GWDG

Security Workshop

science + computing

Holger Gantikow

Security in HPC with Containers

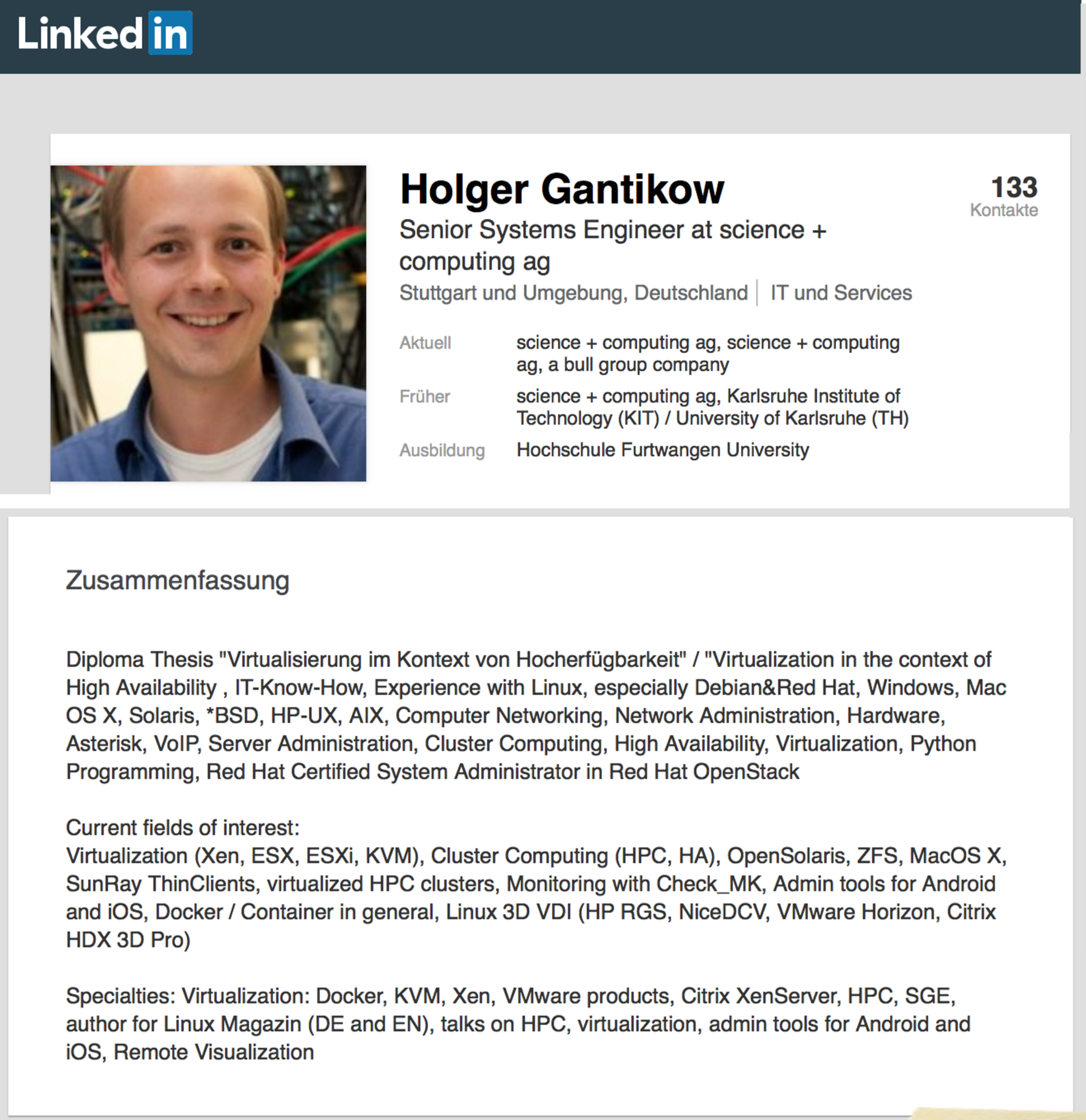
Online, December 2021

Trusted partner for your Digital Journey

Atos

0

whoami



Holger Gantikow

Senior Systems Engineer at science + computing ag

Stuttgart und Umgebung, Deutschland | IT und Services

133 Kontakte

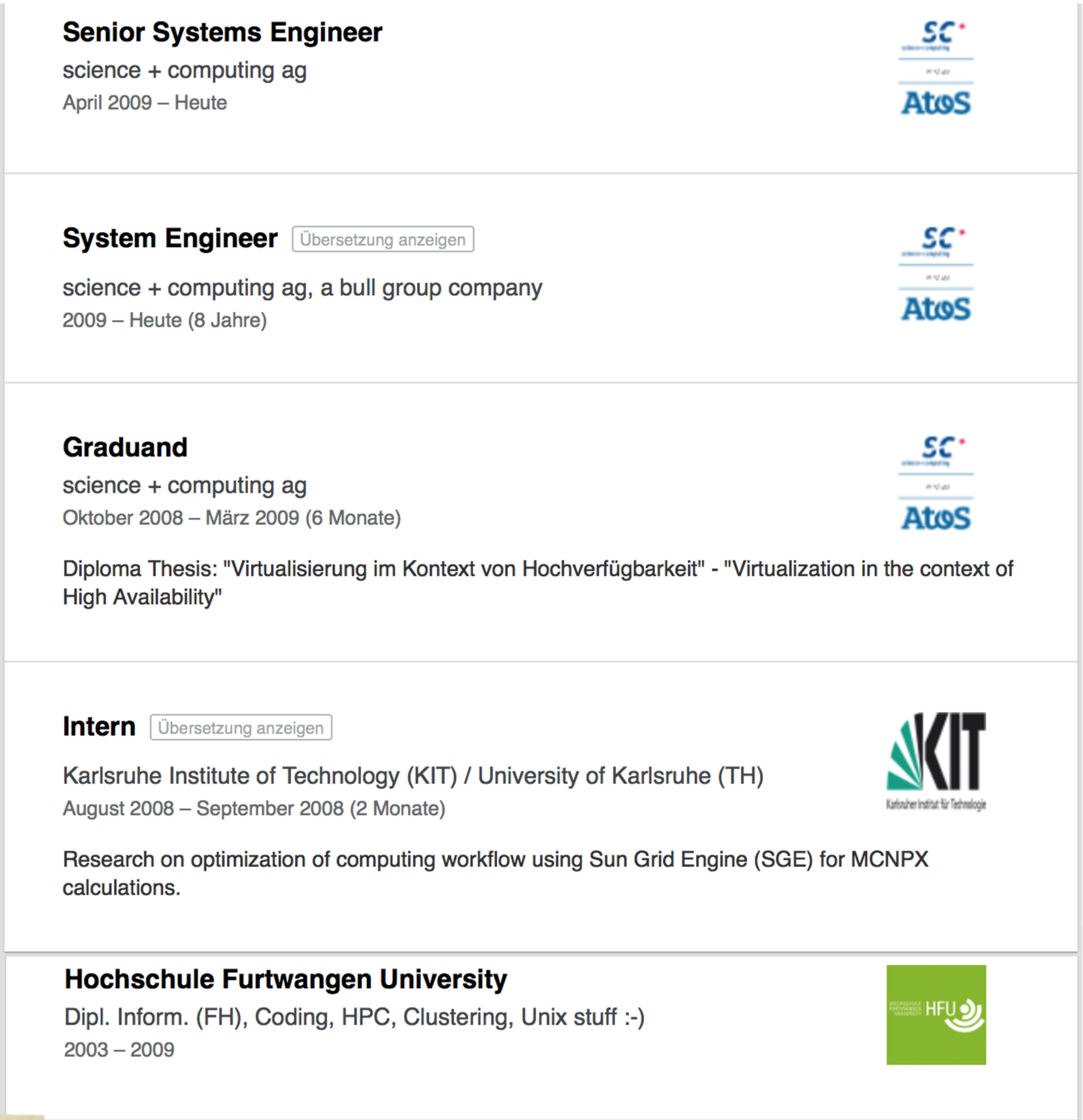
Aktuell	science + computing ag, science + computing ag, a bull group company
Früher	science + computing ag, Karlsruhe Institute of Technology (KIT) / University of Karlsruhe (TH)
Ausbildung	Hochschule Furtwangen University

Zusammenfassung

Diploma Thesis "Virtualisierung im Kontext von Hocherfügbarkeit" / "Virtualization in the context of High Availability , IT-Know-How, Experience with Linux, especially Debian&Red Hat, Windows, Mac OS X, Solaris, *BSD, HP-UX, AIX, Computer Networking, Network Administration, Hardware, Asterisk, VoIP, Server Administration, Cluster Computing, High Availability, Virtualization, Python Programming, Red Hat Certified System Administrator in Red Hat OpenStack

Current fields of interest:
