Confidential Containers in multi-tenant HPC environments

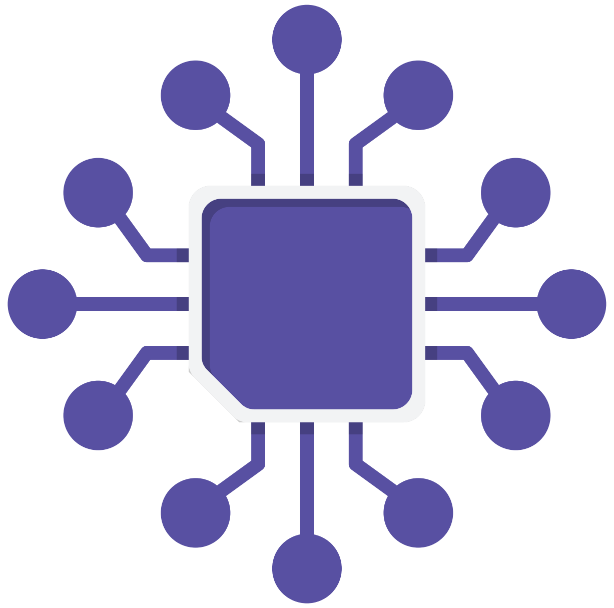
Barbara Krašovec, IJS, EGI CSIRT, SLING
Dejan Lesjak, IJS, SLING

Security and the HPC environment

- Users have direct SSH access to the HPC.
- Users on the system are trusted.
- Login nodes are protected and monitored, as they are accessible from the external network.
- Zero trust is not implemented.
- Defence-in-depth principle implementation is rare
- Clusters are heterogeneous and accept users from different disciplines, organisations, countries...

User software complexity

- multiple dependencies
- legacy software
- updates and changes on the host system have impact on the user software
- performance issues due to software upgrades
- reproducibility
- portability



Convenient solution: containers

Containers on HPC

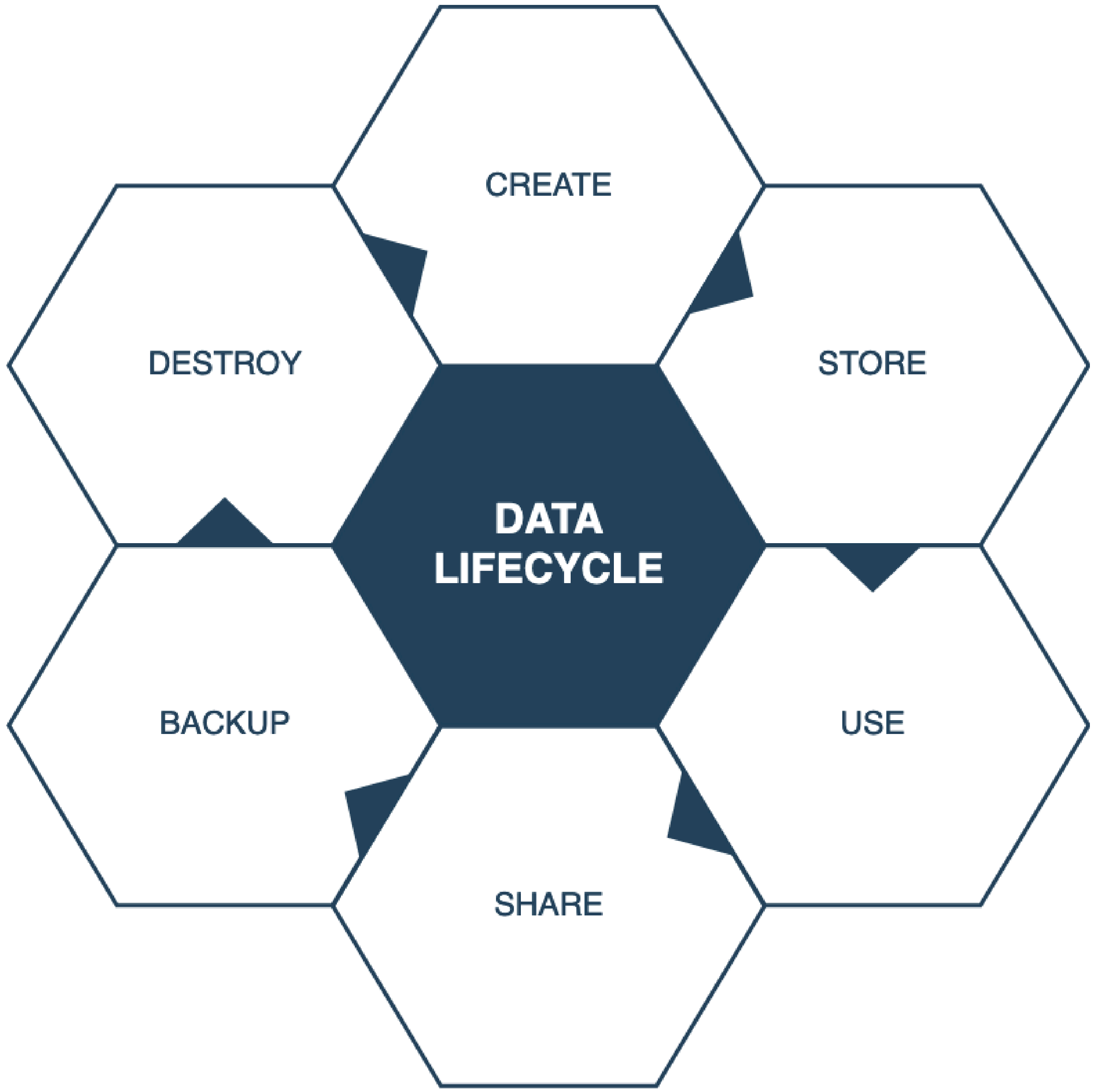
- Containers share the host kernel.
- No hardware isolation.
- User-space abstraction.
- Containers are useful for packaging software.
- Different options available: Singularity, Apptainer, enroot, CharlieCloud, Sarus etc.
- Complex MPI/interconnect compatibility.
- Isolation is ensured by using cgroups and namespaces.

