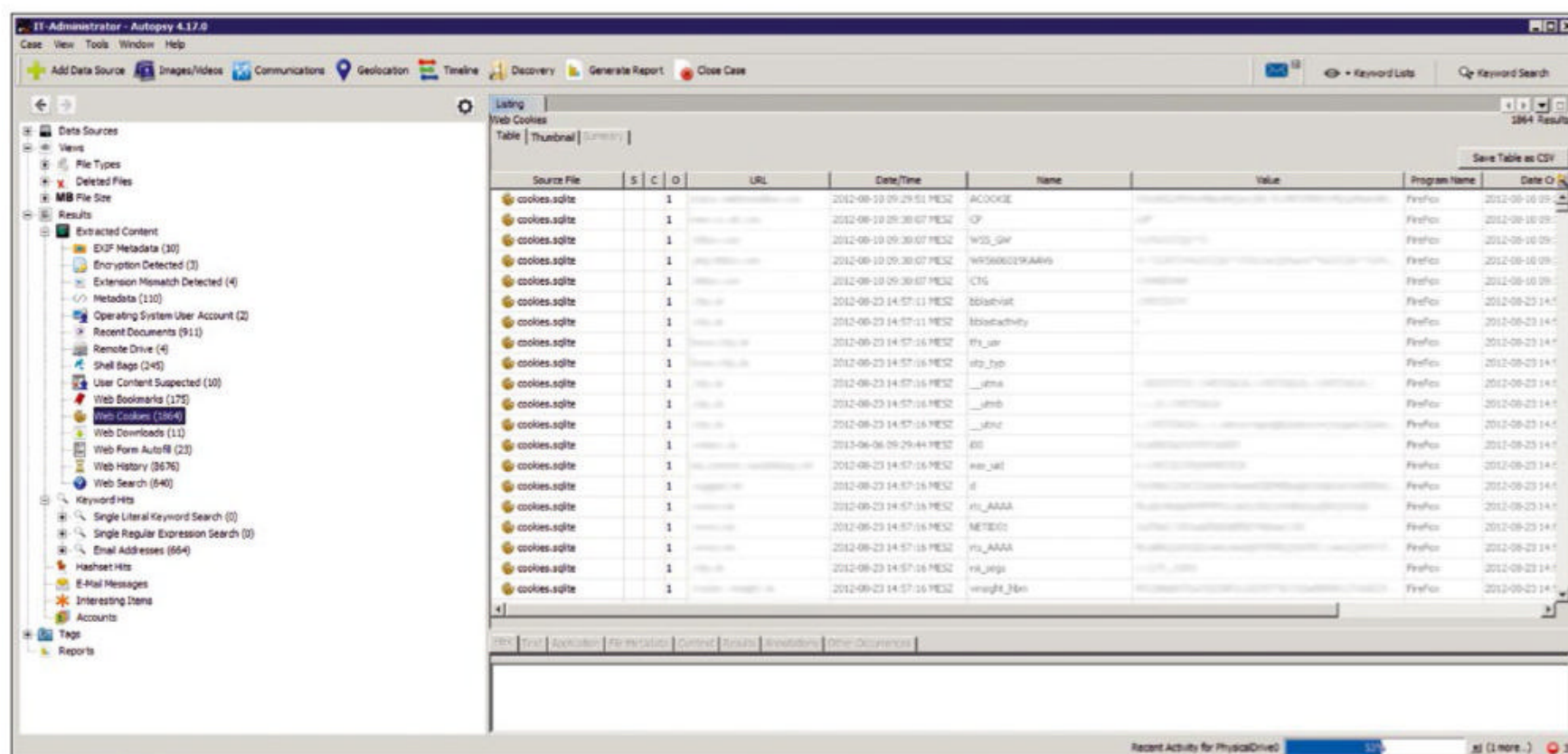


Figure 1: Helpful data snippets such as cookies can be tracked down with Autopsy.

Best practices for secure script programming

Small Wonders

The lax syntax verification of shell scripts and a lack of attention to detail in programming can create impressively dangerous security vulnerabilities. By Tam Hanna

Holistic system security means paying attention to even the smallest detail to avoid it becoming an attack vector. One example of these small, barely noticed, but still potentially very dangerous parts of IT that repeatedly cause serious security issues is shell scripts. Employing best practices can ensure secure script programming.

Letting Off Steam

Impressive proof of the potential harmfulness of shell scripts was provided by US software vendor Valve. The Linux-based version of the Steam game service included a script that was normally only responsible for minor setup tasks [1]. Unfortunately, it contained the following line:

```
rm -rf "$STEAMROOT/"*
```

This command, which is responsible for deleting the `$STEAMROOT` directory, gets into trouble if the environment variable is not set. Bash does not throw an error but simply “disassembles” the environment variable

into an empty string. The reward for this is the command

```
rm -rf /*.
```

which works its way recursively through the entire filesystem and destroys all information.

Some users escaped total ruin by running their Steam execution environment under an SELinux chroot jail. Others were not so lucky, so it is time to take a closer look at defensive programming measures for shell scripts.

Defining the Shell Variant

On Unix-style operating systems, dozens of shells are available – similar only in their support of the POSIX

standard – that come with various proprietary functions. If shell-specific code from one shell is used in other shells, the result is often undefined behavior, which might not be a problem in a controlled VM environment, but deployment in a Docker or other cluster changes the situation.

The most common problem is the use of the `#!/bin/sh` sequence, or “shebang,” which on Unix, is the first line of a script. According to the POSIX standard, the shebang affects the shell pre-selected by the system. It determines which interpreter is to be used for processing the script. The terms sha-bang, hash-bang, pound-bang, or hash-pling can also be found in the literature.

Bash, which is widely used among script programmers, is rarely the default shell. On a workstation based on Ubuntu 18.04, you can check the default by entering the `which` command. The `which sh` command returns the

```
tamhan@TAMHAN18: ~  
File Edit View Search Terminal Help  
tamhan@TAMHAN18:~$ checkbashisms itaworker.sh  
script itaworker.sh does not appear to have a #! interpreter line;  
you may get strange results  
possible bashism in itaworker.sh line 2 ('((' should be '$((':  
for ((i=0; i<3; i++)); do  
tamhan@TAMHAN18:~$
```

Figure 1: The `checkbashisms` command identifies Bash-specific code.

Photo by Jorge Fernández Salas on Unsplash

output `/bin/sh`, and the `which` bash command returns `/bin/bash`.

Several methods help you work around this problem. The simplest is to stipulate explicitly the use of Bash with `#!/bin/bash` in the shebang. Especially in cloud environments, it is not a good idea to assume the presence of the Bash shell. If you build your script without specific instructions, you can save yourself some extra work during deployment. The `checkbashisms` command helps administrators find and remove Bash-specific program elements. To install the tool, you need to load the *devscripts* package then create a small shell script (here, `itaworker.sh`) that contains a Bash-specific `for` loop:

```
#!/bin/sh
for ((i=0; i<3; i++)); do echo "$i"
done
```

If you parse this script with `checkbashisms`, it complains about the loop (Figure 1). Note that the tool only warns you of Bash-specific code if the shebang does not reference Bash. The following version of the script,

```
#!/bin/bash
for ((i=0; i<3; i++)); do echo "$i"
done
```

passes the `checkbashisms` check without problems.

Saving Passwords Securely

Administrators like to use shell scripts to automate system tasks, such as copying files or updating information located on servers, which requires a password or username or both that should not be shared with the general public. On the other hand, you do need to provide the credentials or enter them manually every time you run the script. Attackers like to capture all the shell scripts they can get on hijacked

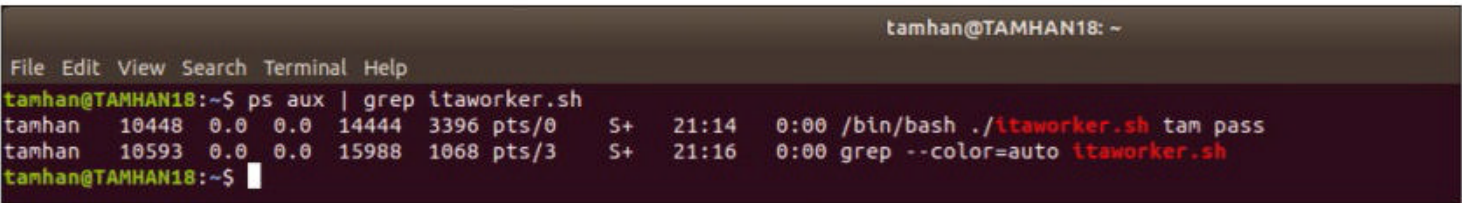


Figure 2: The use of parameters with credentials is not conducive to security.

systems. If they find a group of passwords in the process, the damage done is multiplied. The first way to fulfill this condition is to pass the credentials to the script as parameters. The following procedure would be suitable:

```
#!/bin/bash
a=$1
b=$2
while [ TRUE ]; do sleep 1;
done
```

The variables `$1` and `$2` stand for the first and second parameters. This script is run by typing `./itaworker.sh tam pass`, which is critical from a security point of view. The Linux command-line tool `ps` for listing running processes can output the parameters entered in the call if so desired. Figure 2 shows how the credentials can be delivered directly to the attacker’s doorstep; entering `ps` is one of an attacker’s first steps. If dynamic parameterization of the shell script is absolutely necessary, the recommended approach is to store the credentials in environment variables. These cannot be found by an ordinary `ps` scan, because access to `/proc/self/environ` requires

advanced user rights if the configuration is correct. However, this approach is not universally accepted – the Linux Documentation Project [2] recommends providing the information through a pipe or through redirection. Consensus across the board in literature is that it is definitely better if the password is not found in plain text in the script file. One way to achieve this is to use a reversible hash function that converts its output values, also known as the “salt,” back to the original value. If you use a program for salt conversion that the attacker does not catch in a scan, you can make life more difficult for them. If the attacker wants to have the credentials in plain text, they have to log in again and run the command. Ideally, the binary file will belong to an account that is exclusively used for shell script execution. Option number two (Figure 3) relies on a separate configuration file that has different read permissions. As long as the user responsible for executing the scripts is the only one who is allowed to read this file, an attack on the other user accounts is not critical in terms of credentials.

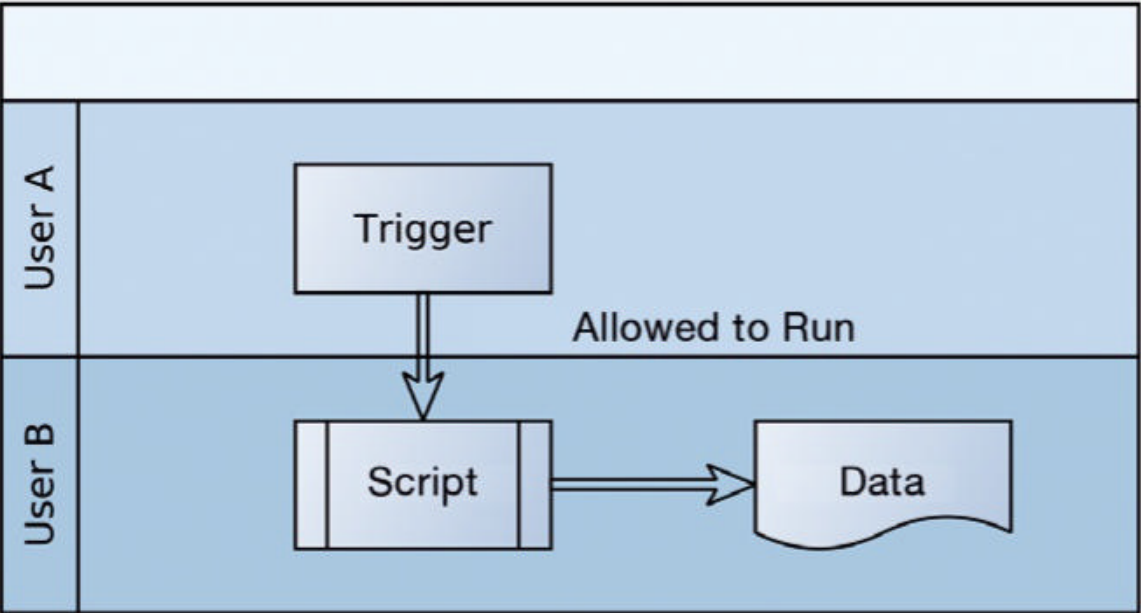


Figure 3: The Unix access control system protects the password configuration file against unauthorized access.

To hide the structure of the script, you can use `shc` [3], developed by Francisco Rosales. The package can be installed by the package manager on Ubuntu. The call expects the `-f` parameter to specify the source file:

```
shc -f itaworker.sh
ls itaworker.sh
```

After processing, you will find two additional files: In addition to the executable file with the extension `.X` is the file `.X.C`, which provides the C code of the script. SHC wraps the shell script in a C wrapper, which is then converted into a binary file by the C compiler. If you delete the `itaworker.sh` and `itaworker.sh.x.c` files, it is more difficult for the attacker to capture the code of the script later on. It should be noted that the wide distribution of SHC has led to the existence of decompilation tools [4].

Vulnerable Temporary Files

Assume a script needs to make the credentials available to other applications. Temporary files are often used for this purpose. If you create the file in the `/tmp/` folder, it should disappear after the script is processed. A classic example of this is the following script, which creates a file with a password:

```
#!/bin/sh
SECRETDATA="ITA says hello"
echo > /tmp/mydata
chmod og-rwx /tmp/mydata
echo "$SECRETDATA" > /tmp/mydata
```

Problems arise if an attacker monitors the temp file with the very fast acting Filesystem Watcher API. In the case of the present script, the attacker could call `fopen` and tag the file as open to harvest the information deposited there immediately or later. As before with plaintext passwords, the safest way is not to write the credentials to a temporary file. However, if this is essential to perform a function, the following approach is recommended:

```
#!/bin/sh
SECRETDATA="ITA says hello"
umask 0177
FILENAME=$(mktemp/tmp/mytempfile.XXXXXX)
echo "$SECRETDATA" > "$FILENAME"
```

First, the `umask` command assigns the access attributes passed in as parameters to newly created files. Second, the `mktemp` command creates a random file name each time, which makes configuring the Filesystem Watcher API more difficult. Numerous sources are critical of the use of temporary files in shell scripts in general [5].

Securing Against Incorrect Parsing

The most frequent point of criticism levied against the shell execution environment is that it lacks methods for string processing. In combination with passing in variables directly to the shell, serious vulnerabilities arise. A classic example is a script that reads in a parameter from STDIN and executes it with an `eval` command:

```
#!/bin/bash
read BAR
eval $BAR
```

The `eval` command is normally intended to perform calculations and harmless commands. One legitimate use case would be to deliver an `echo` command (with a pipe redirect if needed):

```
./itaworker.sh
echo hello
```

A malicious user would not pass an `echo` call with redirection at this point but a command such as `rm`, as mentioned earlier. As a matter of principle, the shell does not recognize this, which is why the script destroys data. Because some `eval`-based tasks are not feasible without this command, shell-checking tools like `shellcheck` do not complain about the use of the command. The script

```
#!/bin/bash
read BAR
eval "$BAR"
```

would trigger the error message *Double quote to prevent globbing and word splitting*. However, because this change does not verify the values contained in `BAR`, the vulnerability still exists. From a security perspective, the only correct way to handle the `eval` command is not to use it at all. If this is not an option, be sure to check the commands entered for strings like `rm` and for non-alphanumeric characters. Queries are another way to provoke problems. In the next example, I'll look at a script that checks to see whether the user enters the string `foo`. The payload, which here consists only of a call to `echo`, should only be executed if `foo` is entered:

```
#!/bin/bash
read F00
if [ x$F00 = xfoo ] ; then echo $F00
fi
```

In a superficial check of the script behavior, the input (e.g., *itahello*) would not lead to the execution of the payload. However, if you pass a string following the pattern `foo = xfoo` <any>, the payload is processed:

```
./itaworker.sh
foo = xfoo -o sdjs
foo = xfoo -o sdjs
~$
```

Damage occurs if something more sensitive, such as an `eval` call, needs to be secured instead of the `echo` payload. In the worst case, the payload then contains a command that causes damage because of invalid values. To work around this problem, you can use quoting, as in other shell application scenarios:

```
if [ "$F00" = "foo" ] ; then echo $F00
fi
```

The “new” version of the script evaluates the content of the variable completely, which results in the rejection of invalid input:


```
./itaworker.sh
foo
foo
./itaworker.sh
foo = xfoo
```

Quoting is even more important when working with older shells. A program built inline with the following pattern could be prompted to execute arbitrary code by entering a semicolon-separated string:

```
#!/bin/sh
read VAL
echo $VAL
```

The cause of this problem is that older shells do not parse before substituting the variable: The semicolon would split the command and cause the section that followed to be executed.

Ruling Out Parsing Errors

The rather archaic language syntax sometimes makes for strange behavior, as in the following example, which is supposed to call the file `/bin/tool_VAR`:

```
ENV_VAR="wrongval"
ENV="tool"
echo /bin/${ENV}_VAR
```

If you run this at the command line, you will see that it uses the value stored in the variable `ENV_VAR`. The shell separates `echo /bin/${ENV}_VAR` at the underscore. To work around this problem, it is advisable to set up all access to variables in curly brackets as a matter of principle. A corrected version of the script would be:

```
ENV_VAR="wrongval"
ENV="tool"
echo /bin/${ENV}_VAR
```

Running the corrected script shows that the string output is now correct:

```
./itaworker.sh
/bin/tool_VAR
```

Problem number two concerns the analysis of the shell user permissions.

The `$UID` and `$USER` variables can be used to analyze the permissions of the currently active user:

```
if [ $UID = 100 -a ?
    $USER = "myusername" ] ; then
    cd $HOME
fi
```

Unfortunately, the two variables are not guaranteed to contain the value corresponding to the active user at runtime. Although many shells now save the variable `$UID`, this is usually not the case for `$USER`: In older shells it is even possible to provide both variables with arbitrary values. To circumvent this problem, it is a good idea to determine the username and user ID by calling operating system commands. The values returned in this way can only be influenced by setting the path variable, which complicates the attack.

In particularly critical execution environments, it is always advisable to write out paths in full. For example, if you call `echo` by its path `/bin/echo`, the activated program is independent of the path variable.

Finally, I should point out that the Bash shell supports a special version of the shebang: `#!/bin/bash -p`. This variant instructs the shell not to execute the `.bashrc` and `.profile` files as part of the startup. The `SHELLOPTS` variable and the start scripts created in `ENV` and `BASH_ENV` are also cast aside in this operating mode. The purpose of this procedure is that manipulations of the environment carried out by the attacker cannot be activated by random script executions.

Stricter Execution Verification

Shell scripts are intended for non-programmers to speed up common tasks in their daily lives on Unix operating systems. As a courtesy to this group of users, virtually all shells are set to be “permissive”: In case of an error, they try to continue executing the script. This approach, which is quite friendly for a less technically experienced user, is critical because ambiguities that cause security problems do not stop the execution of the script. The previously mentioned problem of destroying files would not have occurred with the following shell setting:

```
#!/bin/bash
set -o nounset
TAMS_VAR="wrongval"
echo /bin/${ENV}_VAR
```

When called with the `-o` parameter, the `set` command activates execution options that influence the runtime behavior of the interpreter. The `nounset` option used here considers unset variables to be errors. Execution of the shell script shown before now fails with the error shown in [Figure 4](#). The `set -o errexit` command terminates a program if a command called by the shell script does not return a value of zero. Its use can be checked by copying a non-existent file:

```
#!/bin/bash
set -o errexit
cp "doesnotexist" "tree.txt"
echo "Successfully copied"
```

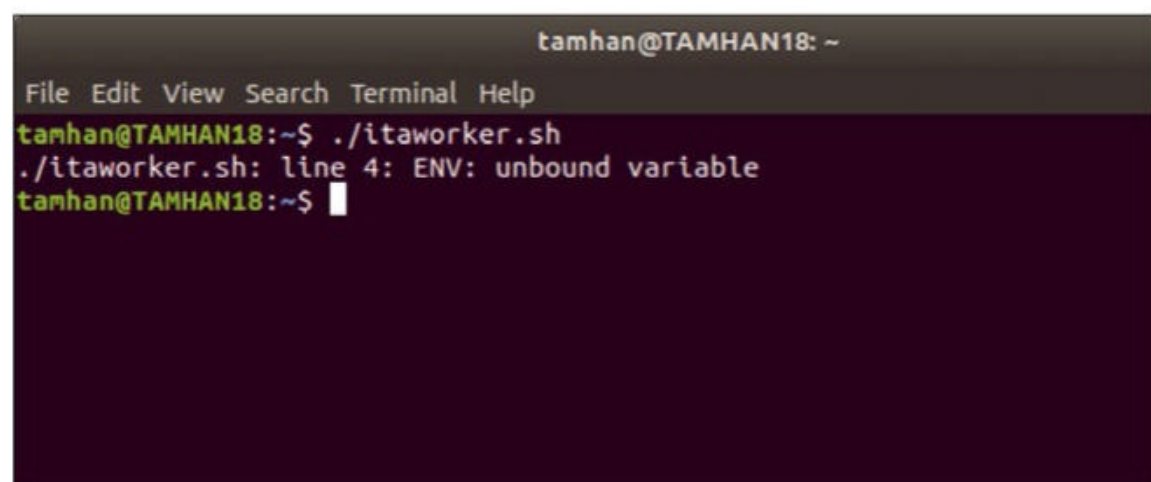


Figure 4: The `nounset` option terminates program execution if it encounters missing variables.