Virtualization (Xen, ESX, ESXi, KVM), Cluster Computing (HPC, HA), OpenSolaris, ZFS, MacOS X, SunRay ThinClients, virtualized HPC clusters, Monitoring with Check_MK, Admin tools for Android and iOS, Docker / Container in general, Linux 3D VDI (HP RGS, NiceDCV, VMware Horizon, Citrix HDX 3D Pro)

Specialties: Virtualization: Docker, KVM, Xen, VMware products, Citrix XenServer, HPC, SGE, author for Linux Magazin (DE and EN), talks on HPC, virtualization, admin tools for Android and iOS, Remote Visualization



Senior Systems Engineer
science + computing ag
April 2009 – Heute

System Engineer Übersetzung anzeigen
science + computing ag, a bull group company
2009 – Heute (8 Jahre)

Graduand
science + computing ag
Oktober 2008 – März 2009 (6 Monate)

Diploma Thesis: "Virtualisierung im Kontext von Hochverfügbarkeit" - "Virtualization in the context of High Availability"

Intern Übersetzung anzeigen
Karlsruhe Institute of Technology (KIT) / University of Karlsruhe (TH)
August 2008 – September 2008 (2 Monate)

Research on optimization of computing workflow using Sun Grid Engine (SGE) for MCNPX calculations.

Hochschule Furtwangen University
Dipl. Inform. (FH), Coding, HPC, Clustering, Unix stuff :-)
2003 – 2009

Find me on Linkedin & Xing – feel free to reach out!