Using containers solves some challenges and brings new ones…

Big data

- No data should be in the container.
- Encryption is often not possible (too much data).
- Security is limited: access controls, RBAC, and token-based access.

What about data in memory?

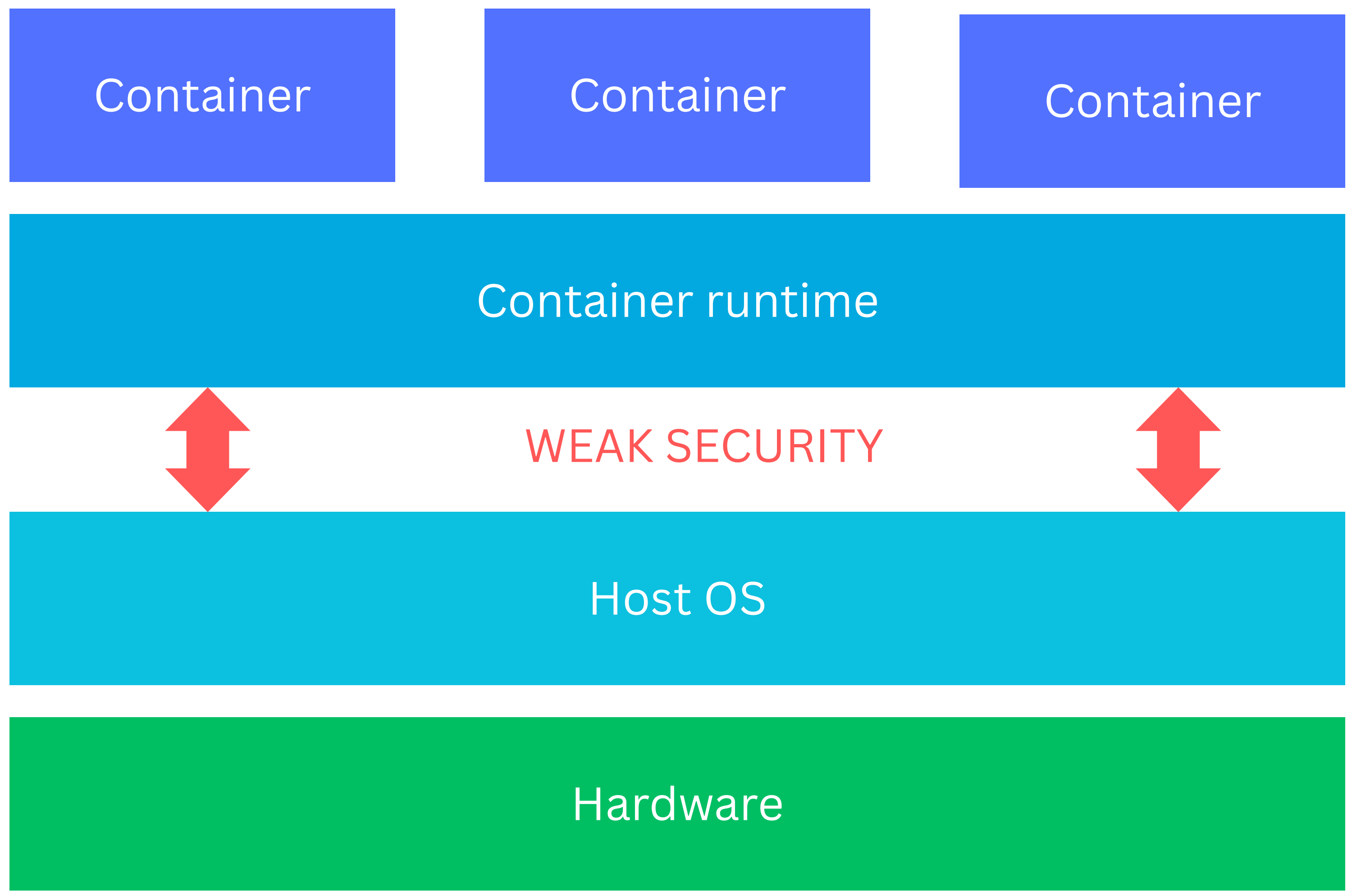


Isolation

- cgroups
- user namespaces

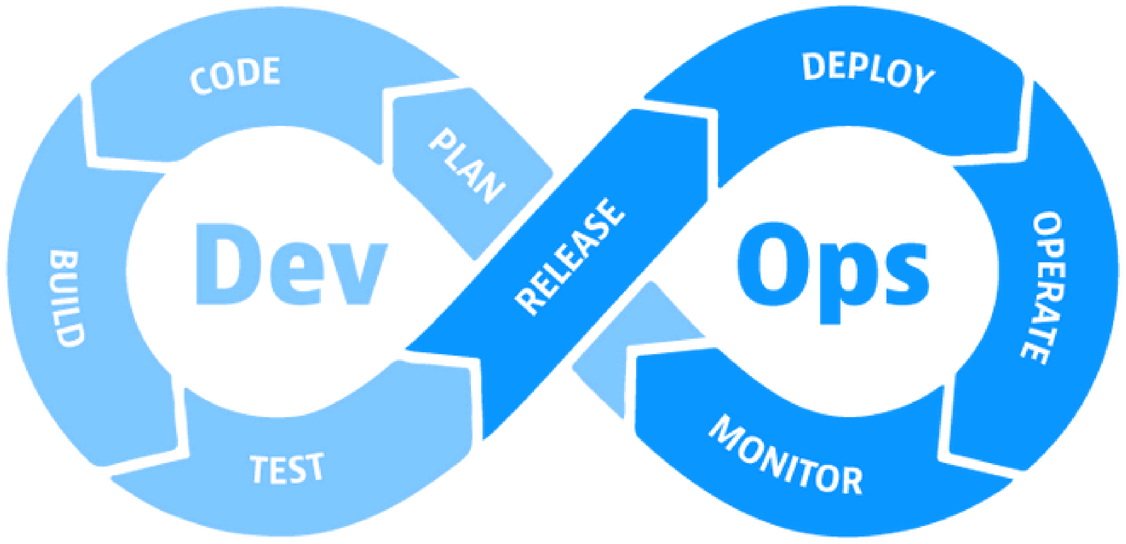
- Control resource usage.
- Dismiss noisy neighbour effect.