The `cp` command does not return zero if it fails to find a parameter, which is why the status message passed to `echo` does not appear at the command line.

In scripts, you will always run into situations in which some of the code to be executed is non-critical, and the commands specified by `-o` can be disabled by calling `+o`. In the following script, the `errexit` option is no longer active when the copy command is processed:

```
#!/bin/bash
set -o errexit
set +o errexit
cp "doesnotexist" "tree.txt"
echo "Successfully copied"
```

A problem can occur when processing commands connected by a pipe. Normally, this kind of command only fails if the last command in the pipe sequence does not return a value of zero. If you want errors somewhere in the hierarchy to cause a termination, then enable the `pipefail` option with the command `set -o pipefail`.

Changes to Global Variables

Another unintended result of lax language syntax is that variables defined in subfunctions develop unexpected side effects. As an example, look at a script that uses the variable `o` both inside and outside a function:

```
#!/bin/bash
hello_world () {
    o=2
    echo 'hello, world'
}
o=1
echo $o
```

```
hello_world
echo $o
```

Developers who grew up with other scripting languages would expect both calls to `echo $o` to return the value `1`. This is not inherently the case; the changes made in the function remain valid even after they have been executed. To fix the problem, you just need to tag the variables in the function with the `local` keyword:

```
#!/bin/bash
hello_world () {
    local o=2
    echo 'hello, world'
}
o=1
echo $o
hello_world
echo $o
```

Although longer shell scripts are not necessarily recommended for maintenance reasons, you should always protect variables used in functions with the `local` keyword. A colleague that recycles shell code at some point will be better protected from unexpected side effects.

ShellCheck

C and C++ programmers have used static analysis to check the results of their work for a long time. A static analysis tool has a knowledge base in which it stores programming errors that it checks against the code at hand. A similar tool for shell scripts is available with ShellCheck [6]. The tool, licensed under the GPLv3, behaves much like `lint` in terms of syntax: It expects the name of the shell file as a parameter, which it then

checks for dodgy elements.

ShellCheck doesn't only look for security-relevant errors. In a list of ShellCheck checks available online [7], four dozen test criteria also identify and complain about general programming errors. Last but not least, a web version of the tool [8] checks scripts for correctness and security without a local installation.

Conclusions

The close integration between scripts and the shell designed to execute commands, accompanied by lax syntax verification, guarantees that a lack of attention in shell programming is quickly punished by dangerous security vulnerabilities.

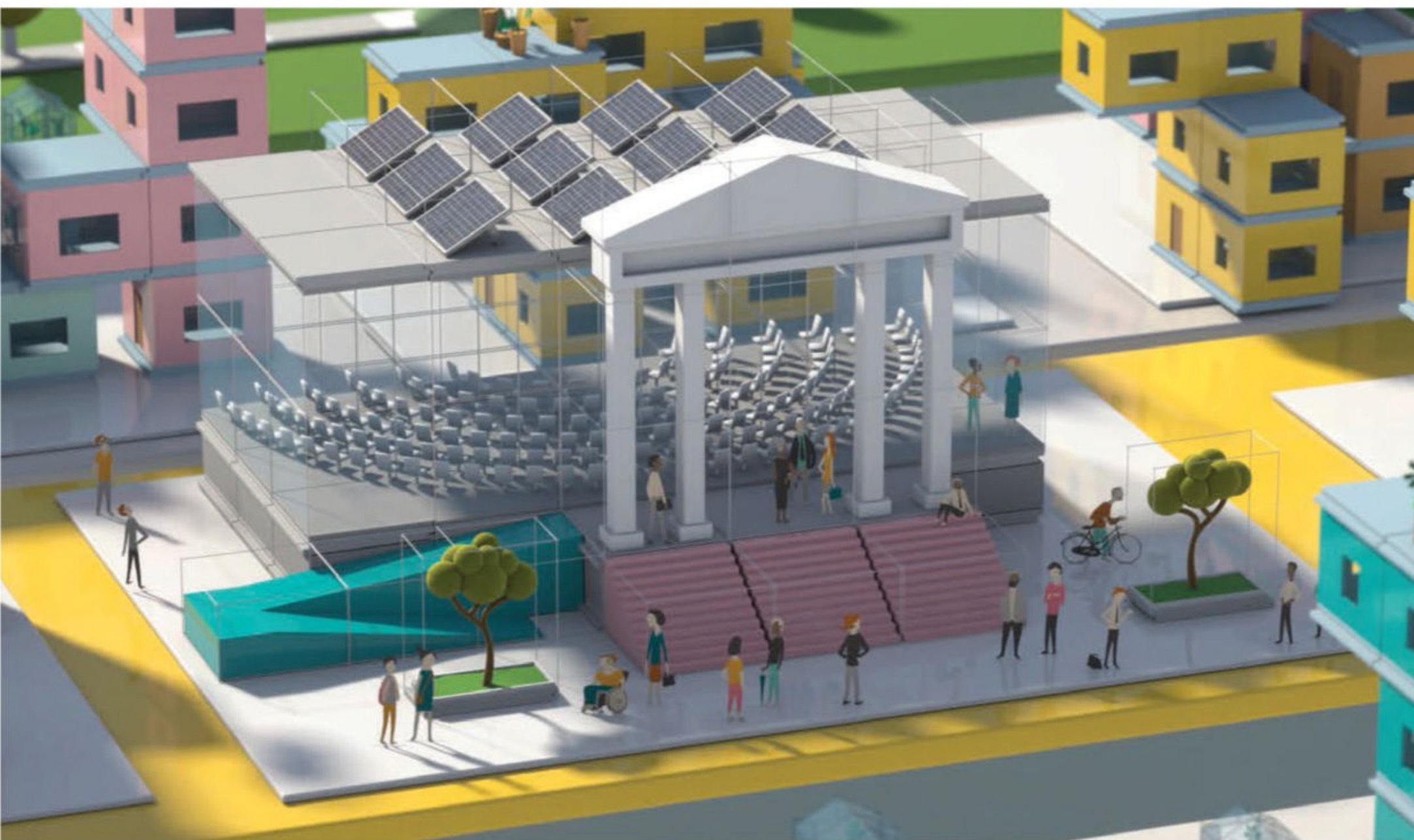
The criteria presented here can help you work around most of the problems. Unlike complicated security vulnerabilities, in the case of shell scripts, the cause is usually a simple lack of understanding that a certain construct can have side effects. ■

Info

- [1] Problems with Steam script: [\[https://github.com/valvesoftware/steam-for-linux/issues/3671\]](https://github.com/valvesoftware/steam-for-linux/issues/3671)
- [2] Linux Documentation Project: [\[https://tldp.org\]](https://tldp.org)
- [3] SHC man page: [\[http://www.datsi.fi.upm.es/~frosal/sources/shc.html\]](http://www.datsi.fi.upm.es/~frosal/sources/shc.html)
- [4] SHC limits: [\[https://www.linuxjournal.com/article/8256\]](https://www.linuxjournal.com/article/8256)
- [5] Secure use of temp files: [\[https://www.netmeister.org/blog/mktemp.html\]](https://www.netmeister.org/blog/mktemp.html)
- [6] ShellCheck: [\[https://github.com/koalaman/shellcheck/\]](https://github.com/koalaman/shellcheck/)
- [7] List of ShellCheck checks: [\[https://github.com/koalaman/shellcheck/wiki/Checks\]](https://github.com/koalaman/shellcheck/wiki/Checks)
- [8] ShellCheck on the web: [\[https://www.shellcheck.net\]](https://www.shellcheck.net)

Public Money

Public Code



Modernising Public Infrastructure with Free Software



Free Software Foundation Europe

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



Manage Windows AD with PowerShell

Organized

PowerShell helpers let you automate searches in Active Directory and secure critical accounts.

By Florian Frommherz

On many networks, Active Directory (AD) is the must-have setup for authentication and assignment of rights and as a directory service. With such a central service, everything should run smoothly with PowerShell automation. In this article, I show you how to search in AD, how to secure critical accounts, and which PowerShell helpers you will want to use. Administrators have gained a lot of experience in maintaining and operating Active Directory over its 20-year history. The tools and how they programmatically and automatically trigger changes in the directory have also changed, both in terms of data administration (i.e., managing users, computers, service accounts, and all the other objects in the directory) and

in terms of the scripts for controlling the directory service itself (i.e., the service that runs on Windows and provides the domain function). Tasks that used to be automated by VBScripts, plain vanilla LDAP, Win32 calls, and, later, .NET are now a little easier for admins and abstracted by PowerShell.

PowerShell Helpers

Even newcomers or occasional scripters should have a few decent tools for creating scripts or one-off commands in their toolbox. On the one hand, the commands can be assembled with autocompletion, after which parameters can be suggested and easily inserted; on the other

hand, tools allow the one-liners or scripts to be executed directly with color coding, thus making copy and paste into a separate PowerShell session unnecessary. The tools also allow individual lines from longer scripts to be executed separately for step-by-step testing. Of course, it is also possible to open a separate PowerShell session and enter and process the commands directly, but why make things more difficult than necessary?

Windows comes with the PowerShell Integrated Scripting Environment (ISE) as an add-on: It is immediately ready for use in PowerShell but is no longer actively developed by Microsoft. You can still create your Windows AD commands with it, specifically because the tool is on board and available on domain controllers with the same feature set.

One alternative is Visual Studio Code (**Figure 1**), which is downloadable free of charge for all current Windows versions, and it offers PowerShell language support for Visual Studio Code as an extension. The extension then comes with intelligent suggestions for parameters and command highlighting for improved visual processing of tasks.

Photo by Kelly Sikkema on Unsplash

Preparing for PowerShell Access to AD

Microsoft provides some ready-made PowerShell commands for AD that, once installed, support easy interaction. These cmdlets then interact with the corresponding services that work on domain controllers and use the APIs that Microsoft provides as part of AD. You don't have to worry about the actual API or functions, as long as the PowerShell wrappers are all you need. These PowerShell commands became part of the OS in Windows 10 version 1809 and are activated manually as a feature; older Windows 10 versions require the Remote Server Administration Tools, which also includes the PowerShell module [1].

On domain controllers, when you promote the server you will be prompted as to whether you want to install the administration tools and PowerShell together with the domain controller

role. If the module is not available, you can install it later with Server Manager, which lets you enable the Windows feature (*Role Administration Tools | AD DS and AD LDS Tools | Active Directory module for Windows PowerShell*). In PowerShell you can enter:

```
Import-Module ServerManager
Add-WindowsFeature 2
-Name "RSAT-AD-PowerShell" 2
-IncludeAllSubFeature
```

Once complete, you can display an overview of all the available cmdlets that you can use for Microsoft Active Directory:

```
Get-Command -Module ActiveDirectory
```

You will quickly see that the commands all have the familiar PowerShell verb at the beginning and then the command with the AD* prefix. You will also recognize some known

objects among the cmdlets – ADUser, ADGroup, ADComputer, ADGroupMember, ADAccount, and many more.

Searching in Active Directory

Users in AD, which you can query with Get-ADUser, are also of interest. Either the sAMAccountName, the DistinguishedName, the ObjectGUID, or the SID are used as the search keys:

```
Get-ADUser -Identity flofromm
```

If you are looking for all employees of a certain department, the filter helps narrow things down on the attribute level:

```
Get-ADUser -Filter "Department -like 'IT*'"
```

The filter works with all common attributes. If all relevant users are already grouped into organizational

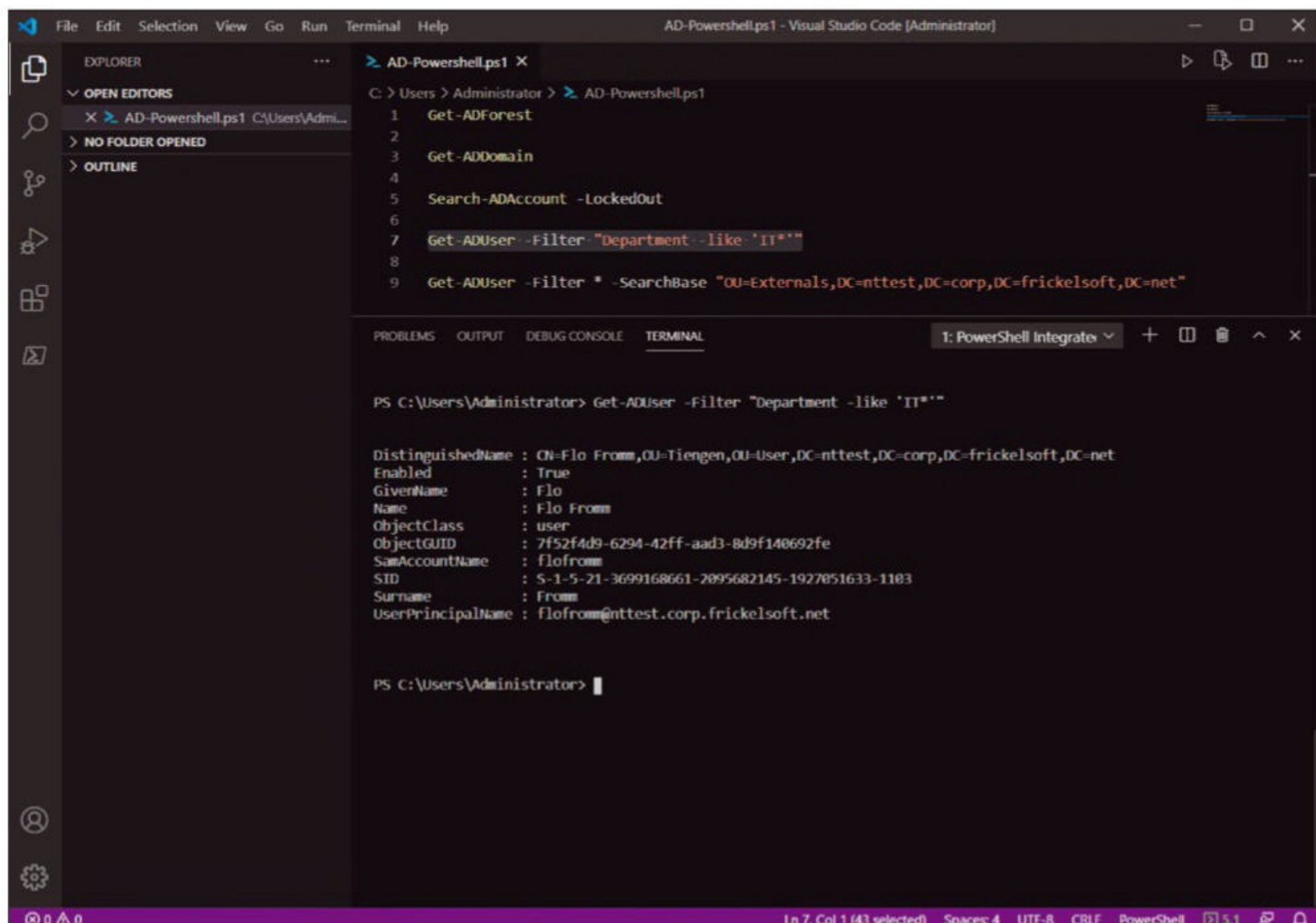


Figure 1: Tools such as Visual Studio Code are worthwhile to avoid trouble with automation, searches, and changes, even if you only occasionally work with scripts.

units, you can find them by specifying the directory path as `SearchBase`. The LDAP notation is used here; the `Externals` organizational unit (OU) below the `corp.frickelsoft.net` domain, would be written as:

```
Get-ADUser -Filter *
-SearchBase "OU=Externals,DC=corp,DC=frickelsoft,DC=net"
```

Of course, you can also combine `SearchBase` with a filter. The `Search-ADAccount` cmdlet is also useful if you are looking for AD accounts but do not want to search by user or computer. The following commands find all locked-out accounts and inactive accounts belonging to both users and computers:

```
Search-ADAccount -LockedOut
Search-ADAccount -AccountInactive
-TimeSpan 120.00:00:00
ft Name,LastLogonDate,Enabled
```

To inspect groups, your best option is the `Get-ADGroup` cmdlet:

```
Get-ADGroup -Filter * -Properties member
```

The cmdlet gives you a good overview of the properties of a group. Besides `SearchBase`, groups can also be narrowed down by `Filter` (e.g., if you are only looking for security groups):

```
Get-ADGroup
-Filter "GroupCategory
-eq 'Security'"
-SearchBase "OU=Groups,DC=corp,DC=frickelsoft,DC=net"
```

If you explicitly query the group members as an attribute with the `Get-ADGroup` cmdlet, you are only given text output in return. For

further use of the group members as PowerShell objects, try the `Get-ADGroupMember` cmdlet, which only returns the group members:

```
Get-ADGroupMember
-Identity 'Enterprise Admins'
-Recursive
Get-ADGroupMember
-Identity 'Domain Admins'
-Recursive
```

The `Recursive` option also resolves nested group memberships. If you want to reuse the member list in another command with a pipe, the cmdlet of choice is `Get-ADGroupMember`:

```
Get-ADGroupMember
-Identity 'Domain Admins' -Recursive |
Get-ADUser
-Properties Emailaddress,
lastLogonDate |
Export-CSV -Path "C:\temp\csv\Domain
Admins.CSV"
```

However, all groups can be queried with the `Get-ADGroup` cmdlet,

```
Get-ADGroup -Filter "Name -like 'HR*'"
-SearchBase 'OU=Groups, DC=nttest,DC=corp,DC=frickelsoft,DC=net'
-SearchScope SubTree | Get-ADGroupMember
Get-ADGroup -Filter "Name -like 'HR*'"
-SearchBase 'OU=Groups,DC=nttest,DC=corp,DC=frickelsoft,DC=net'
-SearchScope SubTree |
Export-CSV -Path 'C:\temp\csv\HR_departmental_groups.csv'
```

and exported (e.g., to a CSV file as in the second command), if so desired.

Making Changes

You can modify users of a certain OU with a one-liner so that they all have a certain attribute value, allowing

other programs (e.g., `AADConnect` for synchronization to the cloud) to find and process them:

```
Get-ADUser -Filter *
-SearchBase "OU=Externals,DC=corp,DC=frickelsoft,DC=net" |
Set-ADUser
-Add @{extensionAttribute4 = "M365"}
```

An imported CSV file lets you edit multiple users:

```
Import-CSV 'C:\temp\csv\users.csv' |
% { if($_.mail
-like '*@frickelsoft.net') {
Set-ADUser
$.sAMAccountName
-Add @{extensionAttribute4 =
"M365"}}}
```

This command imports a CSV file that contains user definitions and parses it line by line. Two columns in the CSV are inspected more closely: `mail` and `sAMAccountName` (Figure 2). If the mail address ends with `@frickelsoft.net`, `extensionAttribute4` of the user in AD is set to `M365`. Nothing happens for the other users found in the CSV file.

To transfer the value of the `region` column to an AD attribute (e.g., `preferredDataLocation` for Office 365), you need to modify the command

```
Import-CSV '.\Downloads\users.csv' |
% { Set-ADUser
-Identity $.sAMAccountName
-Add @{preferredDataLocation =
$.region }}
```

to take the cell value from the CSV file instead of using a fixed value.

Managing Special Accounts

Of course, you will not only want to manage normal user accounts, but also accounts belonging to external partners and suppliers or service accounts. You can simply disable accounts that are no longer needed:

```
Get-ADUser -Identity svc_low_SQL3 |
Disable-ADAccount
```

	A	B	C	D
1	sAMAccountName	mail	title	hireDate
2	franny	franny@frickelsoft.net	TECHNICAL ADVISOR	01-12-99
3	sarah	sarah@frickelmail.net	HR ADVISOR	02-07-10
4	jeff	jeff@frickelsoft.net	SALES ASSISTANT	01-10-17
5	kath	kath@frickelsoft.net	REGIONAL DIRECTOR - EMEA	01-03-03
6				

Figure 2: PowerShell parsing users from an Excel or CSV file and editing their accounts in AD.