science + computing - Quick Facts

Focus on technical & scientific computing with 30 years of expertise

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Founded in 1989

science + computing

Dedicated unit for high-end & business critical IT Services

FY2019

€ 41,5M External Revenue

Atos

Accompanied by Atos / Bull Advanced Computing solutions

Tübingen

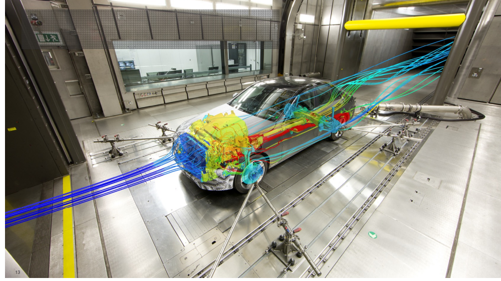
Berlin

Munich

Düsseldorf

Timișoara

Brașov



Agenda

01. Trends related to HPC Security

02. Containers support these trends

03. Software Bill of Material

04. Summary & Conclusion

01. Trends related to HPC Security

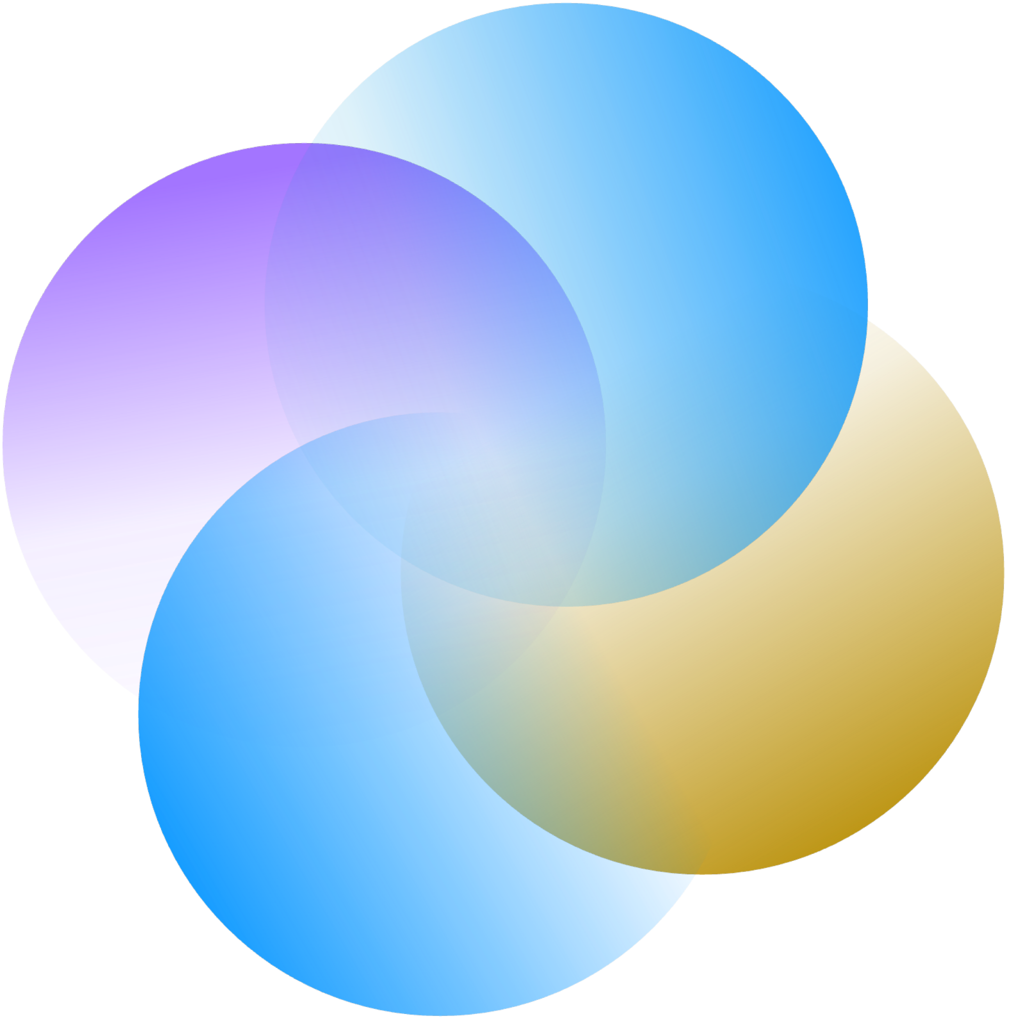
Developments in HPC Security

Securing access is often not enough

- Access to HPC resources usually already well secured
  - VPN, SSH keys, 2FA, only login nodes exposed, …
- Users have great liberty especially in R&D HPC Environments
  - ISV codes, admin installed applications, user supplied code (`~/bin`)
  - HPC != regular Enterprise IT Environment (FOSDEM 2017 ;))
  - Trust in users still a key element
- In Enterprise HPC environments move towards
  - Zero trust
  - Multi-tenancy (environments opening up to external partners, "competition")
  - Supporting future workloads (AI/ML, Data Analytics, …) – all on one big cluster?
  - Multi-site (including cloud)

This implies necessary changes in the way things are done – containers can help here

02. Containers support these trends



Why researchers love containers

Quick recap

Mobility / Portability

- Versatile resources
  - Laptop
  - Workstation
  - HPC
  - Cloud
- Encapsulated SW environment

User-provided applications

- Dependency conflicts
- “Works on my machine”
- Legacy Environments
  - Fortran @CentOS5
- Scientific Collaboration
- Unprivileged build