- Reduce attack surface.

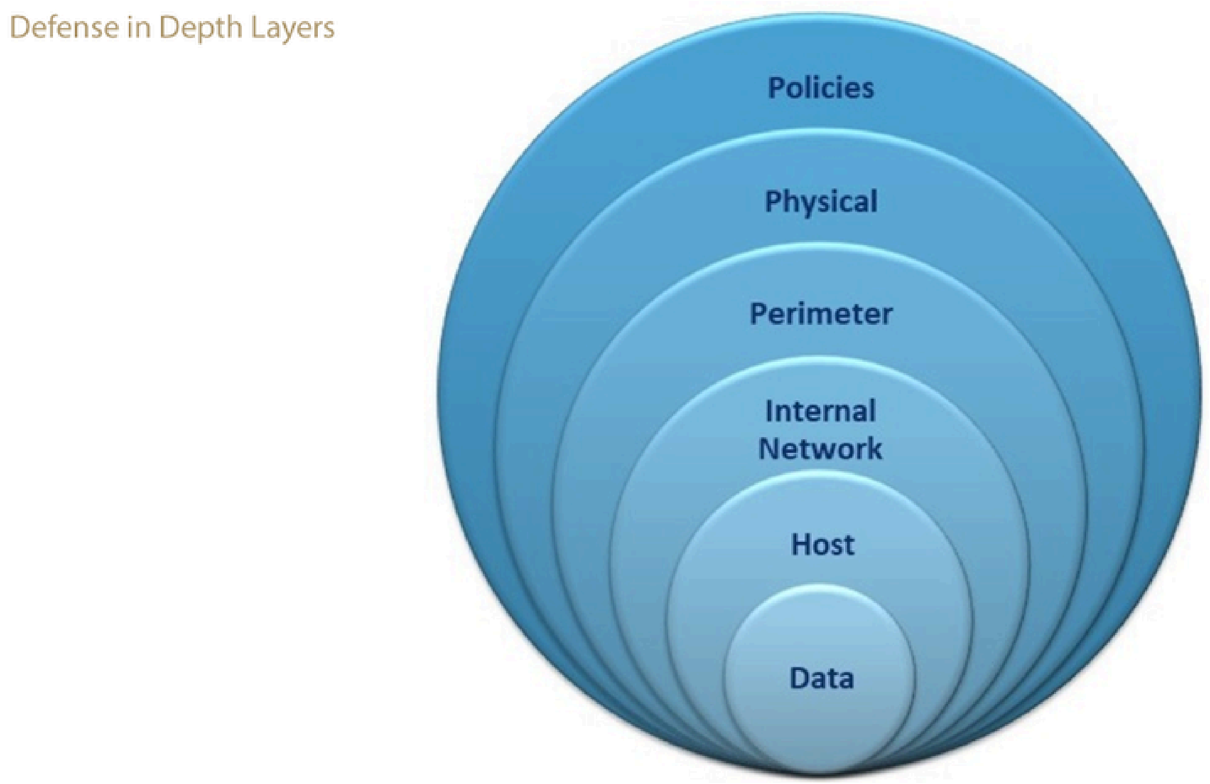
Weak boundary between the host and the container.



Security

- images
    - signatures
    - private registries
    - automated deployment
    - vulnerability management
    - encryption
- HE: AppArmor, SELinux, host hardening, seccomp profiles, Linux kernel capabilities
- per tenant networks





But what if a user has to process confidential or sensitive data?