This method also works with accounts belonging to external identities that are grouped in an OU. To block people or service accounts that have not logged in for 120 days or more, use:

```
$lastLogonCutoff = 2
(Get-Date).AddDays(-120)
Get-ADUser 2
-Filter { LastLogonDate 2
    -lt $lastLogonCutoff 2
    -and Enabled -eq $true } 2
-SearchBase "OU=Externals,DC=corp,2
    DC=frickelsoft,DC=net" | 2
Disable-ADAccount
```

In the best case, the service accounts that you create as normal user accounts and not as group-managed service accounts (gMSA) are restricted to the extent that the logon can only take place on certain machines to avoid misuse. To define this case, use the LogonWorkstations parameter:

```
Set-ADUser svc_low_SQL3 2
-LogonWorkstations "SQL3"
```

This attribute expects a comma-separated list of machine names. For example, you can create a new service account that you want to log on to all SQL servers with:

```
$sqlServers = Get-ADComputer 2
-Filter "Name -like 'SQL*'" 2
-SearchBase "OU=Servers,OU=Tier1,2
    DC=corp,DC=frickelsoft,DC=net" | 2
SELECT sAMAccountName | 2
%{$_.sAMAccountName.Trim("$")}
$sqlServers = $sqlServers -join ","
Set-ADUser svc_high_SQL3 2
-LogonWorkstations $sqlServers
```

If you only want the service account to be valid for a time-limited project, or if the owner of the account is required to come back and renew the account within 90 days at the latest, you can set an expiration date:

```
Set-ADAccountExpiration 2
-Identity svc_low_SQL3 2
-TimeSpan 90.00:00:00
```

You will certainly want to minimize the number of accounts whose passwords never expire. In fact, apart from service accounts, there should be no accounts in AD whose passwords do not expire:

```
Search-ADAccount -PasswordNeverExpires | 2
Export-CSV C:\temp\csv\neverexpires.csv
```

Alternatively, you can move these accounts to a group for delegated administration:

```
Search-ADAccount -PasswordNeverExpires | 2
%{Add-ADGroupMember 2
-Identity "PWDNeverExpires" 2
-Members $_.sAMAccountName }
```

If you specifically need to harden certain accounts against hijacking and ensure that the passwords used are appropriately complex, you can attach custom password policies to these accounts. For example, with a few lines of PowerShell, you can create a new password settings object that contains custom password policies and assign it to a group of service accounts. First, create the password policy object that enforces manual unlocking and long, complex passwords ([Listing 1](#)).

After that, define a new AD group to which you then add the service accounts as members before assigning the policy ([Listing 2](#)).

Finally, search and find the service accounts to be protected in AD:

```
Get-ADUser -Filter 'DisplayName 2
-likes "SVC_HIGH_*"' 2
-SearchBase "OU=Service Accounts, 2
    DC= corp,DC=frickelsoft,DC=net" | 2
% { Add-ADGroupMember "High Sec 2
    Service Accounts" -Members $_ }
```

The next time the password is changed, the service accounts – or the admin who resets and changes the passwords – has to comply with the new password policy.

Managing Forests and AD Services

PowerShell lets you inspect the AD service itself, as well as manage the data in the directory. Cmdlets let you create new domain controllers, domains, AD objects, and partitions and inspect existing objects:

```
$forest = Get-ADForest 2
-Server "corp.frickelsoft.net"
foreach($domain in $forest.Domains) { 2
    Get-ADDomainController -Filter * 2
    -Server $Domain }
```

The following two cmdlets show you the flexible single master operation (FSMO) role holders:

```
Get-ADForest | 2
SELECT DomainNamingMaster, SchemaMaster
Get-ADDomain 2
-Name corp.frickelsoft.net | 2
```

Listing 1: Enforcing Complex Passwords

```
New-ADFineGrainedPasswordPolicy -Name "HighSecServiceAccountsPolicy" -Precedence 500 -ComplexityEnabled $true -Description "Password Policy for High
    Sec Service Accts" -DisplayName "High Sec Service Accts PassPolicy" -LockoutDuration "00:00:00" -LockoutObservationWindow "00:00:00" -LockoutThreshold
    7 -MinPasswordAge "01:00:00" -PasswordHistoryCount 20 -MinPasswordLength 16
```

Listing 2: Defining New AD Group

```
New-ADGroup -Name "High Sec Service Accts" -SamAccountName HighSecServiceAccts -GroupCategory Security -GroupScope Global -DisplayName "High Sec
    Service Accts" -Path "OU=Groups,OU=Service Accounts, DC=corp,DC=frickelsoft,DC=net" -Description "Members of this group are High Security Service
    Accounts"
Add-ADFineGrainedPasswordPolicySubject -Identity HighSecServiceAccountsPolicy -Subjects 'HighSecServiceAccts'
```



```
SELECT InfrastructureMaster, 2
PDCemulator, RIDMaster
```

The `Get-ADDomain` and `Get-ADForest` cmdlets have additional properties that you can use for an inventory or check: The `DomainFunctionalLevel` can be found in `DomainMode` for each domain, and the `ForestFunctionalLevel` is found in `ForestMode` for forest objects. If you want to provision devices that have been added to the domain to an Azure AD, you need to configure a hybrid Azure AD join. One step in the configuration is to create the service connection object manually or automatically in the configuration partition of the AD. To check whether the object has been created, use:

```
configPartition = (Get-ADRootDSE).2
configurationNamingContext
Get-ADObject -Filter * 2
-SearchBase "LDAP://CN=62a0ff2e-97b9-2
4513-943f-0d221bd30080,2
CN=Device Registration Configuration,2
CN=Services,$($configPartition)"
```

You can check the AD schema version with PowerShell with:

```
Get-ADObject 2
(Get-ADRootDSE).schemaNamingContext 2
-Property objectVersion
```

Output of version 88 means Windows Server 2019, 87 means Windows

Server 2016, and 69 means Windows Server 2012 R2.

If you use Exchange, you can find the schema version for Exchange with:

```
Get-ADObject 2
-Identity "CN=2
ms-Exch-Schema-Version-Pt,$(2
(Get-ADRootDSE).schema2
NamingContext)" 2
-Properties rangeUpper | 2
SELECT rangeUpper
```

Exchange Server 2019 has version numbers starting from 17000, whereas Exchange version 2016 is in the range of 15317 to 15333.

Evaluating Password Protection

If you use Azure AD and have premium licenses, you probably also use the AD Password Protection for Windows feature, which extends password checking on domain controllers to include logic and insights from Azure AD. The feature prohibits users from choosing common or easy-to-guess passwords when changing passwords or from choosing passwords that you list as undesirable in Azure AD [2]. When it runs, the agent required for this on the domain controller logs how many password changes were rejected because they were

too weak or are on your undesirables list:

```
Get-AzureADPasswordProtectionSummary2
Report -DomainController NTTEST-DC-01
DomainController: NTTEST-DC-01
PasswordChangesValidated: 4
PasswordSetsValidated: 2
PasswordChangesRejected: 7
PasswordSetsRejected: 5
...
```

Conclusions

This AD PowerShell exploration shows that you can automate common searches and tasks with very little overhead and create tiny scripts that you store in your favorite development environment to make your work easier. Often it doesn't take much work at all: If you structure your administration workstation with a good code editor for PowerShell, you can start automating Active Directory quite quickly and flexibly. ■

Info

- [1] Remote Server Administration Toolkit: [\[https://docs.microsoft.com/en-us/windows-server/remote/remote-server-administration-tools\]](https://docs.microsoft.com/en-us/windows-server/remote/remote-server-administration-tools)
- [2] Azure AD Password Protection: [\[https://docs.microsoft.com/en-us/azure/active-directory/authentication/howto-password-ban-bad-on-premises-deploy\]](https://docs.microsoft.com/en-us/azure/active-directory/authentication/howto-password-ban-bad-on-premises-deploy)



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

GET IT NOW!

FAST DELIVERY
WITH OUR PDF
EDITION



Self-signed certificates with Jenkins

Handmade

Convince Jenkins as a Docker container to recognize self-signed certificates, verify that the instance is connecting to the correct online service, and that your traffic is transmitted in an encrypted format. By Chris Binnie

Whether you are new to continuous integration/continuous deployment (CI/CD) pipelines and the world of DevOps or fully familiar with such practices, one name always comes to the fore when discussing the automation of processes: Jenkins [1]. The highly popular automation server is open source, and although capable of fulfilling many types of complex tasks, it is equally accessible to novice users through its straightforward user interface (UI). In this article, I look at how I solved a recent headache on a project in which I was running Jenkins as a Docker container for the demonstration of a vendor-supplied Jenkins plugin.

Although I haven't used Jenkins much in the past, the quandary I faced and, most importantly, resolved could affect novice and advanced users alike. The problem was that once I had figured out how to update Jenkins plugins online, I couldn't connect a plugin to remote software as a service (SaaS) because it was using self-signed security certificates that were not recognized by Jenkins.

The Version

According to my relatively cursory inspection of the provided Docker containers, the first thing to note about Jenkins is that the version

at two versions in particular: the very latest and greatest weekly release called `jenkins/jenkins:latest` and the `jenkins/jenkins:lts`, where *lts* stands for the long-term support stable version.

The reason the version seems to matter so much is that Jenkins is developed at great speed and is constantly evolving. Admirable and fantastic as that is, every now and again trying to solve an issue on a version that hasn't been written about much online is tricky at best. This situation applies to most modern software and definitely not just the pervasive-for-good-reason Jenkins.

Although Jenkins provides a carefully considered and slick UI, in this article I focus on the container side and a number of command-line interface (CLI) options. Again with reference to the importance of versioning, the issue I encountered when trying to figure out UI options was that several version changes ago a number of menu options were deprecated. Nonetheless, with some trial and error I settled with the LTS version of the con-

you use can be the difference between documentation immediately solving a problem or compounding it further. From the container perspective, I opted to look

Listing 1: `run_jenkins.sh`

```
docker run -d \
  --name jenkins-lts \
  --user root \
  -p 0.0.0.0:8080:8080 \
  -p 0.0.0.0:8443:8443 \
  -p 0.0.0.0:50000:50000 \
  --env JENKINS_OPTS="--httpPort=8080" \
  --env JAVA_OPTS="-Djavax.net.ssl.trustStore=/var/jenkins_home/keystore/cacerts" \
  -v jenkins_config:/var/jenkins_home \
  jenkins/jenkins:lts
```


tainer image and the plugin working as hoped. The results can be reproduced easily to accommodate other online services that Jenkins doesn't immediately work with out of the box because the certificate authority isn't recognized.

The Installation

I prefer to save complex Docker run commands in a small script, as shown in the `run_jenkins.sh` file shown in [Listing 1](#). (Remember to run

```
chmod +x run_jenkins.sh
```

to make it executable.)

Rather than just blindly running that script, which you can do without breaking anything in a laboratory environment, you might first want to read through to the end of this article to understand why `--user root` is present, along with the `JAVA_OPTS` environment variable.

Importantly, before creating a container with the script in [Listing 1](#), I will ensure that changes to configuration data persist between stops and starts by creating and checking a volume called `jenkins_config`:

```
$ docker volume create jenkins_config
$ docker volume ls | grep jenkins
local    jenkins_config
```

To run the script, proceed with creating a container:

```
$ ./run_jenkins.sh
```

If you run `$ docker ps` and no errors are present, you should see it running.

The Plugins Quandary

The more minor of my connectivity issues were that I couldn't update plugins to get the most secure versions. Jenkins uses an online repository to store and serve its incredible library of exceptionally useful plugins. It seems, however, that under some scenarios, even when a proxy isn't used to connect to the Internet, Jenkins struggles to connect over HTTPS.

After hunting online and trying a few different fixes without success, what worked for me was a remarkably simple URL edit ([Figure 1](#)) that let me continue updating and installing the plugins required for the project demo I was preparing. To get to that UI option, navigate to *Manage Jenkins* | *Manage Plugins*, select the *Advanced* tab, and scroll to the bottom of the page.

Simply alter `https` to `http` in the URL and then click the *Submit* button. You can now promptly navigate to the *Check Now* button, which is on the bottom right of that same page. Then scroll back to the top of the page and click the *Updates* tab to check that the fix worked. If for some reason that doesn't work, go into your browser settings and see if you are using a proxy to access the Internet. If you are, at the top of the same page as shown in [Figure 1](#), you can enter your proxy information manually (e.g., `web.proxy.org:8080`), along with login details if they are used. If it works, update all your plugins as required.

If you are keen to use HTTPS for your plugin connectivity (which is obviously recommended for the benefit of security outside of a lab environment), you can potentially use the next solution, or (armed with the knowledge that JUC stands for Jenkins Update Center) you could create your own plugin repository locally, according to instructions online [\[2\]](#). This step could also be useful if you want to limit which plugins developers have access to for security reasons.

The Self-Signed Certificate Palaver

After investigating various online fixes, I was eventually able to connect the proprietary Jenkins plugin I was using to an online service even though a certificate authority didn't recognize

its self-signed certificate. The solution came about through distilling some of the instructions found on Stack Overflow [\[3\]](#). The salient bits are as follows:

- Enter the container as the root user (just ensure the `--user root` entry is on a line near the top of [Listing 1](#)) and run a `keytool` command to add your online service's certificate to the `cacerts` file. Next, copy that certificate to your persistent volume.
- With the use of an environment variable when the container is spawned, ensure that Jenkins uses the correct `cacerts` file on your volume to prevent rebuilds and reboots from failing in the future.

The following workflow shows the simple steps needed to add a recognized self-signed certificate to Jenkins.

Step 1

Enter the Jenkins container by running an `exec` command to access the filesystem of the persistent volume created. Find the hash ID of the Jenkins container – in case you didn't name it with the `-name` option in [Listing 1](#). Use the hash ID to enter the container to run the desired commands:

```
$ hash=$(docker ps | grep jenkins | \
    awk '{print $1}')
bf66cc1b2916
$ docker exec -it ${hash} bash
root@bf66cc1b2916:/#
```

The first command will only work if one instance of Jenkins is running; in the unlikely event you are running two Jenkins instances, edit the command accordingly.

Now you should be able to enter your Jenkins container. You have a root prompt inside your container



Figure 1: Switch from HTTPS to HTTP if you can't update plugins automatically – it might just work.

because you left the `--user root \` line in [Listing 1](#). You need to be the root user to run the `keytool` command, but you should remove that line later for much more security. The container should run as the `jenkins` user by default. Line 47 in an online example that shows the Dockerfile used to create the LTS image should convince you [\[4\]](#).

Step 2

Visit the website that you want Jenkins to trust (e.g., with Google Chrome on Linux, [Figure 2](#) – or you can use `openssl`) and save the certificate to a location by accessing the website and clicking on the address bar padlock to download the certificate file locally. The file you are after usually will end in `.pem` – or `.cer` on Windows. Once you have clicked the *Certificate* field highlighted in [Figure 2](#), click the *Details* tab at the top of the next dialog box and then *Export* at the bottom ([Figure 3](#)).

Once *Export* has been clicked, you can follow the prompts to save the certificate locally. On Linux Mint, I just pressed *Save As*; by default, `cloudnativesecurity.cc` is the suggested name because that's the website I visited in my browser, so I adjusted the name slightly to `cloudnativesecurity.pem`. If you open your PEM file, you can see it's a standard certificate file that will look something like:

```
$ cat cloudnativesecurity.cc
-----BEGIN CERTIFICATE-----
MIIF1zCCBL+gAwIBAgIRAK7AdUDa5C4Y1o6SSOX2
```

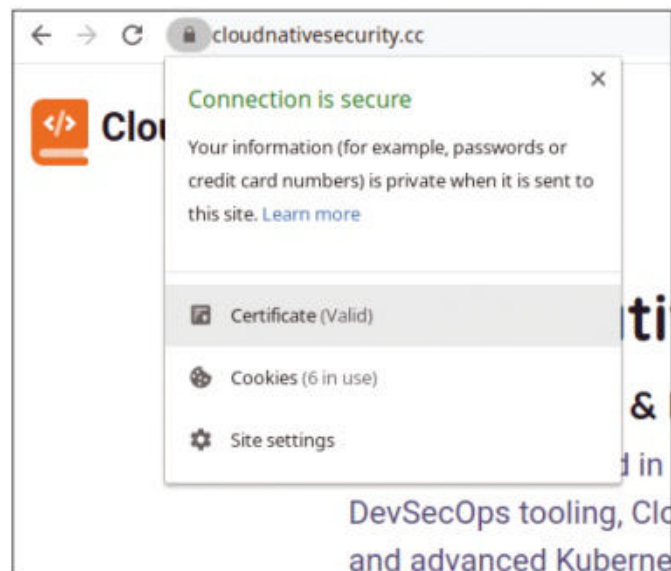


Figure 2: Click the padlock to view certificate information.

```
4aC0wDQYJKoZIhvcNAQEL
...
```

Step 3

Now that you have the certificate you want to trust saved locally, copy it with a Docker `cp` command to the `/tmp` directory in the Jenkins volume, so you can later move it to a more relevant location:

```
$ docker cp cloudnativesecurity.pem 2
${hash}:/tmp/cloudnativesecurity.pem
$ docker exec -it ${hash} ls /tmp
```

Although a relatively simple step, take care with the syntax of this first command, which reuses the hash variable and the filename chosen earlier for the `.pem` file. The last command checks to make sure the file made it to its new location, as hoped. (On my version of Docker, the `cp` command does not seem to show errors if it fails to copy.) If you see your file in the listing, it worked and you can proceed.

Step 4

Now that you have copied the file into your container, you can use your trusty `keytool` command to add it to your trusted certificates by adding the certificate to the sites that Jenkins trusts. This excellent command can

be used to import certificates into the `cacerts` file that Jenkins uses in its keystore, and with a single command you will be asked to confirm whether to trust the self-signed certificate that you have just imported. Before you do that (as I discovered after much reading), the Java virtual machine paths seem to differ in some Jenkins versions, so you need to know precisely where the file resides. For this step, you

need to be inside the container (as per Step 1). Incidentally, as noted, I am entering this container as the root user on purpose, which is required to run `keytool`. Use the `--user root \` option when you are performing these steps, before you complete the process and need to switch from the `jenkins` user (just delete that `--user root \` line in [Listing 1](#) to do so). Now, check where the existing `cacerts` file is from inside the container:

```
root@bf66cc1b2916:/# find / -name cacerts
/usr/local/openjdk-8/jre/lib/security/2
cacerts
```

Found! The next step is updating the local `cacerts` file and then copying it into your persistent volume, having made it readable by the `jenkins` user. The `keytool` command refers to the `/tmp` path from earlier ([Listing 2](#)). After you enter your password (the default appears to be – all lowercase – *changeit*), look at the end of the output where it asks whether you want to trust the certificate. At this point, you can type *yes* to proceed. You are then offered the confirmation *Certificate was added to keystore*. Now that you have updated the `cacerts` file in your container (so that it survives reboots) and the container

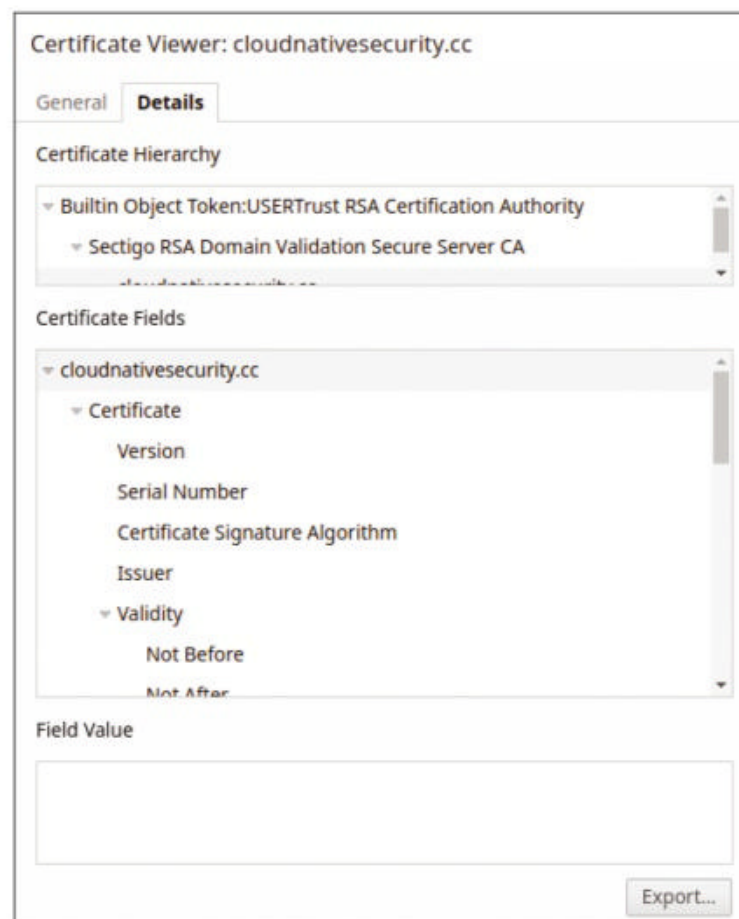


Figure 3: Export the certificate.

receiving the `docker rm` command, copy it to your persistent volume and make sure the *jenkins* user can read from it correctly:

```
$ mkdir /var/jenkins_home/keystore
$ cp /usr/local/openjdk-8/jre/lib/2
security/cacerts /var/jenkins_home/2
keystore/cacerts
$ chown /var/jenkins_home/keystore/cacerts
```

Step 5

Almost finished! You just need to ensure at startup that Jenkins is looking in the correct place for the `cacerts` file to which you have just added your certificate. Add an environment variable that points at your persistent volume path, which the container will use when spawned, by adding the line

```
--env "JENKINS_OPT=2
-Djavax.net.ssl.trustStore=2
/var/jenkins_home/keystore/cacerts"
```

to [Listing 1](#). This step ensures that Jenkins is using the correct `cacerts` file.