Reproducibility

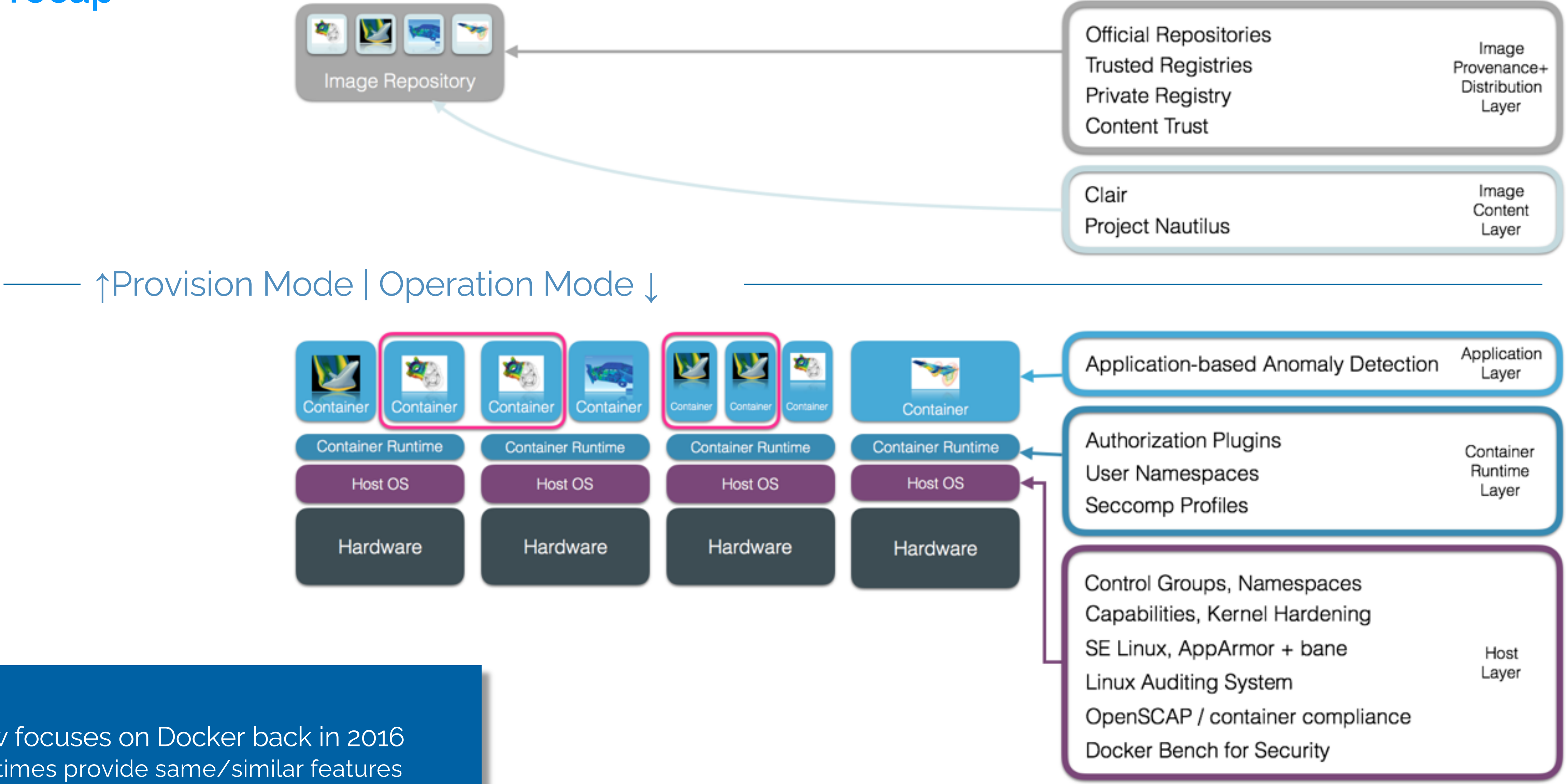
- Scientific Collaboration
- Sharing of SW environment with data alongside publication

Performance

- Minimal overhead
- Close to bare metal
- Backed by many studies

Security Features added over time

Quick recap



Official Repositories
Trusted Registries
Private Registry
Content Trust

Image Provenance+ Distribution Layer

Clair
Project Nautilus

Image Content Layer

Application-based Anomaly Detection

Application Layer

Authorization Plugins
User Namespaces
Seccomp Profiles

Container Runtime Layer

Control Groups, Namespaces
Capabilities, Kernel Hardening
SE Linux, AppArmor + bane
Linux Auditing System
OpenSCAP / container compliance
Docker Bench for Security

Host Layer

+ Rootless Containers
+ Security Monitoring

Notes

* Overview focuses on Docker back in 2016
* Other runtimes provide same/similar features

Source: Gantikow et al., VHPC16, Providing Security in Container-based HPC Runtime Environments

Key aspects in Container Security

From Docker Shocker to Rootless Containers

Container technology has matured over time

- Containers were dreaded in the beginning
  - Docker Shocker, access to the `docker` CLI == possiblilty for privilege escalation

- Lots of **security features added over time**
  - **Beyond namespaces and cgroups** (isolated operations + resource usage limits)
  - **Seccomp**: "Sandboxing" by limiting the System Calls a container can use
  - **Security Monitoring** at runtime: Sysdig, Falco; alerting if a container misbehaves (according to policies)

Nowadays

- Typical container runtimes **do not grant more privileges** than the calling user has directly on the system
- In addition (if workload allows) possibility to **restrict** access to the host system and other workloads
- Provide possibility to **rethink the HPC system software stack**
  - Allen, Benjamin S. et al. "Modernizing the HPC System Software Stack." *(2020)* - *https://arxiv.org/abs/2007.10290*
  - "Containerize all the things" tempting in many cases…

03. Software Bill of Material (SBOM)

Software Bill of Material

Aka "What is running on my cluster?"

Hard to keep track of software used on a large-scale system

- Lots of different applications, with numberless dependencies
- Especially hard when SW is provided beyond `rpm/apt/apk` (pip, jars, go modules, …)

Hard to answer questions like

- What software is outdated / has vulnerabilities?
- What software relies on a specific buggy library version that impacts the results?