Use verified/trusted container images

- **Sign your images** (supported in Singularity 3.0) with PGP key and verify the signature: access tokens can be used.
- Use **automatic deployment** of images and place them in a private registry.
- Perform a **secret and vulnerability scan** of the images.

```
$ singularity keys newpair
Enter your name (e.g., John Doe) : Barbara Krasovec
Enter your email address (e.g., john.doe@example.com) : barbara.krasovec@ijs.si
Enter optional comment (e.g., development keys) : Containers NSC key
Enter a passphrase :
Retype your passphrase :
Generating Entity and OpenPGP Key Pair... done
barbarak@sampo:~$ singularity key list
Public key listing (/home/barbarak/.apptainer/keys/pgp-public):

0)  User:              Barbara Krasovec (containers NSC key) <barbara.krasovec@ijs.si>
 Creation time:     2025-05-09 08:07:07 +0200 CEST
 Fingerprint:       2F1B3AA3C55D9722FF049BA0A6DA71F99B0920CD
 Length (in bits):  4096

$ singularity keys push 2F1B3AA3C55D9722FF049BA0A6DA71F99B0920CD
INFO:    Key server response: Upload successful. This is a new key, a welcome email has been sent.
public key `2F1B3AA3C55D9722FF049BA0A6DA71F99B0920CD' pushed to server successfully

$ singularity keys search barbara.krasovec@ijs.si
Showing 1 results

KEY ID    BITS  NAME/EMAIL
9B0920CD  4096  Barbara Krasovec (containers NSC key) <barbara.krasovec@ijs.si

$ singularity sign singularity-defs/mpi-mellanox.sif
INFO:    Signing image with PGP key material
Enter key passphrase :
INFO:    Signature created and applied to image 'singularity-defs/mpi-mellanox.sif'

$ singularity verify singularity-defs/mpi-mellanox.sif
INFO:    Verifying image with PGP key material
[LOCAL]   Signing entity: Barbara Krasovec (containers NSC key) <barbara.krasovec@ijs.si>
[LOCAL]   Fingerprint: 2F1B3AA3C55D9722FF049BA0A6DA71F99B0920CD
Objects verified:
ID  |GROUP   |LINK    |TYPE
------------------------------------------------
1   |1       |NONE    |Def.FILE
2   |1       |NONE    |JSON.Generic
3   |1       |NONE    |JSON.Generic
4   |1       |NONE    |FS
INFO:    Verified signature(s) from image 'singularity-defs/mpi-mellanox.sif'
```

Make use of Seccomp profiles

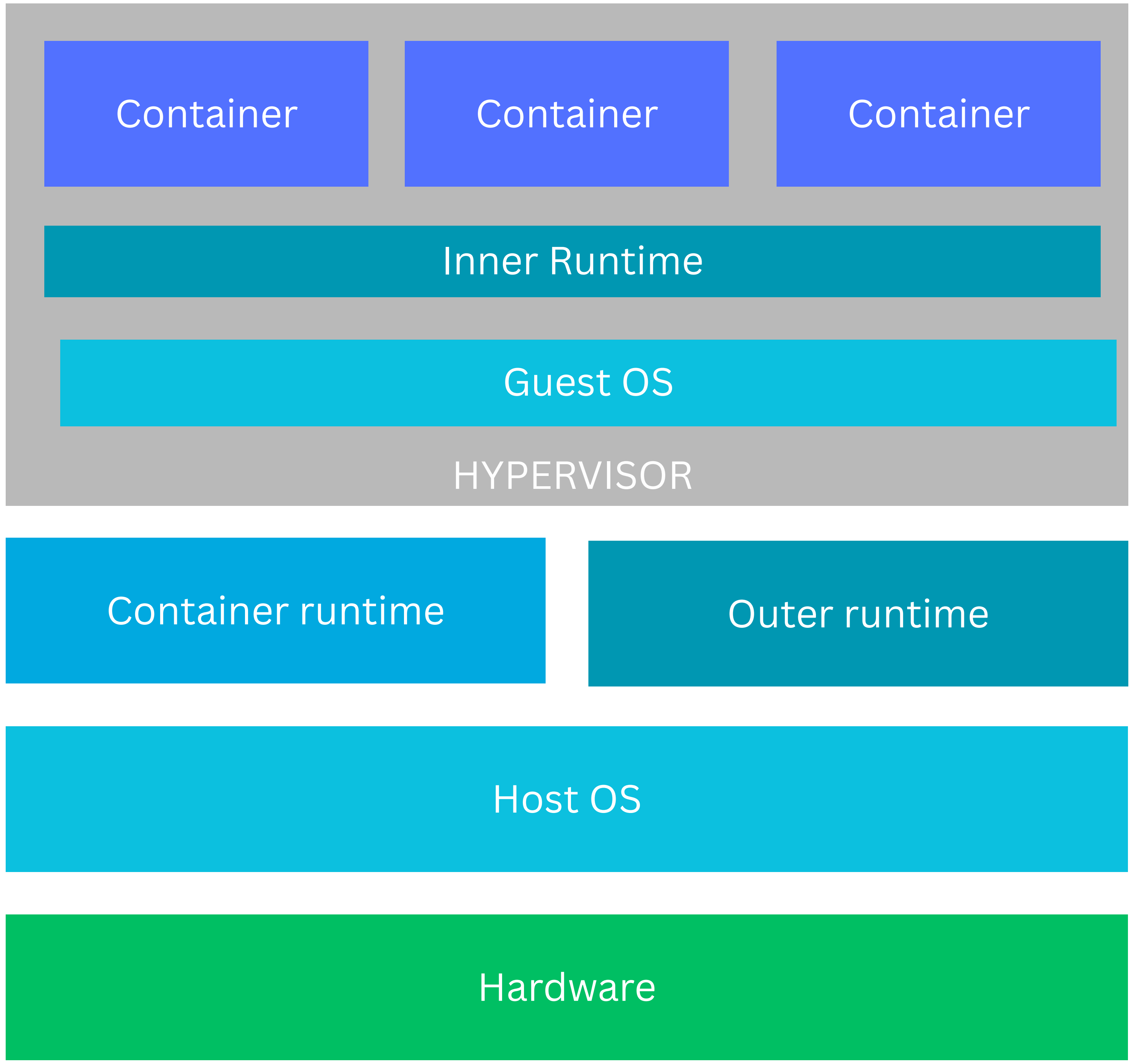
```
SECCOMP
template: https://github.com/apptainer/singularity/blob/master/etc/seccomp-profiles/
default.json

singularity exec --security seccomp:/path/to/seccomp.json my_container.sif

are seccomp profiles supported in the kernel
# grep SECCOMP /boot/config-$(uname -r)
CONFIG_HAVE_ARCH_SECCOMP=y
CONFIG_HAVE_ARCH_SECCOMP_FILTER=y
CONFIG_SECCOMP=y
CONFIG_SECCOMP_FILTER=y
# CONFIG_SECCOMP_CACHE_DEBUG is not set

strace -e trace=socket,connect,accept,bind,listen sendto recvfrom sendmsg recvmsg
ping 8.8.8.8

nsc-login1 ~# dmesg -T| grep seccomp
nsc-login1 ~# journalctl -k | grep seccomp
```