Step 6

Restart Jenkins and ensure that changes persist. Because the data is now safe on the persistent volume, stop and delete the older Jenkins container with the commands

```
$ docker stop ${hash}
$ docker rm ${hash}
```

Having made the appropriate changes to the Jenkins start-up script (removing the `--user root` line) and ensuring the addition of the environment vari-

able for the path, you can now execute your `run_jenkins.sh` script again to start up your container once more.

Once you have patiently waited a couple of minutes for the UI to start up, any plugins that interact with <https://cloudnativesecurity.cc> should do so without generating any errors. Rinse and repeat for other online services to which you need to connect.

The End

Having also tried the Skip Certificate Check plugin [\[5\]](#) for Jenkins updates, I was relieved that adjusting the `cacerts` file worked. I tested a number of apparent fixes, such as adding a `/etc/default/jenkins.xml` file.

The benefits of getting this setup working is that, first, your Jenkins instance is able to confirm that it is connecting to the correct online service (and that no imposters are involved). Second, your traffic is transmitted in an encrypted format. I trust this will help you, too, at some point in the future when using Jenkins. ■

Listing 2: Updating cacerts

```
root@bf66cc1b2916:/# keytool -import -alias CNS-cert -keystore /usr/local/openjdk-8/jre/lib/security/
cacerts -file /tmp/cloudnativesecurity.pem
...
#9: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: E2 B9 A7 59 F6 11 B4 00 3B 76 56 1F 29 5D CF 91 ...Y....;vV.)]..
0010: EA AB 17 F6 ....
]
]

Trust this certificate? [no]:
```

Info

- [\[1\] Jenkins: \[https://www.jenkins.io/\]](https://www.jenkins.io/)
- [\[2\] Setting up custom Jenkins Update Center: \[https://medium.com/@prabhas.gupte/how-to-setup-custom-jenkins-update-center-d4bd6d3772d5\]](https://medium.com/@prabhas.gupte/how-to-setup-custom-jenkins-update-center-d4bd6d3772d5)
- [\[3\] Fixing certification error: \[https://stackoverflow.com/questions/24563694/jenkins-unable-to-find-valid-certification-path-to-requested-target-error-while\]](https://stackoverflow.com/questions/24563694/jenkins-unable-to-find-valid-certification-path-to-requested-target-error-while)
- [\[4\] LTS image Dockerfile: \[https://hub.docker.com/r/jenkins/jenkins/tags?page=1&ordering=last_updated\]](https://hub.docker.com/r/jenkins/jenkins/tags?page=1&ordering=last_updated)
- [\[5\] Skip Certificate Check plugin: \[https://plugins.jenkins.io/skip-certificate-check/\]](https://plugins.jenkins.io/skip-certificate-check/)
- [\[6\] Binnie, Chris, and Rory McCune. *Cloud Native Security*. Wiley, 2021, \[https://cloudnativesecurity.cc\]](https://cloudnativesecurity.cc)

Author

Chris Binnie's new book, *Cloud Native Security*, [\[6\]](#) teaches you how to minimize attack surfaces across all of the key components used in modern cloud native infrastructure. Learn with hands-on examples about container security, DevSecOps tooling, advanced Kubernetes security, and Cloud Security Posture Management.

Unleashing accelerated speeds
with RAM drives

Playing with Blocks



Enable and share performant block devices across a network of compute nodes with the RapidDisk kernel RAM drive module. By Petros Koutoupis

Time is money, and sometimes that means you need a faster way to process data. Solid state drives (SSDs) and, more specifically, non-volatile memory express (NVMe) devices have helped alleviate the burden of processing data to and from a backing store. However, at times, even SSD technology is not quite fast enough, which is where the RAM drive comes into the picture. Typically, the RAM drive is used as temporary storage for two reasons: Its capacities tend to be lower (because the technology is more expensive), and more importantly, it is a volatile technology; that is, if the system were to lose power or go into an unstable state, the contents of that RAM drive would disappear. Depending on the type of data being processed, the reward can often outweigh the risks, which is why the RAM drive can potentially be the better option.

In this article, I rely on the RapidDisk suite to create and manage RAM drives. The RapidDisk software project [1] provides an advanced set of Linux kernel RAM drive and caching modules with which you can dynamically create and remove RAM drives

of any size or map them as a temporary read cache to slower devices. The system used in this article is an older system with limited memory clocked at a slow speed. More modern and faster systems with faster memory will produce significantly different results than those found here. The `dmidecode` command summarizes the configuration and capabilities of memory DIMMs and revealed that my system has four DDR3 RAM devices of 2048MB configured at speeds of 1333MTps (mega transfers per second).

Playing with RAM Drives

To begin, you need to download and build the RapidDisk suite from source code by cloning the latest stable code from the RapidDisk repository [2] (Listing 1).

Next, change into the repository's root directory and build and install both the kernel modules and user-space utilities:

```
$ cd rapiddisk
$ make && sudo make install
```

Assuming that all libraries and package dependencies have been installed, both the build and installation should have completed without failure. Now insert the kernel modules:

```
$ sudo modprobe rapiddisk
$ sudo modprobe rapiddisk-cache
```

At this point, if you invoke the `rapiddisk` utility to list all RAM drive targets, none should be listed:

```
$ sudo rapiddisk -l
rapiddisk 7.2.0
Copyright 2011 - 2021 Petros Koutoupis
```

Unable to locate any RapidDisk devices.

The amount of memory to use for your RAM drive needs to be determined carefully. The `rapiddisk` module allocates memory pages as they are requested. In theory, you can create a RAM drive that is much larger than the amount of system memory, but in good practice, this should never be done. As a RAM drive continues to be filled with data, it will eventually run out of free memory pages to allocate, and it could potentially panic the kernel, requiring a reboot of the system. In this example, the system has 8GB of total memory and about 6.5GB of “free” memory (Listing 2).

The following example creates a 1GB RAM drive:

```
$ sudo rapiddisk -a 1024
rapiddisk 7.2.0
Copyright 2011 - 2021 Petros Koutoupis
```

Attached device `rd0` of size 1024 Mbytes

As mentioned earlier, the RapidDisk suite is designed to create and remove RAM drives of any size dynamically. For instance, if you want to create

Listing 1: Cloning RapidDisk

```
$ git clone https://github.com/pkoutoupis/rapiddisk.git
Cloning into 'rapiddisk'...
remote: Enumerating objects: 1560, done.
remote: Counting objects: 100% (241/241), done.
remote: Compressing objects: 100% (158/158), done.
remote: Total 1560 (delta 149), reused 156 (delta 82), pack-reused 1319
Receiving objects: 100% (1560/1560), 762.11 KiB | 5.82 MiB/s, done.
Resolving deltas: 100% (949/949), done.
```


an additional 32MB RAM drive, you would rerun the same command with a different size (**Listing 3**). The output from the second command verifies that the RAM drives were created. Just as easily as it was created, the RAM drive can be removed:

```
$ sudo rapiddisk -d rd1
rapiddisk 7.2.0
Copyright 2011 - 2021 Petros Koutoupis

Detached device rd1
```

To compare the RAM drive with a local 12Gb Serial Attached SCSI (SAS) spinning hard disk drive (HDD) connected to a 6Gb Host Bus Adaptor (HBA), write 1GB worth of sequential data in 1MB transfers to the HDD with dd:

```
$ sudo dd if=/dev/zero of=/dev/sdf 2
bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) 2
copied, 4.76685 s, 225 MB/s
```

The result, 225MBps throughput, is not bad at all for an HDD. To compare its performance with a memory-backed volume, enter:

```
$ sudo dd if=/dev/zero of=/dev/rd0 2
bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) 2
copied, 0.755754 s, 1.4 GB/s
```

Wow! The output shows 1.4GBps. That is nearly six times the throughput

\$ free -m						
	total	used	free	shared	buff/cache	available
Mem:	7951	202	6678	1	1069	7479
Swap:	4095	0	4095			

of the HDD. On a more modern and faster system, that number should and would be higher: about 16 times the throughput or more. Random access I/O is where the memory device shines. Now, test your performance with the fio performance benchmarking utility by running a random write test on the original HDD with 4KB transfers (**Listing 4**). The output shows 1.5MBps, which isn't fast at all. Now, run the same random write test to the RAM drive (**Listing 5**). Here, you can see an impressive 1GBps; again, on a modern system that number would be much higher. You will observe similar results with random read operations (**Listings 6 and 7**). The HDD produces about 2.5MBps, whereas the RAM drive is up to an impressive 1.2GBps.

NVMe over Fabrics Network

Sometimes it might be necessary to export those performant volumes across a network so that other compute nodes can take advantage of the high speed. In the following example, I rely on the NVMe over Fabrics concept and, more specifically, the NVMe target modules provided by the Linux kernel to export the RAM drive and, in turn, import it to another server where it will look and operate like a local storage volume.

Listing 2: RAM Memory

Most modern distributions will have the NVMe target modules installed and available for use. To insert the NVMe and NVMe TCP target modules, enter:

```
$ sudo modprobe nvmet
$ sudo modprobe nvmet-tcp
```

The NVMe target directory tree will need to be made available by the kernel user configuration filesystem, which will provide access to the entire NVMe target configuration environment. To begin, mount the kernel user configuration filesystem and verify that it has been mounted:

Listing 3: Adding RAM Drive of 32MB

```
$ sudo rapiddisk -a 32
rapiddisk 7.2.0
Copyright 2011 - 2021 Petros Koutoupis

Attached device rd1 of size 32 Mbytes
$ sudo rapiddisk -l
rapiddisk 7.2.0
Copyright 2011 - 2021 Petros Koutoupis

List of RapidDisk device(s):

RapidDisk Device 1: rd1    Size (KB): 32768
RapidDisk Device 2: rd0    Size (KB): 1048576

List of RapidDisk-Cache mapping(s):

None
```

Listing 4: HDD Random Write

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=500m --direct=1 --runtime=60 --filename=/dev/sdf --rw=randwrite --numjobs=1 --name=test
test: (g=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=32
fio-3.16
Starting 1 process
Jobs: 1 (f=1): [w(1)][100.0%][w=1733KiB/s][w=433 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=12944: Sat Jun 19 14:49:58 2021
write: IOPS=421, BW=1685KiB/s (1725kB/s)(98.9MiB/60118msec); 0 zone resets
[ ... ]
Run status group 0 (all jobs):
WRITE: bw=1685KiB/s (1725kB/s), 1685KiB/s-1685KiB/s (1725kB/s-1725kB/s), io=98.9MiB (104MB), run=60118-60118msec

Disk stats (read/write):
sdf: ios=51/25253, merge=0/0, ticks=7/1913272, in_queue=1862556, util=99.90%
```

```
$ sudo /bin/mount -t configfs none 2
/sys/kernel/config/
$ mount|grep configfs
configfs on /sys/kernel/config 2
type configfs (rw,relatime)
```

Now, create an NVMe target directory for the RAM drive under the target subsystem and change to that directory (which will host the NVMe target volume plus its attributes):

```
$ sudo mkdir /sys/kernel/config/2
nvmet/subsystems/nvmet-rd0
```

```
$ cd /sys/kernel/config/nvmet/2
subsystems/nvmet-rd0
```

Because this is an example of general usage, you do not necessarily care about which initiators (i.e., hosts) connect to the exported target:

```
$ echo 1 |sudo tee 2
-a attr_allow_any_host > /dev/null
```

Next, create a namespace, change into the directory, set the RAM drive volume as the device for

the NVMe target, and enable the namespace:

```
$ sudo mkdir namespaces/1
$ cd namespaces/1/
$ echo -n /dev/rd0 |sudo tee 2
-a device_path > /dev/null
$ echo 1|sudo tee -a enable > /dev/null
```

Now that you have defined the target block device, you need to switch your focus and define the target (network) port. To create a port directory in the NVMe target tree, change into

Listing 5: RAM Random Write

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=500m --direct=1 --runtime=60 --filename=/dev/rd0 --rw=randwrite --numjobs=1 --name=test
test: (g=0): rw=randwrite, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=32
fio-3.16
Starting 1 process

test: (groupid=0, jobs=1): err= 0: pid=12936: Sat Jun 19 14:48:48 2021
write: IOPS=250k, BW=977MiB/s (1024MB/s)(500MiB/512msec); 0 zone resets
[ ... ]
Run status group 0 (all jobs):
WRITE: bw=977MiB/s (1024MB/s), 977MiB/s-977MiB/s (1024MB/s-1024MB/s), io=500MiB (524MB), run=512-512msec

Disk stats (read/write):
rd0: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
```

Listing 6: HDD Random Read

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=500m --direct=1 --runtime=60 --filename=/dev/sdf --rw=randread --numjobs=1 --name=test
test: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=32
fio-3.16
Starting 1 process
Jobs: 1 (f=1): [r(1)][100.0%][r=2320KiB/s][r=580 IOPS][eta 00m:00s]
test: (groupid=0, jobs=1): err= 0: pid=12975: Sat Jun 19 14:51:27 2021
read: IOPS=622, BW=2488KiB/s (2548kB/s)(146MiB/60127msec)
[ ... ]
Run status group 0 (all jobs):
READ: bw=2488KiB/s (2548kB/s), 2488KiB/s-2488KiB/s (2548kB/s-2548kB/s), io=146MiB (153MB), run=60127-60127msec

Disk stats (read/write):
sdf: ios=37305/0, merge=0/0, ticks=1913563/0, in_queue=1838228, util=99.89%
```

Listing 7: RAM Random Read

```
$ sudo fio --bs=4k --ioengine=libaio --iodepth=32 --size=500m --direct=1 --runtime=60 --filename=/dev/rd0 --rw=randread --numjobs=1 --name=test
test: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=32
fio-3.16
Starting 1 process

test: (groupid=0, jobs=1): err= 0: pid=12967: Sat Jun 19 14:50:18 2021
read: IOPS=307k, BW=1199MiB/s (1257MB/s)(500MiB/417msec)
[ ... ]
Run status group 0 (all jobs):
READ: bw=1199MiB/s (1257MB/s), 1199MiB/s-1199MiB/s (1257MB/s-1257MB/s), io=500MiB (524MB), run=417-417msec

Disk stats (read/write):
rd0: ios=0/0, merge=0/0, ticks=0/0, in_queue=0, util=0.00%
```


that directory, and set the local IP address from which the export will be visible, enter:

```
$ sudo mkdir /sys/kernel/config/nvmet/ports/1
$ cd /sys/kernel/config/nvmet/ports/1
$ echo 10.0.0.185 |sudo tee -a addr_traddr > /dev/null
```

(The IP address in the last command will need to reflect your server configuration.) Now, set the transport type, port number, and protocol version:

```
$ echo tcp|sudo tee -a addr_trtype > /dev/null
$ echo 4420|sudo tee -a addr_trsvcid > /dev/null
$ echo ipv4|sudo tee -a addr_adrfam > /dev/null
```

Note that for any of this to work, both the target and initiator will need port 4420 to be open in its input/output firewall rules.

To tell the NVMe target tree that the port just created will export the block device defined in the subsystem section above, the commands

```
$ sudo ln -s /sys/kernel/config/nvmet/subsystems/nvmet-rd0/ /sys/kernel/config/nvmet/ports/1/subsystems/nvmet-rd0
$ dmesg |grep "nvmet_tcp"
[14798.568843] nvmet_tcp: enabling port 1 (10.0.0.185:4420)
```

link the target subsystem to the target port and verify the export.

Importing to a Remote Server

To use the RAM drive as if it were a locally attached device, move onto a secondary server (i.e., the server that will connect to the exported target).

Because most modern distributions will have the proper NVMe modules installed and available for use, load the initiator or host-side kernel modules:

```
$ sudo modprobe nvme
$ sudo modprobe nvme-tcp
```

Again, for any of this to work, both the target and initiator will need to have port 4420 open in its input/output firewall rules.

Use the nvme command-line utility [3] to discover the NVMe target exported by the target server (Listing 8), connect to the target server, and import the NVMe device(s) it is exporting (in this case, you should see just the one),

```
$ sudo nvme connect -t tcp -n nvmet-rd0 -a 10.0.0.185 -s 4420
```

and verify that the NVMe subsystem sees the NVMe target (Listing 9). You will notice that the RAM drive is enumerated as the second NVMe drive in the list (i.e., /dev/nvme1n1). Now, verify that the volume is listed in the local device listing:

```
$ cat /proc/partitions |grep nvme
259      0  244198584 nvme0n1
259      1  244197543 nvme0n1p1
259      3   1048576 nvme1n1
```

You are now able to read from and write to /dev/nvme1n1 as if the RAM drive were a locally attached device:

```
$ sudo dd if=/dev/zero of=/dev/nvme1n1 bs=1M count=1024
1024+0 records in
1024+0 records out
1073741824 bytes (1.1 GB, 1.0 GiB) copied, 0.868222 s, 1.2 GB/s
```

Finally, type

```
$ sudo nvme disconnect -d /dev/nvme1n1
```

Listing 8: Discover the NVMe Target

```
$ sudo nvme discover -t tcp -a 10.0.0.185 -s 4420

Discovery Log Number of Records 1, Generation counter 2
=====Discovery Log Entry 0=====
trtype: tcp
adrfam: ipv4
subtype: nvme subsystem
treq: not specified, sq flow control disable supported
portid: 1
trsvcid: 4420
subnqn: nvmet-rd0
traddr: 10.0.0.185
sectype: none
```

which will disconnect the NVMe target volume.

Conclusion

At times, it might be necessary to process data at high speeds, and sometimes you can get away with using volatile memory as your backing store. In this article, I looked at the RapidDisk kernel RAM drive module, which not only enables performant block devices, but also makes them shareable across a network of compute nodes. ■

Info

- [1] RapidDisk project: [\[https://github.com/pkoutoupis/rapiddisk/wiki\]](https://github.com/pkoutoupis/rapiddisk/wiki)
- [2] RapidDisk on GitHub: [\[https://github.com/pkoutoupis/rapiddisk\]](https://github.com/pkoutoupis/rapiddisk)
- [3] nvme: [\[https://manpages.debian.org/testing/nvme-cli/nvme.1.en.html\]](https://manpages.debian.org/testing/nvme-cli/nvme.1.en.html)

Author

Petros Koutoupis is currently a senior performance software engineer at Cray (now HPE) for its Lustre High Performance File System division. He is also the creator and maintainer of the RapidDisk Project (www.rapiddisk.org). Koutoupis has worked in the data storage industry for well over a decade and has helped pioneer the many technologies unleashed in the wild today.

Listing 9: Verify the NVMe Target Is Seen

```
$ sudo nvme list
```

Node	SN	Model	Namespace Usage	Format	FW Rev
/dev/nvme0n1	S3ESNX0JA48075E	Samsung SSD 960 EVO 250GB	1 22.41 GB / 250.06 GB	512 B + 0 B	2B7QCXE7
/dev/nvme1n1	07b4753784e26c18	Linux	1 1.07 GB / 1.07 GB	512 B + 0 B	5.4.0-74

Improving performance with environment variables

Trick or No Trick

By using the LD_PRELOAD environment variable, you can improve performance without making changes to applications. By Jeff Layton



A topic that system administrators

learn as they gain experience is called the “LD_PRELOAD Trick.” This trick can help fix misbehaving applications, upgrade applications, and even improve application performance. Of course, it is not really a trick, just the use of a feature in *nix operating systems.

Have you ever installed an application on Linux and tried to run it only to be told the application can’t be found? To debug the issue, probably the first thing to check is your PATH [1], which is “an environment variable ... that tells the shell which directories to search for executable files.” In short, the path tells Linux where to look for applications. If the application is not in the path, then Linux “thinks” it does not exist.

Fortunately, environment variables in Linux can be changed. If Linux cannot find an application, you can change the environment variable, edit the environment variable, or add to the environment variable. Linux uses other environment variables for defining other aspects of the operating system beyond the location of executables. Many times, applications define and use their own environment variables. Users can even define their own environment variables.

In addition to PATH, which helps locate applications, the LD_LIBRARY_PATH environment variable tells Linux where to search for the shared libraries used by applications, which allows you to control which libraries are “available.” Like PATH, this variable can be changed, and each shell can have its own value.

The variable can be useful when debugging a new library because you can simply change LD_LIBRARY_PATH to the new library, test it, and then change it back. You can also use it when upgrading libraries. If there is little or no change to the APIs in the new library, then a simple change to LD_LIBRARY_PATH allows you to use the new library without changing anything else.

A third environment variable that also works with libraries, and is at the heart of the “trick,” is LD_PRELOAD, an environment variable that contains a delimited list of shared objects (libraries) [2] that are loaded before all others. This variable allows you to have more control over the order that libraries are found by the application than just LD_LIBRARY_PATH.

LD_PRELOAD can be a great help in debugging because you can set it to a new library without changing

LD_LIBRARY_PATH. After debugging, just set LD_PRELOAD to its previous value.

Perhaps the greatest strength of LD_PRELOAD is that you can easily substitute a new library for an existing one, allowing you to upgrade a library in an attempt to get better performance. Inserting a library before another for whatever purpose you have in mind is the so-called LD_PRELOAD trick.

One use I’ve seen of LD_PRELOAD is to load a library that intercepts calls to a normal library. The “intercept library” uses the same symbols (functions) as the usual library so that it will intercept any function calls from the application that were intended for that library. This intercept library can then be used to gather telemetry information from the calling application, perhaps writing it to a file. The intercept library then calls the intended functions in the usual library. With LD_PRELOAD, you can load the intercept library before the usual library without having to change it or the application.

A classic use case for an intercept library is for gathering telemetry (information) about I/O functions. With LD_PRELOAD, the intercept

Listing 1: Single-Precision Square Matrix Multiply