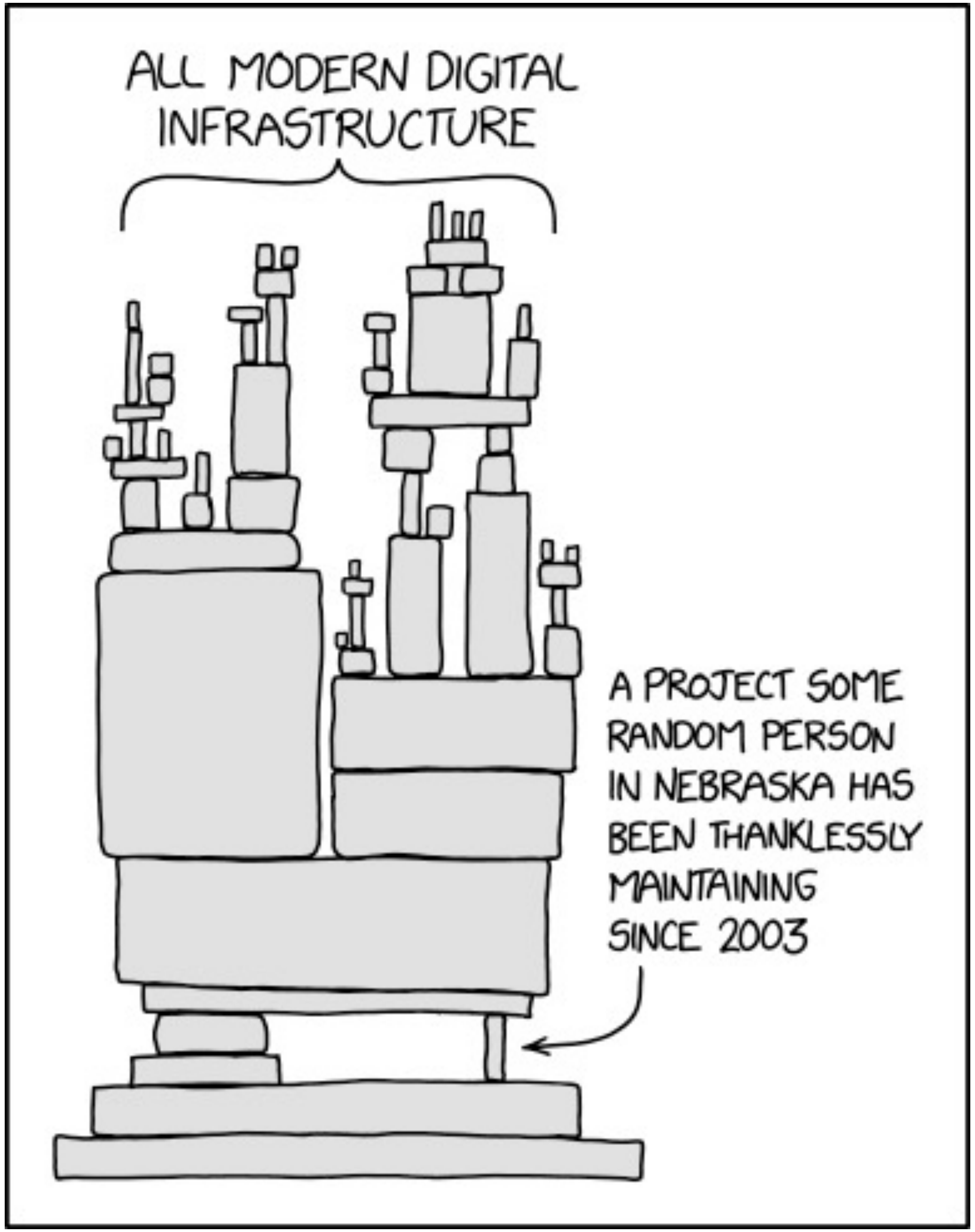
Gets much easier when relying on containers as sole source of software in an environment

- Software used = Host Software + Container image content

OSS software solutions to support this (examples later)

- Various package formats / SW sources, details like Maintainers, Licences, Checksums of files, …
- Should be integrated with image release process / registry ("Container App Store")

By example: Log4j


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Source: https://xkcd.com/2347/