- **Define which system calls are allowed**
- Blacklist commands

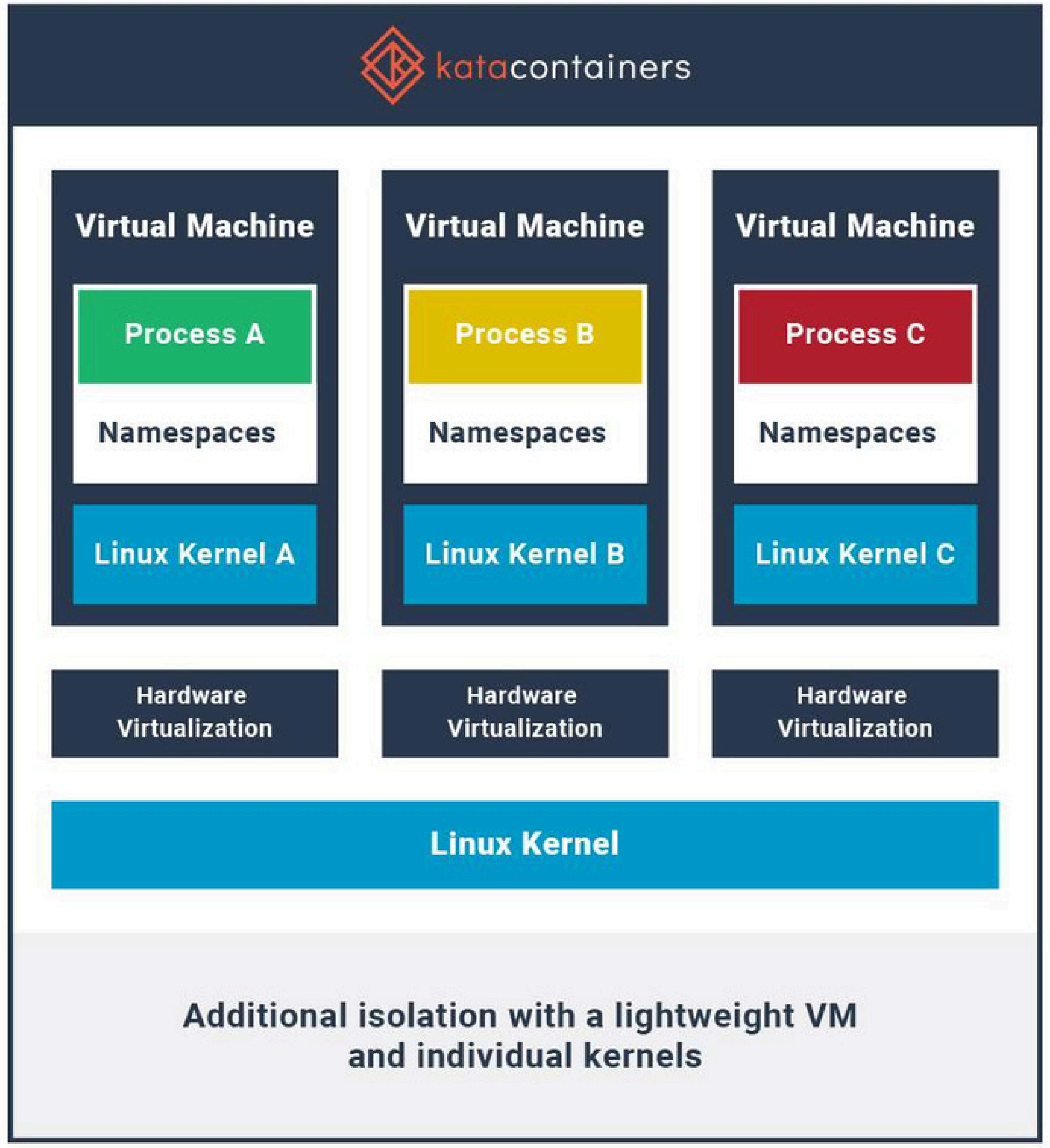
```
--security="seccomp:/usr/local/etc/singularity/seccomp-profiles/default.json"
--security="apparmor:/usr/bin/man"
--security="selinux:context"
--security="uid:1000"
--security="gid:1000"
--security="gid:1000:1:0" (multiple gids, first is always the primary group)
```