```
# Example SGEMM

for N = [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192]

    A = single( rand(N,N) );
    B = single( rand(N,N) );

    start = clock();
    C = A*B;
    elapsedTime = etime(clock(), start);

    gFlops = 2*N*N*N / (elapsedTime * 1e+9);

    disp(sprintf("N = %4d, elapsed Time = %9.6f, GFlops = %9.6f ", ...
        N, elapsedTime, gFlops) );

endfor
```

Listing 2: Double-Precision Square Matrix Multiply

```
# Example DGEMM

for N = [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192]

    A = double( rand(N,N) );
    B = double( rand(N,N) );

    start = clock();
    C = A*B;
    elapsedTime = etime(clock(), start);

    gFlops = 2*N*N*N / (elapsedTime * 1e+9);

    disp(sprintf("N = %4d, elapsed Time = %9.6f, GFlops = %9.6f ", ...
        N, elapsedTime, gFlops) );

endfor
```

library intercepts I/O function calls such as `open()`, `close()`, `read()`, and `write()` to gather information and then passes the function calls to the intended I/O library. The intercept library uses the same function names, but rather than rewrite the I/O functionality for these functions, the new library typically gathers information, writes it to a file, and then calls the normal library to perform the I/O functions. Although this example is a classic use case of LD_PRELOAD, it is not the only use case. The next section presents another use of LD_PRELOAD resulting in increased performance.

Octave

Probably one of the best examples I know for the use of the LD_PRELOAD trick is to push basic linear algebra subprogram (BLAS) [3] computations from a CPU onto an NVidia GPU. I will illustrate this with an example from Octave [4], a mathematics tool similar to Matlab [5]. To demonstrate the process, I'll use two Octave scripts: The first does a simple square matrix multiply in single precision for various matrix sizes (Listing 1). The second script (Listing 2) is the same as Listing 1, but uses double precision. To begin, I'll run these scripts on a test system with the default BLAS library that comes with Octave; then,

I can use the LD_PRELOAD trick to have Octave call a different BLAS library, resulting in different, conceivably better, performance. The test system is my Linux laptop (see Table 1 for specifications). The laptop runs Ubuntu 20.04 with the 455.45.01 NVidia driver, and CUDA 11.2. Octave 5.2.0 was used for the tests. All software was installed from the Apt repository for the specific distribution version. The two scripts were run several times (> 15) for each case to get a feel for the performance; then, they were run for the results presented in this article.

Default BLAS Library

By default, Octave uses a multi-threaded BLAS library. Specifically, Octave used the BLAS library located at `/lib/x86_64-linux-gnu/libblas.so.3`. The two scripts, one for single precision and one for double precision, were run under the default BLAS library. The straightforward command to run the single-precision code with all cores (the default) is:

```
$ octave-cli ./sgemm.m
```

To run with a single core, you modify the command slightly:

```
$ OMP_NUM_THREADS=1 octave-cli ./sgemm.m
```

The results for running the two scripts are presented in Table 2 (where GFLOPS is a billion floating-point operations per second). First, they are run on a single core, and then on all cores. A fair amount of variability is evident for $N=256$ and $N=512$, which is also true for all subsequent CPU results.

OpenBLAS

One of the most popular BLAS libraries is OpenBLAS [8], which you can

Table 1: Test System Specs

CPU: Intel Core i5-10300H CPU [6] @2.50GHz	
Processor base frequency	2.5GHz
Max turbo frequency	4.5GHz
Cache	8MB
Four cores (eight with hyper-threading)	
45W TDP	
8GB DDR4-2933 memory	
Maximum of two memory channels	
Memory bandwidth	45.8GBps
NVidia GeForce 1650 GPU [7]	
Architecture: Turing (TU117)	
Memory	4GB GDDR5
Memory speed	8bps
Memory bandwidth	128GBps
Memory bus	8-bit
L2 cache	1MB
TDP	75W
Base clock	1,485GHz
Boost clock	1,665MHz
896 CUDA cores	

Table 2: Octave Results with Default BLAS Library								
N	Single-Precision, One Core		Double-Precision, One Core		Single-Precision, All Cores		Double-Precision, All Cores	
	Elapsed Time (secs)	GFLOPS	Elapsed Time (secs)	GFLOPS	Elapsed Time (secs)	GFLOPS	Elapsed Time (secs)	GFLOPS
2	0.000702	0.000023	0.000427	0.000037	0.000961	0.000017	0.000137	0.000117
4	0.000069	0.001864	0.000076	0.001678	0.000099	0.001291	0.00092	0.001398
8	0.000069	0.014913	0.000061	0.016777	0.000092	0.011185	0.000084	0.012202
16	0.000061	0.134218	0.000061	0.134218	0.000092	0.089478	0.000084	0.097613
32	0.000076	0.858993	0.000076	0.858993	0.000099	0.660764	0.000107	0.613567
64	0.000099	5.286114	0.000145	3.616815	0.000153	3.435974	0.000206	2.545166
128	0.000313	13.408678	0.000587	7.139686	0.000565	7.429133	0.000473	8.867029
256	0.001785	18.795071	0.003654	9.181725	0.000542	61.944317	0.001144	29.32031
512	0.013779	19.481934	0.027763	9.668693	0.0047	57.117487	0.022438	11.963404
1,024	0.100395	21.390301	0.215065	9.985277	0.02961	72.526405	0.055252	38.867022
2,048	0.776039	22.137891	1.612694	10.652902	0.199173	86.256026	0.455025	37.755903
4,096	5.855209	23.472936	12.275261	11.196418	1.575951	87.21019	3.468651	39.623174
8,192	39.343849	27.946214	102.974144	10.677551	12.247917	89.771315	26.561623	41.394746

use with the PRELOAD trick instead of the default BLAS library. The command to run the single-precision script is:

```
$ OMP_NUM_THREADS=1
LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libopenblas.so.0 octave-cli ./sgemm.m
```

Table 3 contains the results. Note that the OpenBLAS library is installed from the Apt repository for this distribution and version. Likely, one built on the system could produce better results.

NVBLAS

NVidia has several libraries you can use when writing programs. Some of these libraries are standard conforming libraries, such as cuBLAS. NVidia has taken cuBLAS and used it as part of a “drop-in” replacement BLAS library, NVBLAS, that provides BLAS level 3 routines. NVBLAS uses cuBLAS, both of which are included as part of CUDA; simply follow the directions for downloading and installing CUDA. For this article, I used the cuBLAS and NVBLAS that came with the NVidia HPC SDK, version 21.3.

Before using NVBLAS, you have to configure it. From the NVBLAS documentation, “It must be configured through an ASCII text file that describes how many and which GPUs can participate in the intercepted BLAS calls.” To use NVBLAS, create the file nvblas.conf in the directory in which you are running the scripts. For the example in this article, the contents of the file I used were:

```
# This is the configuration
file to use NVBLAS Library
NVBLAS_LOGFILE nvblas.log
```

Table 3: Octave Results with OpenBLAS Library								
N	Single-Precision, One Core		Double-Precision, One Core		Single-Precision, All Cores		Double-Precision, All Cores	
	Elapsed Time (secs)	GFLOPS	Elapsed Time (secs)	GFLOPS	Elapsed Time (secs)	GFLOPS	Elapsed Time (secs)	GFLOPS
2	0.000114	0.00014	0.000114	0.00014	0.001022	0.000016	0.000771	0.000021
4	0.000076	0.001678	0.000076	0.001678	0.000099	0.001291	0.000061	0.002097
8	0.000061	0.016777	0.000061	0.016777	0.000092	0.011185	0.000061	0.016777
16	0.000061	0.134218	0.000069	0.119305	0.000084	0.097613	0.000076	0.107374
32	0.000061	1.073742	0.000076	0.858993	0.000092	0.715828	0.000076	0.858993
64	0.000099	5.286114	0.000137	3.817749	0.000145	3.616815	0.000137	3.817749
128	0.000313	13.408678	0.000572	7.330078	0.000381	10.995116	0.000656	6.392509
256	0.001808	18.557158	0.003624	9.259045	0.000519	64.677155	0.001144	29.32031
512	0.013237	20.279177	0.026962	9.955963	0.004074	65.888337	0.008163	32.882591
1,024	0.101677	21.120656	0.20388	10.533061	0.035118	61.150332	0.052483	40.918008
2,048	0.774956	22.168839	1.59137	10.79565	0.201546	85.240558	0.410416	41.859683
4,096	5.741043	23.939718	11.007278	12.486188	1.558258	88.20038	3.523735	39.003771
8,192	39.33165	27.954882	84.512154	13.010101	12.305489	89.351318	26.867691	40.92319

Table 4: Octave Results with the NVBLAS Library

N	Single-Precision, GPU		Double-Precision, GPU	
	Elapsed Time (secs)	GFLOPS	Elapsed Time (secs)	GFLOPS
2	0.001167	0.000014	0.001007	0.000016
4	0.000076	0.001678	0.000069	0.001864
8	0.000061	0.016777	0.000061	0.016777
16	0.000061	0.134218	0.000069	0.119305
32	0.000076	0.858993	0.000076	0.858993
64	0.000099	5.286114	0.000145	3.616815
128	0.000542	7.74304	0.000603	6.958934
256	0.000549	61.083979	0.001152	29.126136
512	0.016685	16.087962	0.012955	20.721067
1,024	0.008904	241.195353	0.039238	54.72975
2,048	0.01741	986.765913	0.250496	68.583432
4,096	0.093765	1465.776933	1.500099	91.619911
8,192	0.643051	1709.835418	12.03125	91.387979

```
NVBLAS_CPU_BLAS_LIB /usr/lib/
x86_64-linux-gnu/libopenblas.so.0
NVBLAS_GPU_LIST 0
NVBLAS_AUTOPIN_MEM_ENABLED
```

The first line of the file defines the logfile where NVBLAS writes any log information. The next line defines the CPU-only BLAS library for cases in which there is no GPU routine. The code defaults to running on the CPU and falls through to the CPU BLAS library, which the NVBLAS_CPU_BLAS_LIB variable specifies for NVBLAS. In this case, I chose to use the OpenBLAS library.

The third line lists the GPU devices that should be used. The numbering begins with 0. In this case, the laptop only has one NVidia GPU, so only one is listed. You can also use the keyword ALL to define all the GPUs in the system. The last line is something I used from an article about NVBLAS with Octave [13]. After configuring nvblas.conf, you have to take two steps to run Octave. The first step is to export the NVBLAS_CONFIG_FILE environment variable that points to the location of the nvblas.conf file:

```
export NVBLAS_CONFIG_FILE=$HOME/
PROJECTS/OCTAVE/nvblas.conf
```

This environment variable just points to the ASCII configuration file you created. The second step is the run

command itself, which uses the LD_PRELOAD trick to load NVBLAS first:

```
LD_PRELOAD=/opt/nvidia/hpc_sdk/
Linux_x86_64/21.3/math_libs/11.2/
targets/x86_64-linux/lib/
libnvblas.so.11.4.1.1026 octave-cli ./
sgemm.m
```

The command begins by defining LD_PRELOAD, pointing to the NVBLAS library, which is then followed by the command that runs Octave (octave-cli). To run the script, you can simply concatenate the two commands together (I tend to write a one-line Bash script for this). The results for the single- and double-precision scripts are shown in Table 4.

The strange “blurb” in the results for N=512 I cannot explain, but it happens very frequently. Notice the strange results at N=256 and N=512 that also happened when using the CPU. For the CPU results, the double-precision results are about half the single-precision results, which is expected. However, the GPU double-precision performance is less than half of the single-precision results, because the GPU used (the GeForce 1650) is a consumer-grade GPU with the focus primarily on 32-bit performance. However, as you can tell, it can run double-precision code, just not as well as the data center GPUs that focus on 64-bit performance.

Summary

The PRELOAD trick is something of a rite of passage for new system administrators. When they find out about the trick, it is a revelation because of how flexible it can be. Soon, it is no longer a trick but a part of what the admin uses every day. I hope the simple example of LD_PRELOAD in this article with GPUs for computation and without any code changes illustrates its utility.

If you knew of this trick but have forgotten it, or if you are just learning it, I hope this article proved useful. ■

Info

[1] PATH: [http://www.linfo.org/path_env_var.html]
[2] Shared objects: [https://man7.org/linux/man-pages/man8/ld.so.8.html]
[3] BLAS: [https://en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms]
[4] Octave: [https://www.gnu.org/software/octave/index]
[5] Matlab: [https://www.mathworks.com/]
[6] i5-10300H CPU: [https://ark.intel.com/content/www/us/en/ark/products/201839/intel-core-i5-10300h-processor-8m-cache-up-to-4-50-ghz.html]
[7] NVidia GeForce 1650 GPU: [https://www.nvidia.com/en-us/geforce/graphics-cards/gtx-1650/]
[8] OpenBLAS: [https://en.wikipedia.org/wiki/OpenBLAS]
[9] cuBLAS: [https://developer.nvidia.com/cublas]
[10] BLAS Level 3 routines: [https://docs.nvidia.com/cuda/nvblas/index.html]
[11] CUDA: [https://developer.nvidia.com/cuda-toolkit]
[12] NVBLAS documentation: [https://docs.nvidia.com/cuda/nvblas/index.html#configuration-file]
[13] NVBLAS with Octave: [https://developer.nvidia.com/blog/drop-in-acceleration-gnu-octave/]

The Author

Jeff Layton has been in the HPC business for almost 25 years (starting when he was 4 years old). He can be found lounging around at a nearby Frys enjoying the coffee and waiting for sales.

When I/O workloads don't perform

A Peak Under the Hood

Every now and then, you find yourself in a situation where you expect better performance from your data storage drives. Either they once performed very well and one day just stopped, or they came straight out of the box underperforming. We explore a few of the reasons why this might happen. By Petros Koutoupis

Listing 1: sg_logs

```
$ sudo sg_logs /dev/sdc
SEAGATE ST14000NM0001 K001
Supported log pages [0x0]:
0x00 Supported log pages [sp]
0x02 Write error [we]
0x03 Read error [re]
0x05 Verify error [ve]
0x06 Non medium [nm]
0x08 Format status [fs]
0x0d Temperature [temp]
0x0e Start-stop cycle counter [sscc]
0x0f Application client [ac]
0x10 Self test results [str]
0x15 Background scan results [bsr]
0x18 Protocol specific port [psp]
0x1a Power condition transitions [pct]
0x2f Informational exceptions [ie]
0x37 Cache (seagate) [c_se]
0x38
0x3e Factory (seagate) [f_se]
```

Listing 2: Log Page for Write Errors

```
$ sudo sg_logs -p 0x2 /dev/sdc
SEAGATE ST14000NM0001 K001
Write error counter page [0x2]
Errors corrected with possible delays = 0
Total rewrites or rereads = 0
Total errors corrected = 0
Total times correction algorithm processed = 0
Total bytes processed = 3951500537856
Total uncorrected errors = 0
```

Sometimes, the easiest and quickest way to determine the root cause of a slow drive is to check its local logging data. The method by which this log data is stored will differ by the drive type, but in the end, the results are generally the same. For instance, a SCSI-based drive such as a Serial Attached SCSI (SAS) drive collects drive log data and general metrics in something called the SCSI log pages (plural because each page separates the collected data into its respective category). The easiest way to access this data is by using the `sg3_utils` package available for Linux. To find out what categories the drive supports, execute the `sg_logs` binary with the SAS drive or SCSI generic identifier in which you are interested (**Listing 1**).

As you can see, you can observe data for write, read, and drive temperature errors, and more. To specify a specific page, you will need to use the `-p` parameter followed by the page number. For instance, look at the log page for write errors (i.e., 0x2; **Listing 2**). Seemingly, this drive does not have any write errors (corrected and uncorrected by the drive firmware), so it looks to be in good shape. Typically,

if you see errors, especially of the uncorrected type, the printout will include failing logical block addresses (LBAs). If the failed LBA regions (i.e., sectors) were listed under the read error category, it would likely be in a *pending reallocation* state (waiting for a future write to the same address). A sector pending reallocation is a sector that is unable to be read from or written to and must be reallocated elsewhere on the disk drive. This reallocation will only happen on the next write operation to that failed sector, if the drive has spare sectors it can use to relocate the data. A failing sector or a sector pending reallocation by the drive's firmware will affect overall drive performance, and if enough of it occurs, it would be highly recommended to replace the drive as soon as possible.

Another thing that needs to be understood is that if a log page starts to list a significant count of corrected read or write errors, chances are that the disk drive's surrounding environment may be at fault. For instance, vibration will often cause a disk drive's head to misread or miswrite a length of sectors on a drive track, which

results in the firmware taking action to correct it. This process alone will introduce unwanted I/O latencies (reducing performance to the drive).

If you'd like to list all of the log pages at once, use the `-a` parameter (**Listing 3**). (Warning: You will get a lot of information.)

Other tools exist to extract similar and sometimes the same amount of data from a SAS drive (e.g., `smartctl`). If a drive supports the industry standard Self-Monitoring, Analysis and Reporting Technology (S.M.A.R.T.), you can use the `smartmontools` package and, again, more specifically, the `smartctl` binary (**Listing 4**).

The `smartmontools` package is most beneficial for Serial ATA (SATA) drives, because most SATA drives tend to support the feature out of the box. Note that the S.M.A.R.T. output, such as the type of data and the way it is formatted, will differ on SATA drives from its SAS counterpart (**Listing 5**).

For the most part, the information is generally the same. For instance, when you look at the drive attributes category, attribute 197 or `Current_Pending_Sector` is the same sector pending

Listing 3: List All Log Pages

```
$ sudo sg_logs -a /dev/sdc
SEAGATE ST14000NM0001 K001

Supported log pages [0x0]:
 0x00 Supported log pages [sp]
 0x02 Write error [we]
 0x03 Read error [re]
 0x05 Verify error [ve]
 0x06 Non medium [nm]
 0x08 Format status [fs]
 0x0d Temperature [temp]
 0x0e Start-stop cycle counter [sscc]
 0x0f Application client [ac]
 0x10 Self test results [str]
 0x15 Background scan results [bsr]
 0x18 Protocol specific port [psp]
 0x1a Power condition transitions [pct]
 0x2f Informational exceptions [ie]
 0x37 Cache (seagate) [c_se]
 0x38
 0x3e Factory (seagate) [f_se]

Write error counter page [0x2]
Errors corrected with possible delays = 0
Total rewrites or rereads = 0
Total errors corrected = 0
Total times correction algorithm processed = 0
Total bytes processed = 3951500537856
Total uncorrected errors = 0
Reserved or vendor specific [0xf800] = 0
Reserved or vendor specific [0xf801] = 0
Reserved or vendor specific [0xf802] = 0
Reserved or vendor specific [0xf803] = 0
Reserved or vendor specific [0xf804] = 0
Reserved or vendor specific [0xf805] = 0
Reserved or vendor specific [0xf806] = 0
Reserved or vendor specific [0xf807] = 0
Reserved or vendor specific [0xf810] = 0
Reserved or vendor specific [0xf811] = 0
Reserved or vendor specific [0xf812] = 0
Reserved or vendor specific [0xf813] = 0
Reserved or vendor specific [0xf814] = 0
Reserved or vendor specific [0xf815] = 0
Reserved or vendor specific [0xf816] = 0
Reserved or vendor specific [0xf817] = 0

Reserved or vendor specific [0xf820] = 0

Read error counter page [0x3]
Errors corrected without substantial delay = 0
Errors corrected with possible delays = 0
Total rewrites or rereads = 0
Total errors corrected = 0
Total times correction algorithm processed = 0
Total bytes processed = 35801804845056
Total uncorrected errors = 0