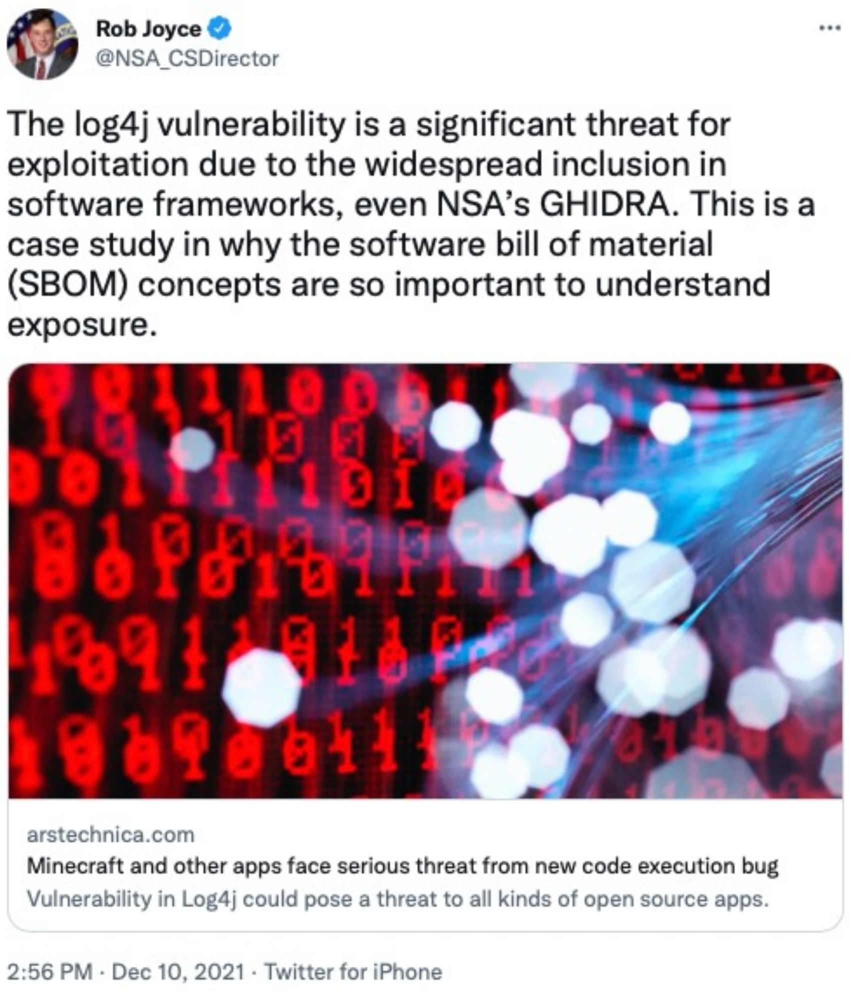
By example: Log4j

YOUR NEXT TASK IS TO FIGURE OUT
WHICH OF YOUR APPLICATIONS USE LOG4J
imgflip.com

Source: https://imgflip.com/i/5y1mv8

SBOM

Software Bill of Material



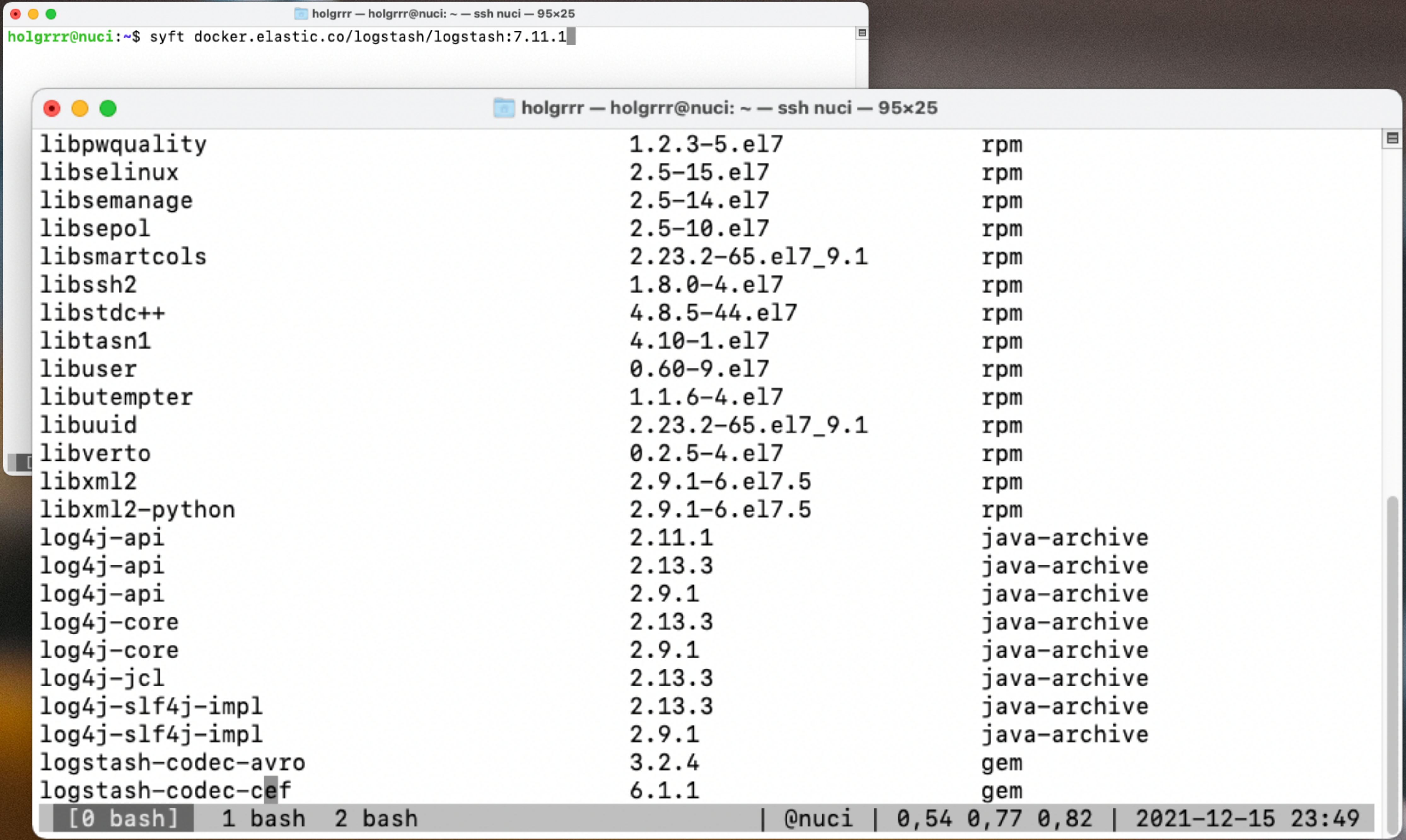
Rob Joyce
@NSA_CSDirector

The log4j vulnerability is a significant threat for exploitation due to the widespread inclusion in software frameworks, even NSA's GHIDRA. This is a case study in why the software bill of material (SBOM) concepts are so important to understand exposure.

arstechnica.com
Minecraft and other apps face serious threat from new code execution bug
Vulnerability in Log4j could pose a threat to all kinds of open source apps.