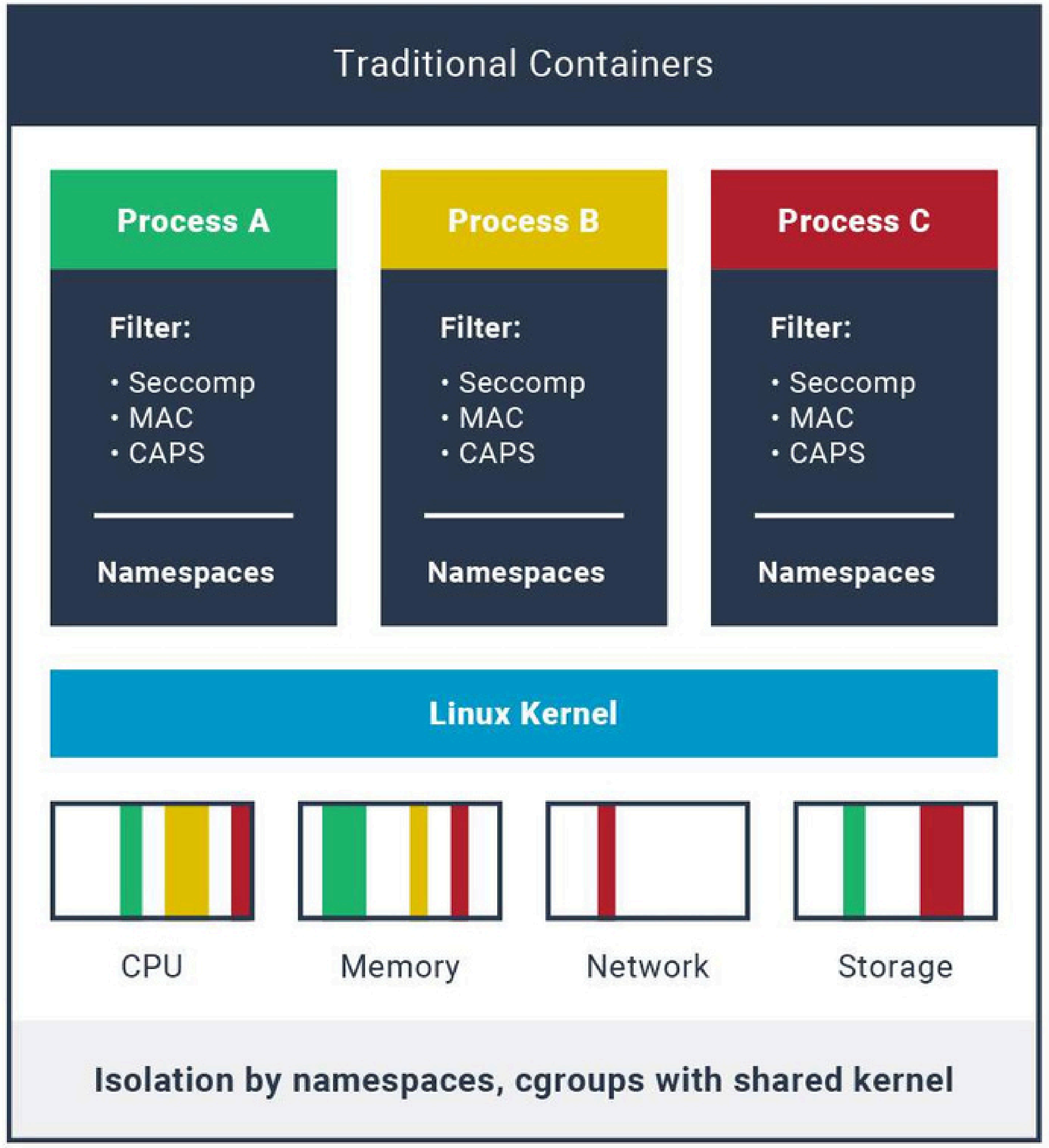
Sandboxing

- Different solutions available:
  - **IBM Nabla**: unikernel as a process with reduced syscalls
  - **gVisor**: user space kernel, implementation of a majority of syscalls, but no GPU support
  - **Firecracker**: uses KVM, works as a microVM, no GPU support
  - **Kata Coontainers**: microVM, supports multiple hypervisors, has support for GPU
  - **Bubblewrap**: wrapper around namespaces and seccomp profiles
  - **Sydbox:** uses seccomp profiles, namespaces, landlock, ptrace and MDWE
  - **Firejail**: requires setuid → large attack surface



E.g.: Kata containers vs Traditional containers





Source: https://katacontainers.io/learn/

Bubblewrap example

```
bwrap --ro-bind /usr /usr --ro-bind /bin /bin --ro-bind /lib /lib --ro-bind
/lib64 /lib64 --ro-bind /etc/passwd /etc/passwd --ro-bind /etc/group
/etc/group --dev /dev --proc /proc --ro-bind /cvmfs /cvmfs --ro-bind
/ceph/grid/home/barbara /home --bind /tmp /tmp  --unshare-net --unshare-user
--uid 1977400011 --gid 1977400011 bash
```

- Since these sandboxes use user namespaces and seccomp profiles, they cannot be combined with containers.
- Similar isolation as the containers provide.