Verify error counter page [0x5]
Errors corrected without substantial delay = 0
Errors corrected with possible delays = 0
Total rewrites or rereads = 0
Total errors corrected = 0
Total times correction algorithm processed = 0
Total bytes processed = 0
Total uncorrected errors = 0

Non-medium error page [0x6]
Non-medium error count = 0

Format status page [0x8]
Format data out: <not available>
Grown defects during certification <not available>
Total blocks reassigned during format <not available>
Total new blocks reassigned <not available>
Power on minutes since format <not available>

Temperature page [0xd]
Current temperature = 28 C
Reference temperature = 60 C

Start-stop cycle counter page [0xe]
Date of manufacture, year: 2019, week: 26
Accounting date, year: , week:
Specified cycle count over device lifetime = 50000
Accumulated start-stop cycles = 498
Specified load-unload count over device lifetime = 600000
Accumulated load-unload cycles = 553

[ ... ]
```

Listing 4: smartctl on SAS Drive

```
$ sudo smartctl -a /dev/sdc
smartctl 7.1 2019-12-30 r5022 [x86_64-linux-5.4.0-66-generic] (local build)
Copyright (C) 2002-19, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Vendor:                SEAGATE
Product:               ST14000NM0001
Revision:              K001
Compliance:            SPC-5
User Capacity:         7,000,259,821,568 bytes [7.00 TB]
Logical block size:    4096 bytes
LU is fully provisioned

Rotation Rate:         7200 rpm
Form Factor:           3.5 inches
Logical Unit id:       0x6000c500a7b3ceeb0000000000000000
Serial number:         ZKL00CYG0000G925020A
Device type:           disk
Transport protocol:    SAS (SPL-3)
Local Time is:         Sun Mar 21 15:00:00 2021 UTC
SMART support is:      Available - device has SMART capability.
SMART support is:      Enabled
Temperature Warning:   Disabled or Not Supported
```

Listing 4: smartctl on SAS Drive (continued)

```
=== START OF READ SMART DATA SECTION ===
SMART Health Status: OK

Grown defects during certification <not available>
Total blocks reassigned during format <not available>
Total new blocks reassigned <not available>
Power on minutes since format <not available>
Current Drive Temperature:      28 C
Drive Trip Temperature:         60 C

Manufactured in week 26 of year 2019
Specified cycle count over device lifetime: 50000
Accumulated start-stop cycles: 498
Specified load-unload count over device lifetime: 600000
Accumulated load-unload cycles: 553
Elements in grown defect list: 0

Vendor (Seagate Cache) information
  Blocks sent to initiator = 150743545

  Blocks received from initiator = 964465354
  Blocks read from cache and sent to initiator = 1014000851
  Number of read and write commands whose size <= segment size = 8318611
  Number of read and write commands whose size > segment size = 12

Vendor (Seagate/Hitachi) factory information
  number of hours powered up = 264.67
  number of minutes until next internal SMART test = 48

Error counter log:
   Errors Corrected by      Total   Correction   Gigabytes   Total
      ECC      rereads/    errors   algorithm   processed   uncorrected
   fast | delayed  rewrites  corrected  invocations [10^9 bytes] errors
read:   0         0         0         0           0       35801.805        0
write:  0         0         0         0           0       3951.501         0

Non-medium error count:           0

No Self-tests have been logged
```

reallocation discussed earlier. Again, you can gather drive temperature information, lifetime hours, and more.

How About CPU and Drive Utilization?

Now you have checked all your drives, but for some reason, they are still not performing as expected. The next step should be to determine whether drive utilization is too high or the CPU is too busy and is having a difficult time keeping up with I/O requests. The `sysstat` package provides a nice little utility called `iostat` that gathers both sets of data. In the example in Listing 6, `iostat` is showing an extended set of metrics and the CPU utilization at two-second intervals. The first interval should probably be ignored because `iostat` has no real data with which to compare (i.e., no initial state), so the numbers look a bit off. Once the utility stabilizes by the second interval, you will see a more accurate picture of your disk drive and what it reports with reads per second (r/s), writes per second (w/s), average I/O waiting to complete in reads (r_await) and writes (w_await), a calculation of how much of the drive is in use (%util), and more. The higher the %util number, the busier the drive is likely to be

completing I/O requests. If that number is high, you might need to con-

sider methods either to throttle the amount of I/O sent to the drive or find

Listing 5: smartctl on SATA Drive

```
$ sudo smartctl -a /dev/sda
smartctl 7.1 2019-12-30 r5022 [x86_64-linux-5.4.0-66-generic] (local build)
Copyright (C) 2002-19, Bruce Allen, Christian Franke, www.smartmontools.org

=== START OF INFORMATION SECTION ===
Model Family:      Seagate Constellation ES (SATA 6Gb/s)
Device Model:      ST500NM0011
Serial Number:     Z1M11WAJ
LU WWN Device Id:  5 000c50 04edcb79a
Add. Product Id:   DELL(tm)
Firmware Version:  PA08
User Capacity:     500,107,862,016 bytes [500 GB]
Sector Size:       512 bytes logical/physical
Rotation Rate:     7200 rpm
Form Factor:       3.5 inches
Device is:         In smartctl database [for details use: -P show]
ATA Version is:    ATA8-ACS T13/1699-D revision 4
SATA Version is:   SATA 3.0, 3.0 Gb/s (current: 3.0 Gb/s)
Local Time is:     Sun Mar 21 15:00:38 2021 UTC
SMART support is:  Available - device has SMART capability.
SMART support is:  Enabled

=== START OF READ SMART DATA SECTION ===
SMART overall-health self-assessment test result: PASSED

General SMART Values:
Offline data collection status: (0x82)  Offline data collection activity
was completed without error.
Auto Offline Data Collection: Enabled.
Self-test execution status:      (  0)  The previous self-test routine completed
without error or no self-test has ever
been run.
Total time to complete Offline
data collection:      ( 609) seconds.
Offline data collection
capabilities:          (0x7b) SMART execute Offline immediate.
Auto Offline data collection on/off support.
Suspend Offline collection upon new
command.
Offline surface scan supported.
Self-test supported.
```


ways to balance the same I/O across multiple drives (e.g., in a RAID0, 5, or 6 configuration).

Also, notice the average CPU metrics at the top of each interval. Here, you will find a breakdown of how much of the collected CPU is busy performing tasks, waiting on completion of tasks (%iowait), idling, and so on. The less idle in the system, the more affected your drive performance. You can view a real-time breakdown of these CPU cores with the top utility. After opening the top application at the command line, press the *l* key (Listing 7).

Enough Free Memory?

If the CPU is not the problem and the drives are being underutilized, do you have a constraint on memory resources? The easiest and quickest way to check is with free (Listing 8). The free utility dumps the amount of total, used, and free memory on the system, but it will also show how much of it is used as a buffer or temporary cache (buff/cache) and how much of it is available and reclaimable (available). The output is in megabytes – the -g argument reports output in gigabytes and -k in kilobytes – and will match the data found in /proc/meminfo. Note that the output only looks different from the free output because it is calculated in kilobytes instead of megabytes:

```
$ cat /proc/meminfo | grep -e "^Mem"
MemTotal:      8142012 kB
MemFree:       7204048 kB
MemAvailable:  7672148 kB
```

When available memory starts to increase while free memory drastically decreases, a lot of memory is being consumed by the system and its applications, a percentage of which can be reclaimed from temporary caches when the operating system is under memory pressure. If the system begins to reclaim memory, it will affect overall system performance. You will also observe a kswapd or swapper message in the dmesg or syslog output, indicating that the kernel is hard at

Listing 5: smartctl on SATA Drive (continued)

```
Conveyance Self-test supported.
Selective Self-test supported.
SMART capabilities:          (0x0003)  Saves SMART data before entering
                                power-saving mode.
                                Supports SMART auto save timer.
Error logging capability:    (0x01)    Error logging supported.
                                General Purpose Logging supported.
Short self-test routine
recommended polling time:    (  2) minutes.
Extended self-test routine
recommended polling time:    ( 75) minutes.
Conveyance self-test routine
recommended polling time:    (  3) minutes.
SCT capabilities:            (0x10bd)    SCT Status supported.
                                SCT Error Recovery Control supported.
                                SCT Feature Control supported.
                                SCT Data Table supported.

SMART Attributes Data Structure revision number: 10
Vendor Specific SMART Attributes with Thresholds:
ID# ATTRIBUTE_NAME          FLAG     VALUE WORST THRESH TYPE      UPDATED  WHEN_FAILED RAW_VALUE
  1 Raw_Read_Error_Rate     0x000f   077    063   ---   Pre-fail Always      -       56770409
  3 Spin_Up_Time             0x0003   096    092   ---   Pre-fail Always      -         0
  4 Start_Stop_Count         0x0032   100    100   ---   Old_age Always      -       137
  5 Reallocated_Sector_Ct    0x0033   100    100   ---   Pre-fail Always      -         0
  7 Seek_Error_Rate          0x000f   067    060   ---   Pre-fail Always      -     5578572
  9 Power_On_Hours           0x0032   100    100   ---   Old_age Always      -        400
 10 Spin_Retry_Count         0x0013   100    099   ---   Pre-fail Always      -         0
 12 Power_Cycle_Count        0x0032   100    100   ---   Old_age Always      -       135
184 End-to-End_Error         0x0032   100    100   ---   Old_age Always      -         0
187 Reported_Uncorrect       0x0032   100    100   ---   Old_age Always      -         0
188 Command_Timeout          0x0032   100    100   ---   Old_age Always      -         0
189 High_Fly_Writes          0x003a   100    100   ---   Old_age Always      -         0
190 Airflow_Temperature_Cel 0x0022   072    058   ---   Old_age Always      -        28 (Min/Max
22/28)
191 G-Sense_Error_Rate       0x0032   100    100   ---   Old_age Always      -         0
192 Power-Off_Retract_Count  0x0032   100    100   ---   Old_age Always      -        35
193 Load_Cycle_Count         0x0032   100    100   ---   Old_age Always      -       487
194 Temperature_Celsius      0x0022   028    042   ---   Old_age Always      -     28 (0 18 0 0 0)
195 Hardware_ECC_Recovered   0x001a   113    099   ---   Old_age Always      -     56770409
197 Current_Pending_Sector    0x0012   100    100   ---   Old_age Always      -         0
198 Offline_Uncorrectable     0x0010   100    100   ---   Old_age Offline     -         0
199 UDMA_CRC_Error_Count      0x003e   200    200   ---   Old_age Always      -         0
240 Head_Flying_Hours        0x0000   100    253   ---   Old_age Offline     -     344 (218 109 0)
241 Total_LBAs_Written        0x0000   100    253   ---   Old_age Offline     -    1389095282
242 Total_LBAs_Read           0x0000   100    253   ---   Old_age Offline     -    619165492

SMART Error Log Version: 1
No Errors Logged

SMART Self-test log structure revision number 1
Num Test_Description      Status                Remaining  LifeTime(hours)  LBA_of_first_error
# 1 Short offline          Completed without error   00%           2                -
# 2 Extended offline        Completed without error   00%           2                -

SMART Selective self-test log data structure revision number 1
SPAN  MIN_LBA  MAX_LBA  CURRENT_TEST_STATUS
  1      0      0  Not_testing
  2      0      0  Not_testing
  3      0      0  Not_testing
  4      0      0  Not_testing
  5      0      0  Not_testing

Selective self-test flags (0x0):
  After scanning selected spans, do NOT read-scan remainder of disk.
If Selective self-test is pending on power-up, resume after 0 minute delay.
```

Listing 6: iostat Output

```
$ iostat -x -d 2 -c
Linux 5.4.12-050412-generic (dev-machine)      03/14/2021      _x86_64_      (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.79    0.07    1.19    2.89    0.00   95.06

Device            r/s     w/s    rkB/s    wkB/s  rrqm/s  wrqm/s  %rrqm  %wrqm  r_await  w_await  aqu-sz  rareq-sz  wareq-sz  svctm  %util
sda                10.91    6.97   768.20   584.64    4.87   18.20   30.85   72.31   13.16   20.40    0.26    70.44    83.89    1.97   3.52
nvme0n1           58.80   12.22  17720.47   48.71   230.91    0.01   79.70    0.08    0.42    0.03    0.00   301.34     3.98    1.02   7.24
sdb                0.31   55.97     4.13  17676.32    0.00   231.64    0.00   80.54    2.50    8.47    0.32    13.45   315.84    1.30   7.32
sdc                0.24    0.00     3.76    0.00    0.00    0.00    0.00    0.00    2.47    0.00    0.00    15.64     0.00    1.03   0.02
sde                2.47    0.00    62.57    0.00    0.00    0.00    0.00    0.00    0.63    0.00    0.00    25.34     0.00    0.29   0.07
sdf                1.51    0.00    32.42    0.00    0.00    0.00    0.00    0.00    0.69    0.00    0.00    21.40     0.00    0.31   0.05
sdd                1.42    0.00    50.96    0.00    0.00    0.00    0.00    0.00    0.44    0.00    0.00    35.83     0.00    0.38   0.05
md0               12.43   12.17    54.39    48.68    0.00    0.00    0.00    0.00    0.00    0.00    0.00     4.37     4.00    0.00   0.00

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.76    0.00    3.03    1.26    0.00   94.95

Device            r/s     w/s    rkB/s    wkB/s  rrqm/s  wrqm/s  %rrqm  %wrqm  r_await  w_await  aqu-sz  rareq-sz  wareq-sz  svctm  %util
sda                0.00    9.00     0.00    88.00    0.00    8.00    0.00   47.06    0.00   30.83    0.26     0.00     9.78    0.67   0.60
nvme0n1          2769.50  2682.00  29592.00  10723.25  241.00    0.00    8.01    0.00    0.11    0.02    0.01    10.68     4.00    0.14  77.60
sdb                0.00  2731.00     0.00  27814.00    0.00   241.00    0.00    8.11    0.00   12.20   30.13     0.00    10.18    0.29  79.40
sdc                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
sde                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
sdf                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
sdd                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
md0              2717.50  2679.00  10870.00  10716.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     4.00     4.00    0.00   0.00

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.51    0.00    2.42     0.00    0.00   97.07

Device            r/s     w/s    rkB/s    wkB/s  rrqm/s  wrqm/s  %rrqm  %wrqm  r_await  w_await  aqu-sz  rareq-sz  wareq-sz  svctm  %util
sda                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
nvme0n1          2739.00  2747.50  27336.00  10988.50  210.00    0.00    7.12    0.00    0.12    0.02    0.00     9.98     4.00    0.14  77.20
sdb                0.00  2797.50     0.00  28270.00    0.00   210.00    0.00    6.98    0.00   11.75   29.38     0.00    10.11    0.28  78.80
sdc                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
sde                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
sdf                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
sdd                0.00    0.00     0.00     0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     0.00     0.00    0.00   0.00
md0              2688.00  2746.50  10752.00  10986.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00     4.00     4.00    0.00   0.00
```

Listing 7: top Output

```
top - 19:44:01 up 15 min,  3 users,  load average: 1.08, 0.68, 0.42
Tasks: 145 total,   1 running, 144 sleeping,   0 stopped,   0 zombie
%Cpu0  :  0.7 us,  1.4 sy,  0.0 ni, 97.3 id,  0.0 wa,  0.0 hi,  0.7 si,  0.0 st
%Cpu1  :  0.3 us,  3.1 sy,  0.0 ni, 95.9 id,  0.0 wa,  0.0 hi,  0.7 si,  0.0 st
%Cpu2  :  0.3 us,  1.7 sy,  0.0 ni, 97.6 id,  0.0 wa,  0.0 hi,  0.3 si,  0.0 st
%Cpu3  :  0.0 us,  1.3 sy,  0.0 ni, 98.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  7951.2 total,  6269.5 free,   210.9 used,  1470.8 buff/cache
MiB Swap:  3934.0 total,  3934.0 free,     0.0 used.  7079.6 avail Mem

   PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
  3294 root        20   0  748016  4760   988 S  13.3   0.1   0:01.19  fio
  3155 root        20   0        0      0      0 D   2.3   0.0   0:14.55  md0_resync
    18 root        20   0        0      0      0 S   0.3   0.0   0:00.05  ksoftirqd/1
  3152 root        20   0        0      0      0 S   0.3   0.0   0:05.38  md0_raid1
  3284 petros     20   0   9496   4080  3356 R   0.3   0.1   0:00.04  top
  3286 root        20   0 813548 428684 424916 S   0.3   5.3   0:00.37  fio
```

work freeing up reclaimable memory. If the system is in a situation in which both free and available memory decreases, it means the system has less memory to reclaim, so when an application asks for more memory, it will fault on the page allocation, likely forcing the application to exit early. This condition is typically accompanied by an entry in `dmesg` or `syslog` notifying the system administrator or user that a page allocation has occurred. If you find yourself in a situation in which the system is struggling to find memory resources to serve

applications and I/O requests, you might have to figure out the greatest offender(s) and attempt to find a resolution to the problem. In a worst case scenario, an application may contain a memory leak and not properly free unused memory, consuming more system memory that cannot be reclaimed. This scenario will need to be addressed in the application’s code. A simple way to observe the greatest consumers of your memory resources is with the `ps` utility (Listing 9). The fourth column (%MEM) is the column on which you should focus. In this example, the `fio` utility is consuming 5.2% of the system memory, and as soon as it exits, it will free that memory back into the larger pool of available memory for future use. Other things worth considering are tuning the kernel’s virtual memory subsystem with the `sysctl` utility. A guide of what parameters can be tuned are found in the *Documentation/admin-guide/sysctl/vm.txt* file of the Linux kernel source. Tunables include filesystem buffering and caching thresholds, memory swapping, and others.

Is Other Traffic Hitting the Drive?

Now is the time to dive a little deeper by looking at the traffic hitting your drive. Are you seeing unexpected I/O requests? The `blktrace` and `blkparse` applications are appropriately built to dive into this space. Start by creating a small shell script to generate “outside” traffic. The script will rely on the `sg_inq` binary provided by the `sg3_utils` package mentioned earlier:

```
#!/bin/sh

while [ 1 ]; do
    sg_inq /dev/sdf
    sleep 2
done

To execute the script, enter:

$ sh send_sg_inq.sh
```

If you were to use `blktrace` to capture the traffic hitting the drive (Listing 10),

you would see `sg_inq` or a SCSI IN-QUIRY request being sent to the drive. This operation is considered a read (outside of the general data path) because it is requesting information back from the drive. Therefore, the seventh column of the lines following those ending in `sg_inq` are labeled with an R. What will this look like when you send more I/O? You can use the `dd` utility to send 1MB of all-zero data to the drive:

`$ sudo dd if=/dev/zero of=/dev/sdf1 bs=1M`

The `blktrace` utility will dump the captured output (Listing 11); the seventh column is saying that a write (W) was sent, and the eighth column shows the size of the transfer (1342768 + 8). In one line, you see that a kernel worker thread is tasked with sending the write, and in the following line you see the logical block addressing (LBA) location rela-

Listing 8: Checking Memory Constraints

```
01 $ free -m
02          total        used        free      shared  buff/cache   available
03 Mem:      7951         201        7037           1         712        7493
04 Swap:      4095           0        4095
```

Listing 9: ps Output

```
$ ps aux | head -1; ps aux | sort -rnk 4 | head -9
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root      5088  5.0  5.2 815852 429928 pts/2    Sl+   15:35   0:00 fio --bs=1M --ioengine=libaio
--iodepth=32 --size=10g --direct=0 --runtime=60 --filename=/dev/sdf --rw=randread
--numjobs=4 --name=test
root      5097  2.2  0.4 783092 38248 ?        Ds    15:35   0:00 fio --bs=1M --ioengine=libaio
--iodepth=32 --size=10g --direct=0 --runtime=60 --filename=/dev/sdf --rw=randread
--numjobs=4 --name=test
root      5096  2.1  0.4 783088 38168 ?        Ds    15:35   0:00 fio --bs=1M --ioengine=libaio
--iodepth=32 --size=10g --direct=0 --runtime=60 --filename=/dev/sdf --rw=randread
--numjobs=4 --name=test
root      5095  2.0  0.4 783084 38204 ?        Ds    15:35   0:00 fio --bs=1M --ioengine=libaio
--iodepth=32 --size=10g --direct=0 --runtime=60 --filename=/dev/sdf --rw=randread
--numjobs=4 --name=test
root      5094  2.1  0.4 783080 38236 ?        Ds    15:35   0:00 fio --bs=1M --ioengine=libaio
--iodepth=32 --size=10g --direct=0 --runtime=60 --filename=/dev/sdf --rw=randread
--numjobs=4 --name=test
root      1421  0.2  0.3 1295556 29472 ?        Ssl   14:51   0:05 /usr/lib/snapd/snapd
root       990  0.0  0.2 107888 16912 ?        Ssl   14:50   0:00 /usr/bin/python3 /usr/share/
unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
root       844  0.0  0.2 280452 18256 ?        Ssl   14:50   0:00 /sbin/multipathd -d -s
systemd+   912  0.0  0.1 24092 10612 ?        Ss    14:50   0:00 /lib/systemd/systemd-resolved
```

Listing 10: Using blktrace to Capture Traffic