2:56 PM · Dec 10, 2021 · Twitter for iPhone

Source: https://twitter.com/NSA_CSDirector/status/1469305071116636167

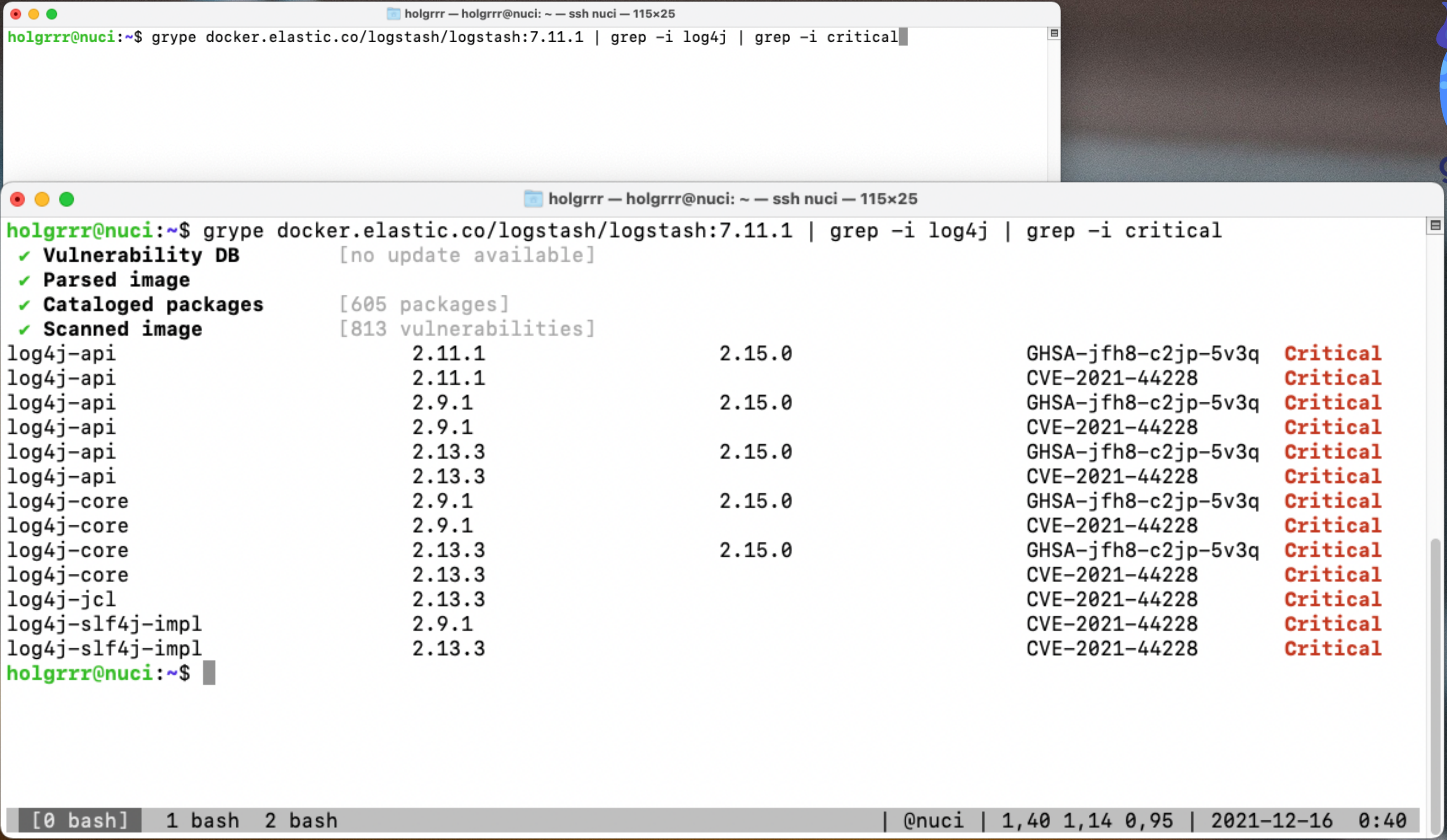

holgrrr — holgrrr@nuci: ~ — ssh nuci — 95×25
holgrrr — holgrrr@nuci: ~ — ssh nuci — 95×25

```
holgrrr@nuci:~$ syft docker.elastic.co/logstash/logstash:7.11.1
```

```
libpwquality                            1.2.3-5.el7                 rpm
libselinux                              2.5-15.el7                  rpm
libsemanage                             2.5-14.el7                  rpm
libsepol                                2.5-10.el7                  rpm
libsmartcols                            2.23.2-65.el7_9.1           rpm
libssh2                                 1.8.0-4.el7                 rpm
libstdc++                               4.8.5-44.el7                rpm
libtasn1                                4.10-1.el7                  rpm
libuser                                 0.60-9.el7                  rpm
libutempter                             1.1.6-4.el7                 rpm
libuuid                                 2.23.2-65.el7_9.1           rpm
libverto                                0.2.5-4.el7                 rpm
libxml2                                 2.9.1-6.el7.5               rpm
libxml2-python                          2.9.1-6.el7.5               rpm
log4j-api                               2.11.1                      java-archive
log4j-api                               2.13.3                      java-archive
log4j-api                               2.9.1                       java-archive
log4j-core                              2.13.3                      java-archive
log4j-core                              2.9.1                       java-archive
log4j-jcl                               2.13.3                      java-archive
log4j-slf4j-impl                        2.13.3                      java-archive
log4j-slf4j-impl                        2.9.1                       java-archive
logstash-codec-avro                     3.2.4                       gem
logstash-codec-cef                      6.1.1                       gem
 [0 bash]  1 bash  2 bash                         | @nuci | 0,54 0,77 0,82 | 2021-12-15 23:49
```

syft

Note: A deliberately old version was used. Command: `syft docker.elastic.co/logstash/logstash:7.11.1`


holgrrr — holgrrr@nuci: ~ — ssh nuci — 115×25

```
holgrrr@nuci:~$ grype docker.elastic.co/logstash/logstash:7.11.1 | grep -i log4j | grep -i critical
```