Linux kernel capabilities

Linux capabilities split root privilege into multiple capabilities/privileges that can be granted to processes

- Apptainer/Singularity by default use CAP_SYS_ADMIN, CAP_MKNOD, CAP_SETUID, CAP_SETGID, CAP_DAC_OVERRIDE and CAP_CHOWN
- Capabilities can be added to user
  (see https://docs.sylabs.io/guides/3.0/admin-guide/configfiles.html)

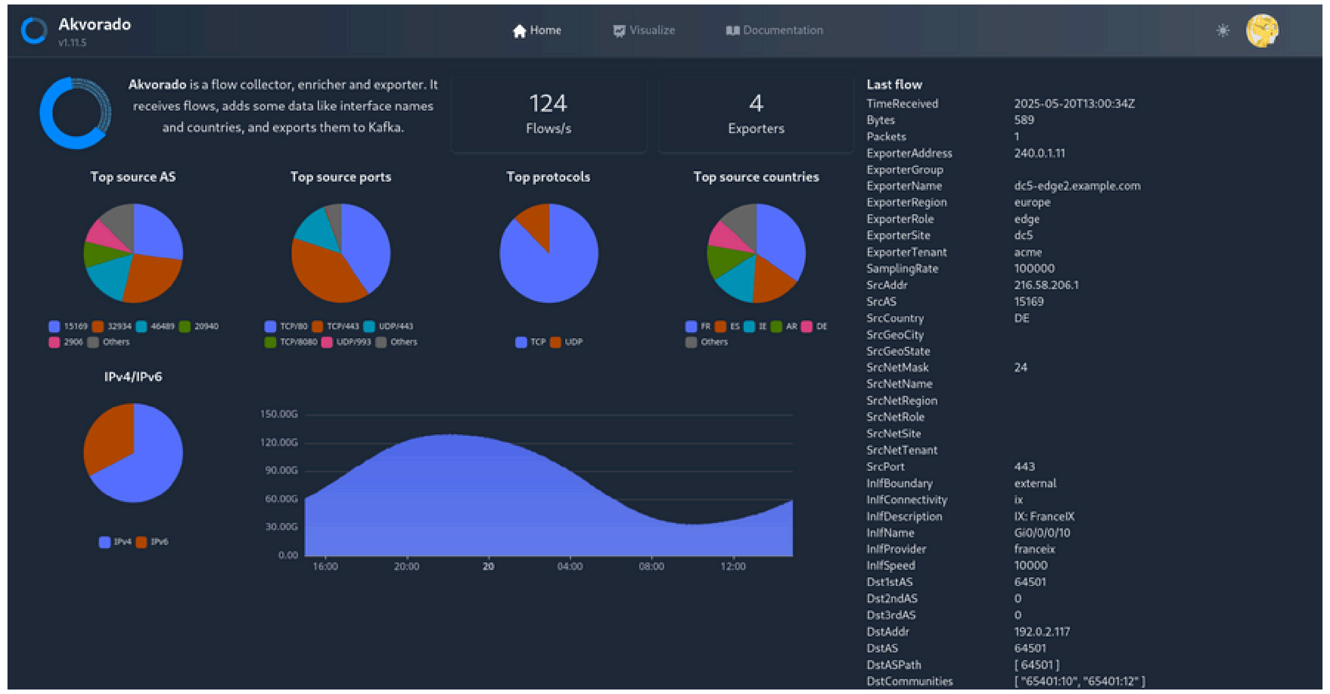
```
nsc-login1 ~# apptainer capability add --user=barbara CAP_NET_RAW
WARNING: Adding 'CAP_NET_RAW' capability will likely allow user barbara to
escalate privilege on the host
WARNING: Use 'apptainer capability drop --user barbara CAP_NET_RAW' to reverse
this action if necessary

nsc-login1 ~# apptainer capability list barbara
barbara [user]: CAP_NET_RAW
```

Linux kernel capabilities (cont.)

```
$ capsh --print
Current: =
Bounding set
=cap_chown,cap_dac_override,cap_dac_read_search,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap
_setpcap,cap_linux_immutable,cap_net_bind_service,cap_net_broadcast,cap_net_admin,cap_net_raw,cap_ipc_lo
ck,cap_ipc_owner,cap_sys_module,cap_sys_rawio,cap_sys_chroot,cap_sys_ptrace,cap_sys_pacct,cap_sys_admin,
cap_sys_boot,cap_sys_nice,cap_sys_resource,cap_sys_time,cap_sys_tty_config,cap_mknod,cap_lease,cap_audit
_write,cap_audit_control,cap_setfcap,cap_mac_override,cap_mac_admin,cap_syslog,cap_wake_alarm,cap_block_
suspend,cap_audit_read,cap_perfmon,cap_bpf,cap_checkpoint_restore
Ambient set =
Current IAB:
Securebits: 00/0x0/1'b0 (no-new-privs=0)
 secure-noroot: no (unlocked)
 secure-no-suid-fixup: no (unlocked)
 secure-keep-caps: no (unlocked)
 secure-no-ambient-raise: no (unlocked)
uid=1977400011(barbara) euid=1977400011(barbara)
gid=1977400011(barbara)
```