```
$ sudo blktrace -d /dev/sdf -o -|blkparse -i -
8,80  1      1      0.000000000 4693 D  R 36 [sg_inq]
8,80  1      2      0.000590071    0 C  R [0]
8,80  1      3      0.000624346 4693 D  R 144 [sg_inq]
8,80  1      4      0.001093812    0 C  R [0]
8,80  1      5      0.001121290 4693 D  R 252 [sg_inq]
8,80  1      6      0.001577685    0 C  R [0]
8,80  1      7      0.001593793 4693 D  R 252 [sg_inq]
8,80  1      8      0.002063534    0 C  R [0]
8,80  0      1      2.004478167 4695 D  R 36 [sg_inq]
8,80  0      2      2.005054640    0 C  R [0]
8,80  0      3      2.005078936 4695 D  R 144 [sg_inq]
```

Listing 11: blktrace Output

8,80	1	4398	0.027331017	218	M	W	1342768	+ 8	[kworker/u16:3]
8,80	1	4399	0.027332069	218	A	W	1342776	+ 8 <- (8,81)	1326392
8,80	1	4400	0.027332354	218	Q	W	1342776	+ 8	[kworker/u16:3]
8,80	1	4401	0.027332675	218	M	W	1342776	+ 8	[kworker/u16:3]
8,80	1	4402	0.027333775	218	A	W	1342784	+ 8 <- (8,81)	1326400
8,80	1	4403	0.027334066	218	Q	W	1342784	+ 8	[kworker/u16:3]
8,80	1	4404	0.027334387	218	M	W	1342784	+ 8	[kworker/u16:3]
8,80	1	4405	0.027335587	218	A	W	1342792	+ 8 <- (8,81)	1326408
8,80	1	4406	0.027335872	218	Q	W	1342792	+ 8	[kworker/u16:3]
8,80	1	4407	0.027336186	218	M	W	1342792	+ 8	[kworker/u16:3]
8,80	1	4408	0.027337256	218	A	W	1342800	+ 8 <- (8,81)	1326416
8,80	1	4409	0.027337542	218	Q	W	1342800	+ 8	[kworker/u16:3]
8,80	1	4410	0.027337900	218	M	W	1342800	+ 8	[kworker/u16:3]
8,80	1	4411	0.027339749	218	A	W	1342808	+ 8 <- (8,81)	1326424

tive to that drive or drive partition’s starting point (e.g., 1326392) This output can and will get large and ugly, and it might take some time to parse through the content manually. Be sure not to let the utility capture for too long. Similar to blktrace, you are able to view traffic hitting the SAS drives with the SCSI logging level function of the Linux kernel. The simplest way to observe and manage the kernel’s logging level settings is with the `scsi_logging_level` tool. For instance, to list the current logging level setting, invoke the command with the `-g` option:

```
$ sudo scsi_logging_level -g
Current scsi logging level:
/proc/sys/dev/scsi/logging_level = 0
SCSI_LOG_ERROR=0
SCSI_LOG_TIMEOUT=0
SCSI_LOG_SCAN=0
SCSI_LOG_MLQUEUE=0
SCSI_LOG_MLCOMPLETE=0
SCSI_LOG_LLQUEUE=0
SCSI_LOG_LLCOMPLETE=0
SCSI_LOG_HLQUEUE=0
SCSI_LOG_HLCOMPLETE=0
SCSI_LOG_IOCTL=0
```

The Linux SCSI subsystem is separated into three main levels: the upper layer comprises entry point drivers that allow you to connect with the underlying device (e.g., `sd`, `sg`, `sr`), and the mid layer connects the upper level to the interface driver of the lower level (i.e., Fibre Channel, SAS, iSCSI, etc.). Aside from connecting both upper and lower levels, the mid level is responsible for

failure handling and error correction of failed requests propagated up from the underlying device or the host bus adaptor (HBA) in the low level connecting to that underlying device (Figure 1). To enable the highest level of logging for all SCSI errors and traffic in the low and mid layers, type:

```
$ sudo scsi_logging_level -s -E 7 -L 7 -M 7
New scsi logging level:
/proc/sys/dev/scsi/logging_level = 2096647
SCSI_LOG_ERROR=7
SCSI_LOG_TIMEOUT=0
SCSI_LOG_SCAN=0
SCSI_LOG_MLQUEUE=7
SCSI_LOG_MLCOMPLETE=7
SCSI_LOG_LLQUEUE=7
SCSI_LOG_LLCOMPLETE=7
SCSI_LOG_HLQUEUE=0
SCSI_LOG_HLCOMPLETE=0
SCSI_LOG_IOCTL=0
```

When logging is enabled, it will print SCSI level traces in both `dmesg`

and `syslog` or the kernel log. For instance, in the example shown in Listing 12, you can see the SCSI Command Descriptor Blocks (CDBs) and traces printed for write, cache synchronization, and inquiry commands. Much like blktrace, this log can get pretty large and pretty ugly and should be enabled for brief periods at a time to avoid overwhelming the system. Another thing worth noting is that, with capture tools such as these, latencies are introduced and may alter the original I/O profile being chased, so proceed with caution. To disable all logging, type:

```
$ sudo scsi_logging_level -s -E 0 -L 0 -M 0
New scsi logging level:
/proc/sys/dev/scsi/logging_level = 7
SCSI_LOG_ERROR=7
SCSI_LOG_TIMEOUT=0
SCSI_LOG_SCAN=0
SCSI_LOG_MLQUEUE=0
SCSI_LOG_MLCOMPLETE=0
SCSI_LOG_LLQUEUE=0
SCSI_LOG_LLCOMPLETE=0
SCSI_LOG_HLQUEUE=0
SCSI_LOG_HLCOMPLETE=0
SCSI_LOG_IOCTL=0
```

Perf Tracing

Another tool captures system traces, but at the Linux kernel level and in the context of CPU-executed operations. That tool is called `perf` and can be obtained from the Linux Kernel Archives [1]. Some Linux distributions will offer it in their downloadable package repositories, but it might be

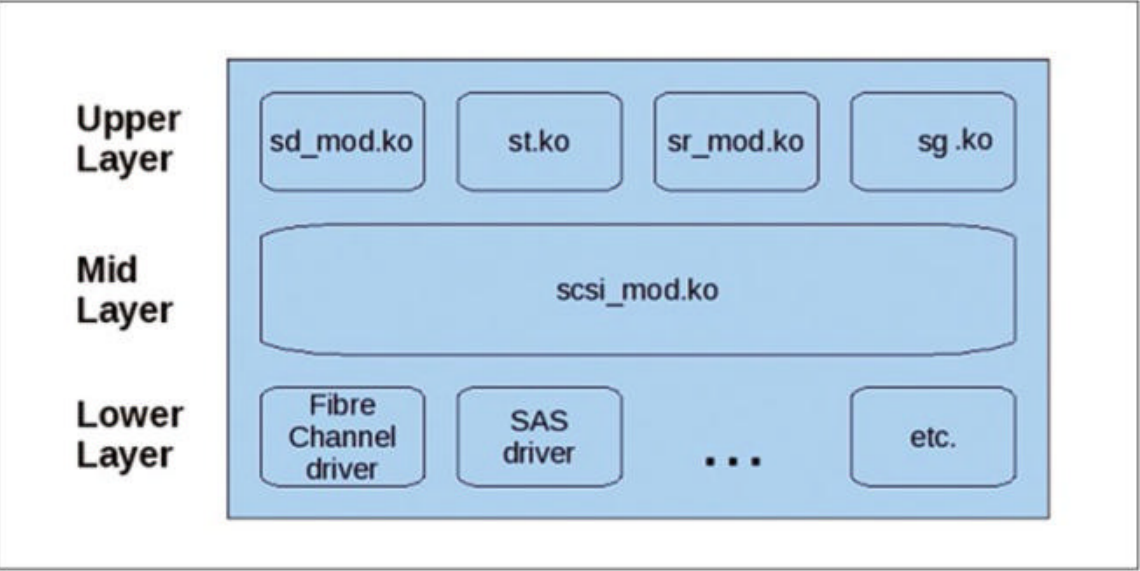


Figure 1: The general outline of the Linux SCSI subsystem.

best to download the latest and greatest from the website ([Listing 13](#)). Extract the file into a local working directory

```
$ tar xJf perf-5.9.0.tar.xz
and change into the perf source code
subdirectory:
```

```
$ cd perf-5.9.0/tools/perf/
```

Resolve any and all dependencies and run `make` and `make install` utilities. Now that `perf` is installed, you can generate I/O to a disk drive with `fio`,

```
$ sudo fio --bs=1K --ioengine=libaio --
--iodepth=32 --size=10g --direct=0 --
--runtime=60 --filename=/dev/sdf --
--rw=randread --numjobs=1 --name=test
```

and in another terminal window, invoke `perf` to record kernel operations executed at the CPU level for 30 seconds:

```
$ sudo ./perf record -a -g -- sleep 30
[ perf record: 2
Woken up 8 times to write data ]
[ perf record: Captured and wrote 2
2.476 MB perf.data (12373 samples) ]
```

A `perf.data` file is generated in the local working directory. You can use tools to open this file directly for analysis or to generate other output from that file. For instance, to generate a report summarizing the 30 seconds, enter:

```
$ sudo ./perf script > out.script
```

```
$ sudo ./perf report > out.report
```

You will need both files just generated:

```
$ ls out.*
out.report out.script
```

If you open the report file (i.e., `out.report`), you will see a summary of all the kernel operations and the amount of CPU it utilized during that 30-second snapshot, and you will see the subroutines tied to the top-level functions ([Listing 14](#)).

The `perf` utility is useful when you might need to see where the CPU spends most of its time and provides insight into what could be holding up the completion of a function or shed

Listing 12: Kernel Log

```
Mar 27 15:21:24 dev-machine kernel: [ 387.117129] sd 0:0:0:0: [sda] tag#3 Send: scmd 0x000000001842ae5b
Mar 27 15:21:24 dev-machine kernel: [ 387.117134] sd 0:0:0:0: [sda] tag#3 CDB: Write(10) 2a 00 1d 04 5d 88 00 00 20 00
Mar 27 15:21:24 dev-machine kernel: [ 387.117468] sd 0:0:0:0: [sda] tag#3 Done: SUCCESS Result: hostbyte=DID_OK driverbyte=DRIVER_OK
Mar 27 15:21:24 dev-machine kernel: [ 387.117485] sd 0:0:0:0: [sda] tag#3 CDB: Write(10) 2a 00 1d 04 5d 88 00 00 20 00
Mar 27 15:21:24 dev-machine kernel: [ 387.117491] sd 0:0:0:0: [sda] tag#3 scsi host busy 1 failed 0
Mar 27 15:21:24 dev-machine kernel: [ 387.117495] sd 0:0:0:0: Notifying upper driver of completion (result 0)
Mar 27 15:21:24 dev-machine kernel: [ 387.117585] sd 0:0:0:0: [sda] tag#24 Send: scmd 0x00000000fa2f2e1a
Mar 27 15:21:24 dev-machine kernel: [ 387.117590] sd 0:0:0:0: [sda] tag#24 CDB: Synchronize Cache(10) 35 00 00 00 00 00 00 00 00
Mar 27 15:21:24 dev-machine kernel: [ 387.130198] sd 0:0:0:0: [sda] tag#24 Done: SUCCESS Result: hostbyte=DID_OK driverbyte=DRIVER_OK
Mar 27 15:21:24 dev-machine kernel: [ 387.130202] sd 0:0:0:0: [sda] tag#24 CDB: Synchronize Cache(10) 35 00 00 00 00 00 00 00 00
Mar 27 15:21:24 dev-machine kernel: [ 387.130205] sd 0:0:0:0: [sda] tag#24 scsi host busy 1 failed 0
Mar 27 15:21:24 dev-machine kernel: [ 387.130207] sd 0:0:0:0: Notifying upper driver of completion (result 0)

[ ... ]

Mar 27 15:21:25 dev-machine kernel: [ 387.655119] sd 2:0:1:1: [sdf] tag#3669 Send: scmd 0x00000000b06d61c4
Mar 27 15:21:25 dev-machine kernel: [ 387.655122] sd 2:0:1:1: [sdf] tag#3669 CDB: Inquiry 12 00 00 00 90 00
Mar 27 15:21:25 dev-machine kernel: [ 387.655587] sd 2:0:1:1: [sdf] tag#3669 Done: SUCCESS Result: hostbyte=DID_OK driverbyte=DRIVER_OK
Mar 27 15:21:25 dev-machine kernel: [ 387.655589] sd 2:0:1:1: [sdf] tag#3669 CDB: Inquiry 12 00 00 00 90 00
Mar 27 15:21:25 dev-machine kernel: [ 387.655591] sd 2:0:1:1: [sdf] tag#3669 scsi host busy 1 failed 0
Mar 27 15:21:25 dev-machine kernel: [ 387.655593] sd 2:0:1:1: Notifying upper driver of completion (result 0)
```

Listing 13: Get perf from kernel.org

```
$ wget https://mirrors.edge.kernel.org/pub/linux/kernel/tools/perf/v5.9.0/perf-5.9.0.tar.xz
--2021-03-21 15:40:30-- https://mirrors.edge.kernel.org/pub/linux/kernel/tools/perf/v5.9.0/perf-5.9.0.tar.xz
Resolving mirrors.edge.kernel.org (mirrors.edge.kernel.org)... 2604:1380:1:3600::1, 147.75.197.195
Connecting to mirrors.edge.kernel.org (mirrors.edge.kernel.org)|2604:1380:1:3600::1|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1857268 (1.8M) [application/x-xz]
Saving to: 'perf-5.9.0.tar.xz'

perf-5.9.0.tar.xz          100%[=====>]
1.77M  8.85MB/s   in 0.2s

2021-03-21 15:40:30 (8.85 MB/s) - 'perf-5.9.0.tar.xz' saved [1857268/1857268]
```

Listing 14: Summary of Kernel Operations

```
# To display the perf.data header info, please use --header/--header-only options.
#
#
# Total Lost Samples: 0
#
# Samples: 12K of event 'cycles'
# Event count (approx.): 1354753461
#
# Children      Self  Command      Shared Object      Symbol
# .....
#
# 57.94%    0.00%  swapper      [kernel.kallsyms]  [k] secondary_startup_64
#      |
#      |--secondary_startup_64
#      |
#      |--48.28%--start_secondary
#      |
#      |      |--48.25%--cpu_startup_entry
#      |
#      |      |--47.68%--do_idle
#      |
#      |      |--36.89%--call_cpuidle
#      |
#      |      |--36.81%--cpuidle_enter
#      |
#      |      |--35.93%--cpuidle_enter_state
#      |
#      |      |--16.95%--intel_idle
#      |
#      |      |--10.25%--ret_from_intr
#      |
#      |      |--10.21%--do_IRQ
#      |
#      |      |--7.62%--irq_exit
#      |
#      |      |--7.47%--__softirqentry_text_start
#      |
#      |      |--7.17%--blk_done_softirq
#
# [ ... ]
#
# 0.51%    0.04%  fio          [kernel.kallsyms]  [k] mpage_alloc.isra.0
# 0.50%    0.00%  fio          libc-2.31.so       [.] 0x00007f0576ab611a
#      |
#      |--0x7f0576ab611a
#
# 0.50%    0.24%  swapper      [kernel.kallsyms]  [k] sched_clock_cpu
# 0.49%    0.00%  perf         perf               [.] 0x00005570000237de
# 0.48%    0.18%  swapper      [kernel.kallsyms]  [k] ttwu_do_wakeup
# 0.46%    0.19%  swapper      [kernel.kallsyms]  [k] set_next_entity
# 0.46%    0.00%  rcu_sched    [kernel.kallsyms]  [k] ret_from_fork
# 0.46%    0.00%  rcu_sched    [kernel.kallsyms]  [k] kthread
# 0.46%    0.10%  swapper      [kernel.kallsyms]  [k] load_balance
#
# [ ... ]
```

Listing 15: Generating a Flame Graph

```
$ git clone https://github.com/brendangregg/FlameGraph.git
Cloning into 'FlameGraph'...
remote: Enumerating objects: 1067, done.
remote: Total 1067 (delta 0), reused 0 (delta 0), pack-reused 1067
Receiving objects: 100% (1067/1067), 1.87 MiB | 6.67 MiB/s, done.
Resolving deltas: 100% (612/612), done.