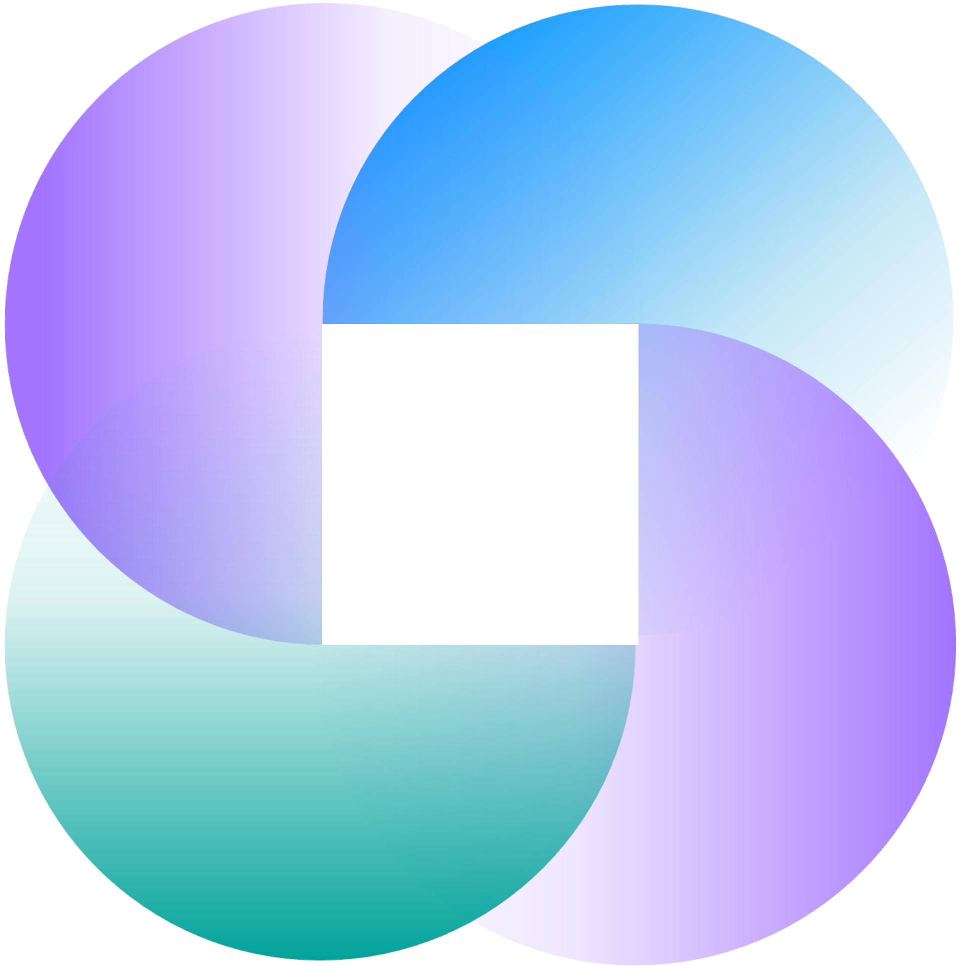
holgrrr — holgrrr@nuci: ~ — ssh nuci — 115×25

```
holgrrr@nuci:~$ grype docker.elastic.co/logstash/logstash:7.11.1 | grep -i log4j | grep -i critical
 ✔ Vulnerability DB        [no update available]
 ✔ Parsed image
 ✔ Cataloged packages      [605 packages]
 ✔ Scanned image           [813 vulnerabilities]
log4j-api                       2.11.1                   2.15.0                   GHSA-jfh8-c2jp-5v3q  Critical
log4j-api                       2.11.1                                            CVE-2021-44228       Critical
log4j-api                       2.9.1                    2.15.0                   GHSA-jfh8-c2jp-5v3q  Critical
log4j-api                       2.9.1                                             CVE-2021-44228       Critical
log4j-api                       2.13.3                   2.15.0                   GHSA-jfh8-c2jp-5v3q  Critical
log4j-api                       2.13.3                                            CVE-2021-44228       Critical
log4j-core                      2.9.1                    2.15.0                   GHSA-jfh8-c2jp-5v3q  Critical
log4j-core                      2.9.1                                             CVE-2021-44228       Critical
log4j-core                      2.13.3                   2.15.0                   GHSA-jfh8-c2jp-5v3q  Critical
log4j-core                      2.13.3                                            CVE-2021-44228       Critical
log4j-jcl                       2.13.3                                            CVE-2021-44228       Critical
log4j-slf4j-impl                2.9.1                                             CVE-2021-44228       Critical
log4j-slf4j-impl                2.13.3                                            CVE-2021-44228       Critical
holgrrr@nuci:~$


 [0 bash]  1 bash  2 bash                                           | @nuci | 1,40 1,14 0,95 | 2021-12-16  0:40
```

Note: A deliberately old version was used. Command: `grype docker.elastic.co/logstash/logstash:7.11.1 | grep -i log4j | grep -i critical`

04. Summary & Conclusion



Summary & Conclusion

Containers have come a long way

- Containers != Docker - Many options usable in HPC: Singularity, Charliecloud, Sarus, Podman, …
- High level of acceptance in HPC environments
- Good way for users to bring along their own SW environment

Containers support many trends seen in HPC environments

- Provide the possibility to rethink application deployment
  - Admin curated images + user-provided applications based on site base image, …
- Beneficial for security – if workload allows: isolation, security monitoring, …
- Will improve insights regarding software running on the system

Insights might lead to additional effort

- Especially decisions how to deal with vulnerabable code (image rebuild, …) – automate early!

Q&A

Thank you

Contact Information

Holger Gantikow
science + computing ag

holger.gantikow@atos.net