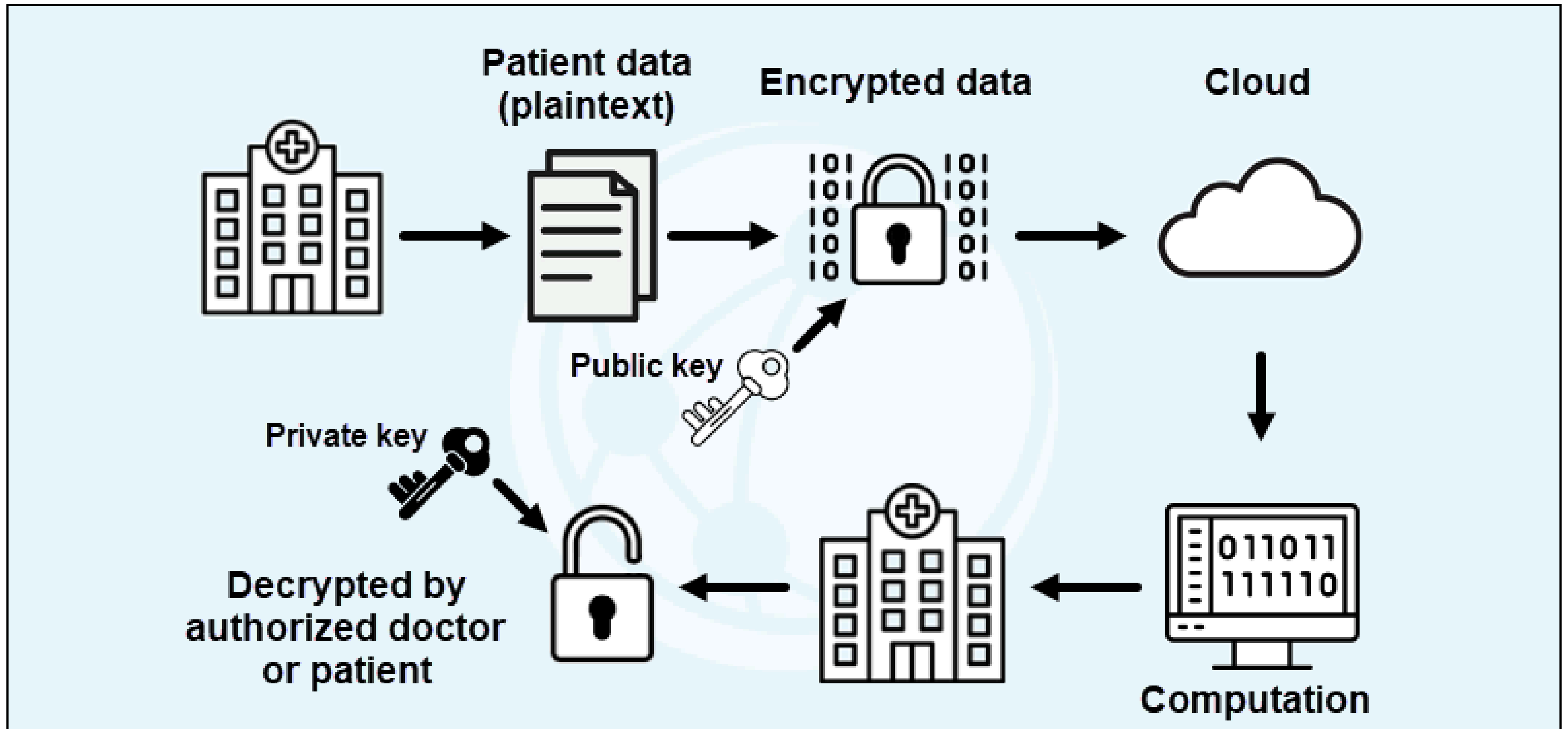
Security monitoring / auditing

- Enable auditing.
- Run active security monitoring on the logs.
- Use RBAC.
- Scan images for vulnerabilities (Trivy, Clair).
- Monitor resource usage by Prometheus/Grafana.



Confidential containers

Homomorphic Encryption



https://phoenixnap.com/kb/homomorphic-encryption

- Allows computations to be performed on encrypted data.
- Sensitive information can remain secure and unexposed while still being processed.
- Computationally very expensive and may impact performance .
- Different tools available, such as IBM's Fully Homomorphic Encryption (FHE) Toolkit for Linux, Microsoft SEAL, HElib, and PALISADE.

Hardware encryption

- Securing data using dedicated hardware components.
- Most (all?) public cloud providers provide hardware encryption.
- Trusted execution environments (TEEs) are isolated execution environment within a device's processor, (e.g., Intel SGX, AMD SEV, or Arm TrustZone) and are based on hardware encryption.
- TEEs provide a secure enclave that isolates data and computations from the rest of the system.
- To protect data in memory also from administrator on the system.
- Trust moves to CPU vendor, using remote attestation.

Hardware encryption: TEE

- process based
- VM based - extends hardware virtualisation support

- CPU
  - Intel SGX
  - Arm TrustZone
  - ARM CCA
  - AMD SEV
  - Intel TDX
  - RISC-V CovE
- GPU
  - relying on CPU TEEs
  - GPU virtualisation
    - GVT (Intel)
    - vGPU
    - MPS
    - MIG

Confidential Containers architecture

- Builds on container runtime, such as Kata Containers, which works as a sandboxed operator.
- Hardware-Based Security: secure enclaves (like Intel SGX or AMD SEV-SNP) to protect data in use.
- Encrypted Memory
- Remote Attestation: Verifies the integrity of the container before execution.
- A confidential attestation operator (Trustee) is needed to provide remote attestation capability (like Intel Trust Authority) and communicates with the Trustee agent in the Kata container.

Confidential computing and GPUs

- Support for confidential computing: **NVIDIA H100 (in 2022) and B100 (in 2024).**

- **H100** uses bounce buffers to exchange data between CPU and GPU, if the workload has a lot of communication between CPU and GPU, the performance overhead is significant (CPU limitations).
- **B100** supports **TDISP/IDE***, new PCI security standard, all encryption is done on the PCI-E bus, you get full performance of the Blackwell architecture.

* https://pcisig.com/blog/ide-and-tdisp-overview-pcie%C2%AE-technology-security-features

Encryption of network traffic

- multi-tenant network
- NVIDIA's BlueField-3 is a Data Processing Unit (DPU)
- multi-tenant environments using virtual interfaces
- offloading and accelerating software-defined networking functions
- hardware-enforced isolation between tenants, ensuring that workloads remain secure and independent.
- network traffic can be encrypted per tenant
- reduces CPU utilisation

Thank you!

Questions?