$ ./FlameGraph/stackcollapse-perf.pl out.script > out.folded
$ ./FlameGraph/flamegraph.pl out.folded > out.svg
```

light on poorly developed functions in the kernel space.

Sometimes it can be difficult to parse this data, which is why you can convert it into a visual chart called a flame graph [2]. To do this, clone the git repo and convert the original output script file into an SVG (Listing 15).

When you open the file, you will see a visual representation of the 30-sec-

drastically once the SSD reaches its *write cliff*, or the point at which the drive exhausts its first writes to each NAND memory cell. Once you reach the drive’s write cliff, you are now exercising a programmable erase (PE) cycle for each write operation. This multistep process involves erasing an entire cell before writing to it. Each NAND cell has a finite number of PE cycles. The PE cycles to a cell will vary on the type of NAND technology employed. Vendors resort to something called over-provisioning to prolong or delay this write cliff. Often you will get double the NAND memory when purchasing a drive. For instance, on a 3TB PCIe SSD, you might be paying for 6TB of storage, of which you are only able to use 3TB. There are two methods by which

someone can restore an SSD to its former performing glory. One is by employing a technique called TRIM (called Discard on SAS technologies). In this background process, the software tells the drive to prep unused data blocks for a future write operation. Tools such as `fstrim` will enable this function, but many filesystems (e.g., ext4) and volume managers (e.g., logical volume management, LVM) offer *discard* support.

Conclusion

When something does not seem quite right, you have many ways to peek into a system – a few of these mechanisms were explored here. This guide does not cover everything and just provides a starting point, but it is

enough of a starting point to get you moving in the right direction of isolating and diagnosing the root cause of your performance troubles.

Info
[1] Linux Kernel Archives: <http://kernel.org/>
[2] Flame graph: <https://github.com/brendangregg/FlameGraph>

The Author
Petros Koutoupis is currently a senior performance software engineer at Cray (now HPE) for its Lustre High Performance File System division. He is also the creator and maintainer of the RapidDisk Project (<http://www.rapiddisk.org/>). Koutoupis has worked in the data storage industry for well over a decade and has helped pioneer the many technologies unleashed in the wild today.

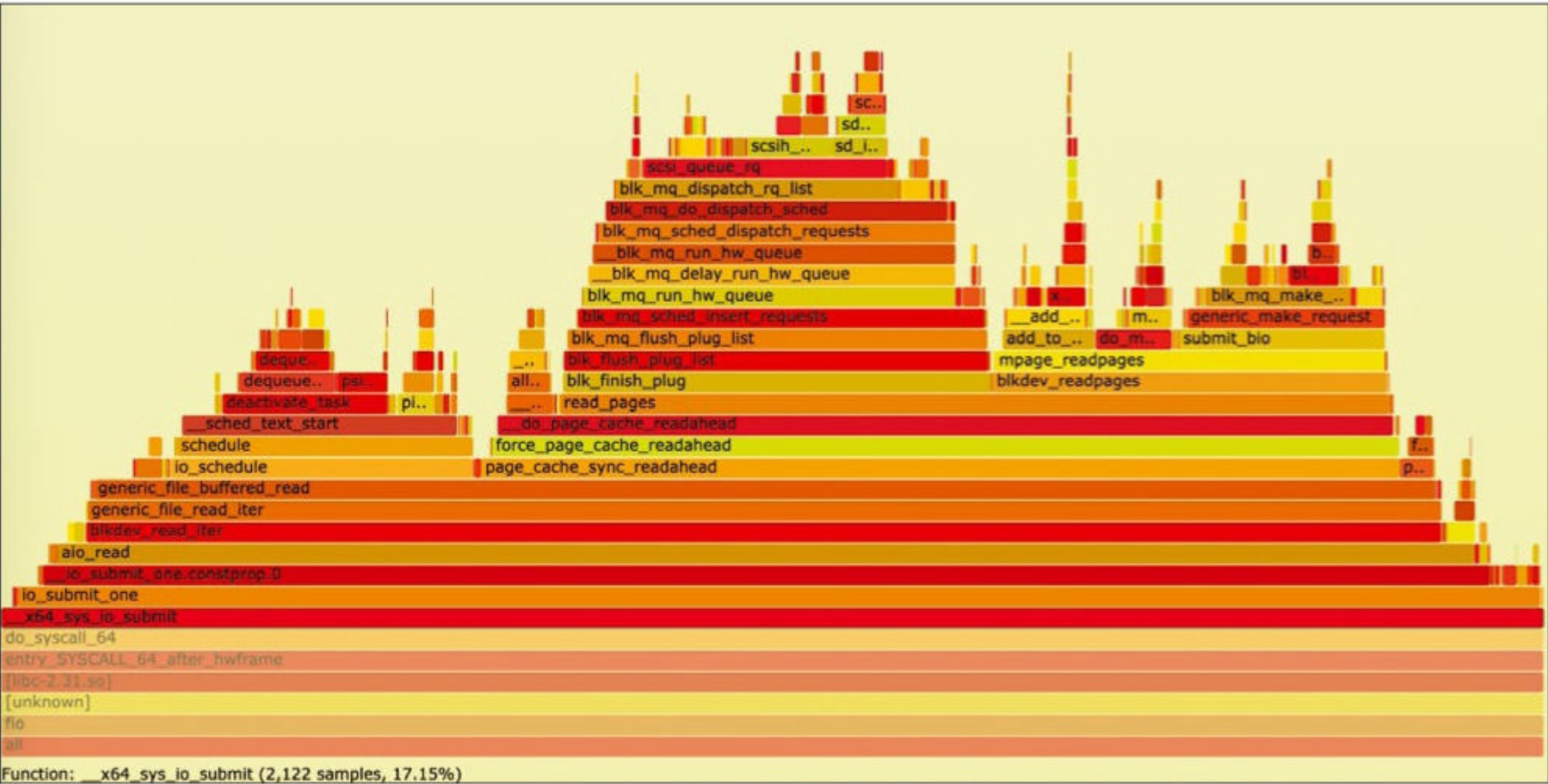


Figure 4: Drilling deeper into the flame graph and subfunctions.



Next-generation terminal UI tools

Cursed Monitor

The implementation of terminal user interface libraries enables an impressive variety of in-terminal graphics. By Federico Lucifredi

The venerable **curses** [1] library first enabled plotting graphics on a Unix terminal, allowing the emergence of dynamic, interactive tools such as **top** [2] and its brood of descendants to improve greatly the ability to monitor a system's state continuously. The transition from static tables of numbers to charts and sometimes even dynamic data representations was followed by new implementations of terminal user interface (TUI) libraries, including **Ncurses** [3] and **Newt** [4]. Jeff Layton has previously graced these pages with a tutorial on how TUI tools are developed today [5], and yours truly followed in the next issue with a survey of network monitoring tools adopting in-terminal graphics [6]. Two years later, I revisit the subject to witness the impressive improvement of the state of the art taking place since.

Not Your Father's Curses

The **bashtop** [7] utility is one of the most remarkable new entrants – so much so it has been featured by the likes of *Forbes* magazine [8], hardly what I would call conventional treatment for a Linux utility. Originally

written in Bash (hence its name), it was re-implemented in Python [9] last year, reducing its CPU consumption by two-thirds and simultaneously improving speed and extending functionality. A third rewrite, this time in C++, is in progress at the time of this writing. The performance enhancements are welcome for administrators

who leave the tool running in the background, but all versions of the tool are equally suitable for intermittent use – just keep in mind that newer ports provide additional functionality not found in the original; for example, mouse support absent in the Bash version has been added with the Python port.



Figure 1: Bashtop makes good use of terminals wider than 80 columns, dynamically adjusting its layout.



Figure 2: The 1980s coin-op arcade vibe of Bashtop's configuration screen.



Figure 3: Detail of the CPU ribbon display, detailing aggregate and per-core histories.

Bashtop (Figure 1) supports both Linux and macOS (when used with iTerm2 [10]) and is controlled from the keyboard. (Menu entries are keyed by letters highlighted in grey.) The Python port is available as part of the Ubuntu standard repositories [11] starting with Ubuntu 20.10 (Groovy Gorilla), whereas the original Bash implementation is easily built on stock 20.04 (Focal Fossa) from the author's repository with `sudo make install`.

In its default configuration, Bashtop provides a ranked process listing including memory and CPU usage, as well as system-wide aggregate status for CPU, memory, permanent storage, and network connections. The configuration screen features a retro arcade game look, speaking to the author's attention to detail (Figure 2).

A more practical nod to UX is the fading out of process listing entries as you get farther from the top, and the increased brightness of entries as they approach 100 percent. Although the tool can be used to monitor systems with many disks or network connections, the most striking visualization is reserved for CPU load history (Figure 3). It presents a history ribbon

view of the CPU, breaks out individual cores, and includes load average [12] figures. Network interfaces benefit from a similar visualization (Figure 4), with the Python implementation bpytop extend-

ing the treatment to memory use, as well (Figure 5).

At a Glance

The older `glances` [13] tool might not push the limits of TUI graphics as aggressively as its newer challenger, but it more than makes it up in portability and extensibility. Supporting Linux, Solaris, *BSD, macOS, and even Windows, `glances` is my go-to system monitoring: I quite simply install it everywhere (Figure 6). Take note to use binaries where a build is available, because like many other highly portable applications, a build of `Glances` can take quite some time to complete on account of its many dependencies. On macOS, I use `brew install glances` as a stress generator for new hosts, akin to running a kernel build on Linux. `Glances` can also provide remote monitoring when needed (both as a web application and in client-server mode), although I choose SSH when the occasion calls for remote access.

Alongside processes listing and network, disk, and CPU system met-

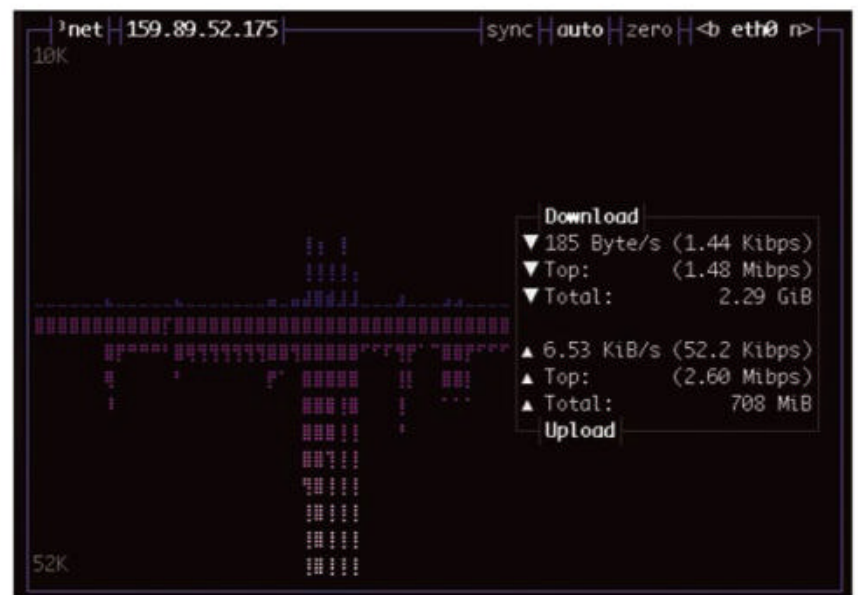


Figure 4: Network traffic history can span multiple interfaces (if present).

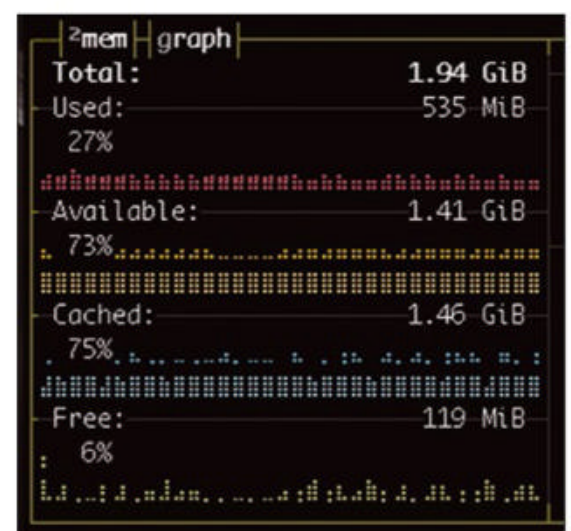


Figure 5: Memory use history is a great addition found in the Python port.

rics, a `glances` summary screen can display sensor data, notably for CPU and disk temperatures, which is critical information for servers. Sensor instrumentation can include other plugins, such as battery charge levels on laptops. A unique capability of `glances` is its built-in container monitoring, putting Docker containers on equal footing with Linux processes, making it very interesting in



Figure 6: `Glances` employs a simple but effective bar graph load visualization style.



Figure 7: Glances can integrate with Docker's API to include a container view.

Shop the Shop  shop.linuxnewmedia.com

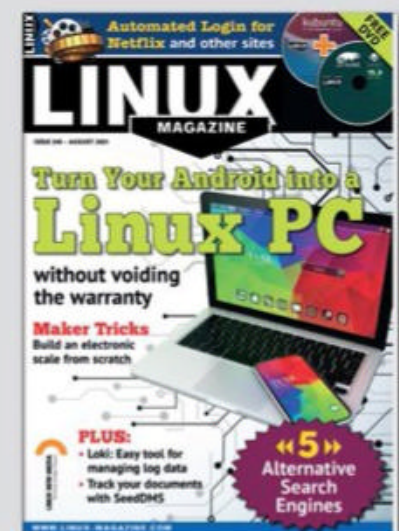
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

Want to subscribe?

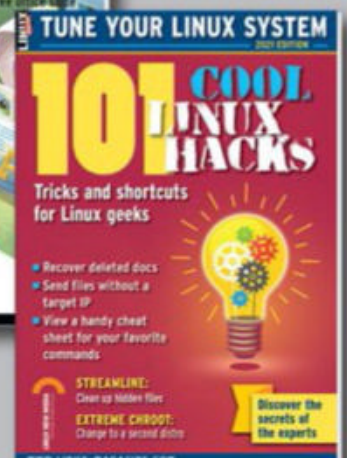
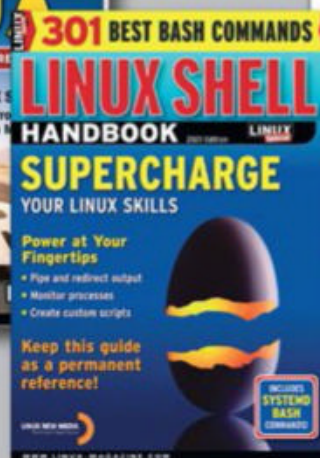
Searching for that back issue you really wish you'd picked up at the newsstand?

➤ shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



modern environments with containerized applications (Figure 7). Glances itself is available as a Docker image with the one-liner

```
docker pull nicolargo/glances.
```

Network Watch

The venerable ping [14] tool has also received a TUI facelift – two in fact. The prettytyping [15] Bash and Awk wrapper for the original tool requires no installation and no special permissions but greatly enhances the visibility of latency variance or the appearance of spurious errors while tracking aggregate statistics for the last 60 packets (Figure 8). On packet loss, prettytyping breaks to a new line and highlights latency changes with a predefined color scale. Less apt at highlighting failures, but impressive at charting change in latency, is the newfangled gping [16], which charts latency in what is perhaps the best dynamically resizing in-terminal graph to date (Figure 9). The gping tool pushes the limits of in-terminal graphs, and it is perfectly suited to examining continuous changes in network performance, as opposed to highlighting spurious events. ■

Info

- [1] Curses: [\[https://en.wikipedia.org/wiki/Curses_\(programming_library\)\]](https://en.wikipedia.org/wiki/Curses_(programming_library))
- [2] top(1) man page: [\[https://manpages.ubuntu.com/manpages/focal/en/man1/top.1.html\]](https://manpages.ubuntu.com/manpages/focal/en/man1/top.1.html)
- [3] Ncurses: [\[https://en.wikipedia.org/wiki/Ncurses\]](https://en.wikipedia.org/wiki/Ncurses)
- [4] Newt: [\[https://en.wikipedia.org/wiki/Newt_\(programming_library\)\]](https://en.wikipedia.org/wiki/Newt_(programming_library))

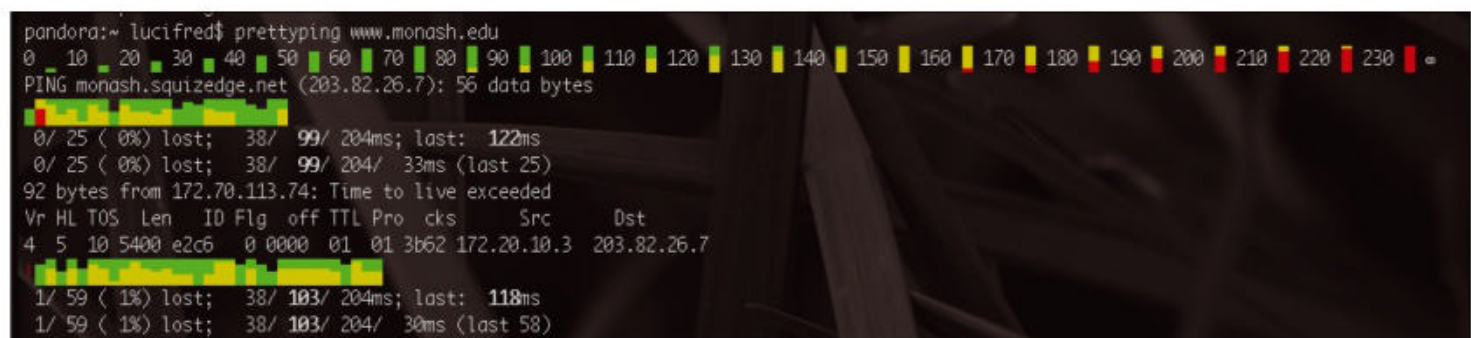


Figure 8: prettytyping colorizes ping output to highlight packet loss and latency spikes.

- [5] “TUI Tools” by Jeff Layton, *ADMIN*, issue 50, 2019, [\[https://www.admin-magazine.com/Archive/2019/50/A-smoke-jumping-admin-s-best-friend\]](https://www.admin-magazine.com/Archive/2019/50/A-smoke-jumping-admin-s-best-friend)
- [6] “Network Performance In-Terminal Graphics Tools” by Federico Lucifredi, *ADMIN*, issue 51, 2019, [\[https://www.admin-magazine.com/Archive/2019/51/Network-performance-in-terminal-graphics-tools\]](https://www.admin-magazine.com/Archive/2019/51/Network-performance-in-terminal-graphics-tools)
- [7] Bashtop: [\[https://github.com/aristocratos/bashtop\]](https://github.com/aristocratos/bashtop)
- [8] “You Need This Beautiful Linux and MacOS App in Your Terminal” by Jason Evangelho, *Forbes*, Aug 2020, [\[https://www.forbes.com/sites/jasonevangelho/2020/08/05/meet-the-most-beautiful-linux-app-you-need-in-your-terminal/\]](https://www.forbes.com/sites/jasonevangelho/2020/08/05/meet-the-most-beautiful-linux-app-you-need-in-your-terminal/)
- [9] Bpytop: [\[https://github.com/aristocratos/bpytop\]](https://github.com/aristocratos/bpytop)
- [10] iTerm2 for macOS: [\[https://iterm2.com/\]](https://iterm2.com/)
- [11] bashtop(1) man page: [\[https://manpages.ubuntu.com/manpages/groovy/en/man1/bashtop.1.html\]](https://manpages.ubuntu.com/manpages/groovy/en/man1/bashtop.1.html)

- [12] “The Iconic Load Average Metric” by Federico Lucifredi, *ADMIN*, issue 11, 2012
- [13] Glances: [\[https://github.com/nicolargo/glances\]](https://github.com/nicolargo/glances)
- [14] ping(1) man page: [\[https://manpages.ubuntu.com/manpages/focal/en/man1/ping.1.html\]](https://manpages.ubuntu.com/manpages/focal/en/man1/ping.1.html)
- [15] prettytyping, Denilson Sá Maia blog: [\[http://denilson.sa.nom.br/prettytyping/\]](http://denilson.sa.nom.br/prettytyping/)
- [16] gping: [\[https://github.com/orf/gping\]](https://github.com/orf/gping)

Author

Federico Lucifredi (@0xf2) is the Product Management Director for Ceph Storage at Red Hat, 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*.

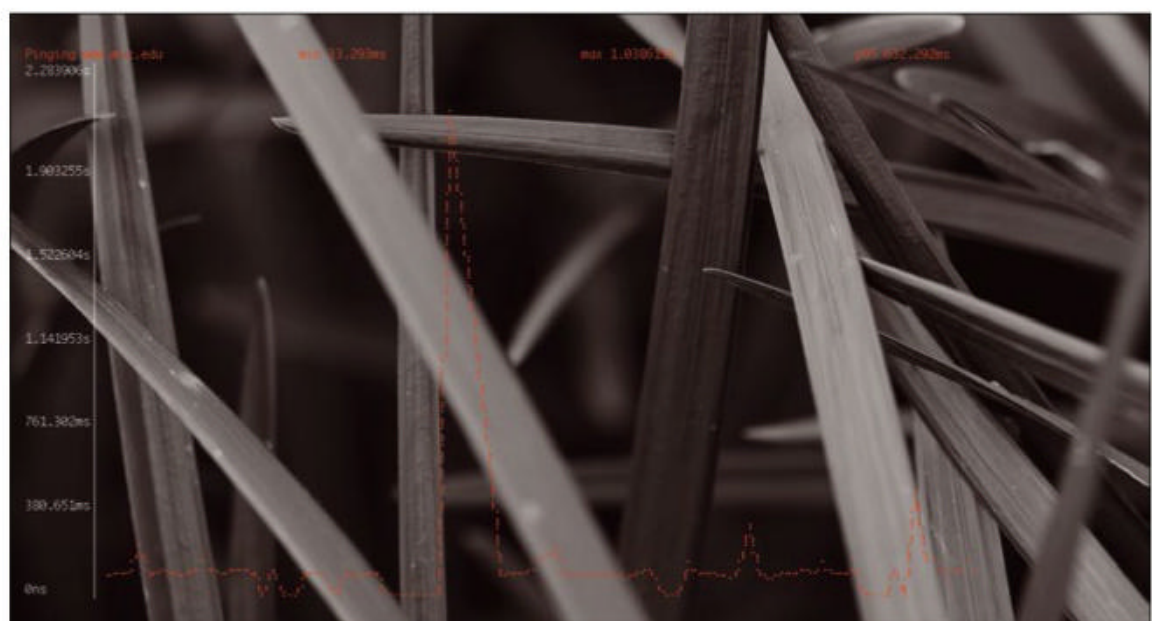


Figure 9: gping charting the performance of a network connection.

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.

#63/May/June 2021

Automation

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

On the DVD: Ubuntu 21.04 Server



#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

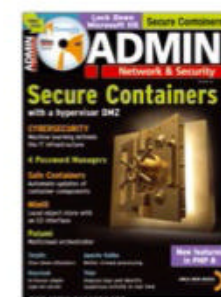


#61/January/February 2021

Secure Containers

Security is the watchword this issue, and we begin with eliminating container security concerns.

On the DVD: Clonezilla Live 2.7.0

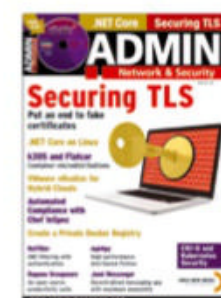


#60/November/December 2020

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

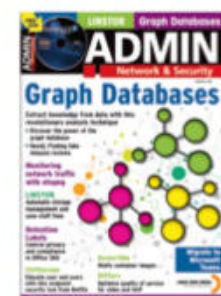


#58/July/August 2020

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)



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	
Erik Bärwaldt	26
Chris Binnie	48, 52, 70
Markus Feilner	36
Florian Frommherz	64
Tam Hanna	58
Ken Hess	3
Thomas Joos	44
Petros Koutoupis	74, 82
Jeff Layton	78
Martin Loschwitz	20, 36
Federico Lucifredi	93
Sebastian Meyer	16
Bernd Müller	10
Steffen Schmalstieg	10
Christian Schneemann	16
Jack Wallen	8
Matthias Wübbeling	56

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, Aubrey Vaughn

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen, Illustration based on graphics by Yongyut Rukkachatsuwan, 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. July/August 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.



Sharpen your view!

TUXEDO InfinityBook Pro 14



Intel Core i7-1165G7
Intel Iris Xe Graphics



3K Omnia Display
16:10 | 2880 x 1800 Pixels



**Deep Grey
magnesium chassis**
1,5 cm thin | 1 kg



Thunderbolt 4
Full featured USB-C 4.0



100%
Linux

5

Year
Warranty



Lifetime
Support



Built in
Germany



German
Privacy



Local
Support

TUXEDO
COMPUTERS

 tuxedocomputers.com

A Linux Distribution For Professionals

openSUSE Leap 15.3

Leap is a desirable distribution for IT

professionals, entrepreneurs,

hobbyists, small businesses

and educational practitioners



get.opensuse.org