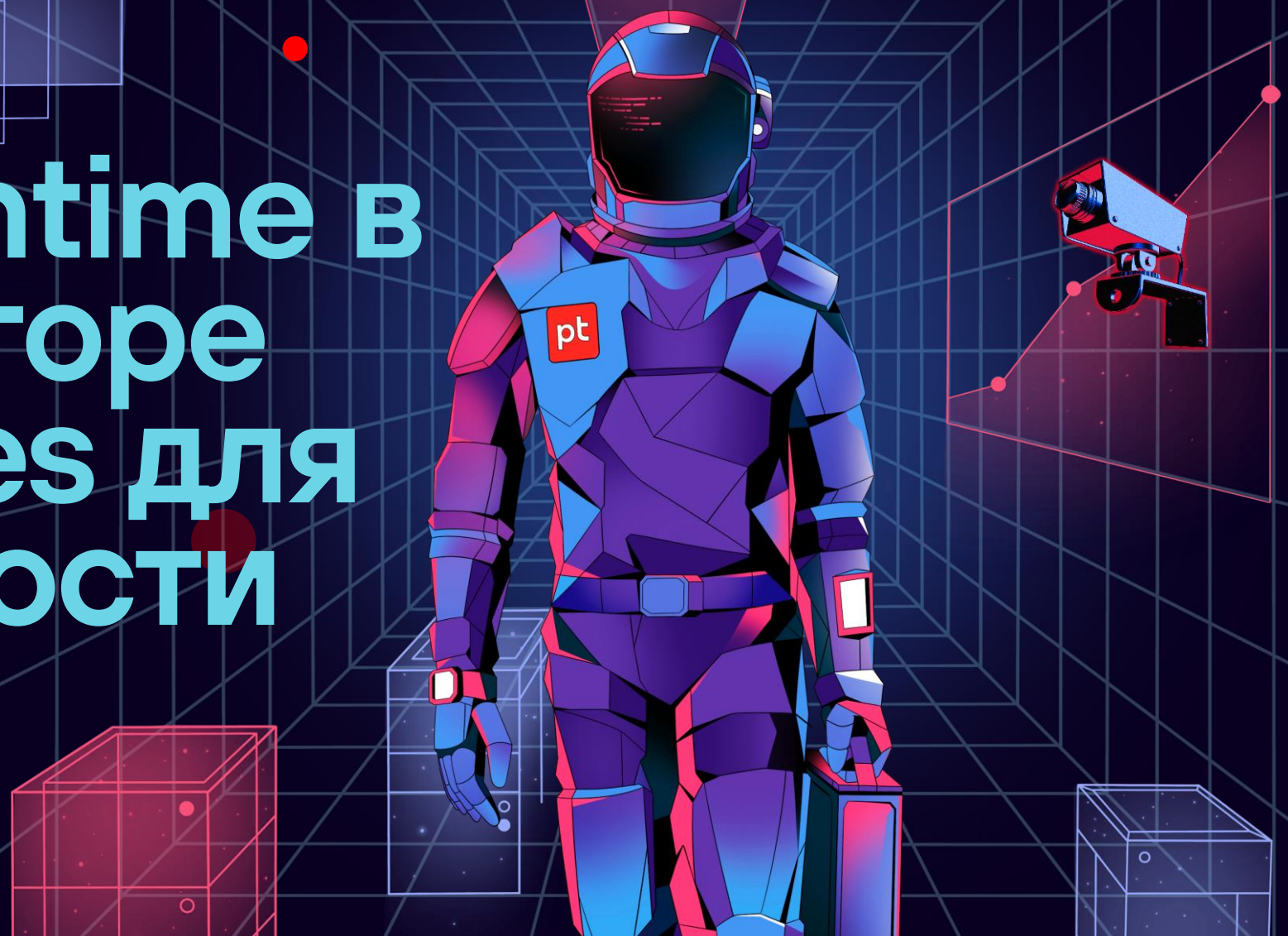
Дмитрий Евдокимов
Основатель и технический директор Luntry

phd 2 Positive Hack Days Fest
от ■ positive technologies

Кубик Runtime в конструкторе Kubernetes для безопасности



WhoAmI

- Основатель и технический директор Luntry
- Опыт в ИБ более 15 лет
- Специализация безопасность контейнеров и Kubernetes
- CFP DevOpsConf, HighLoad++
- Бывший автор статей и редактор рубрик в журнале “ХАКЕР”
- Автор Telegram-канала “k8s (in)security”
- Автор курса “Cloud Native безопасность в Kubernetes”
- Не верит, что систему можно сделать надежной и безопасной, не понимая ее
- Организатор конференции по безопасности контейнеров - БеКон
- Докладчик: BlackHat, HITB, ZeroNights, HackInParis, Confidence, SAS, OFFZONE, PHDays, Kazhackstan, DevOpsConf, DevOops, KuberConf, VK Kubernetes Conference, HighLoad++, БеКон и др.

План выступления

1. Поверхность атаки и побеги из контейнеров
2. RuntimeClass в K8s
3. Изоляция
    - Sandbox/App kernel
        - gVisor
    - microVM
        - kata
        - Confidential Containers (CoCo)
    - WASM
        - WasmEdge
    - Sandbox API
        - Kuasar
4. Заключение

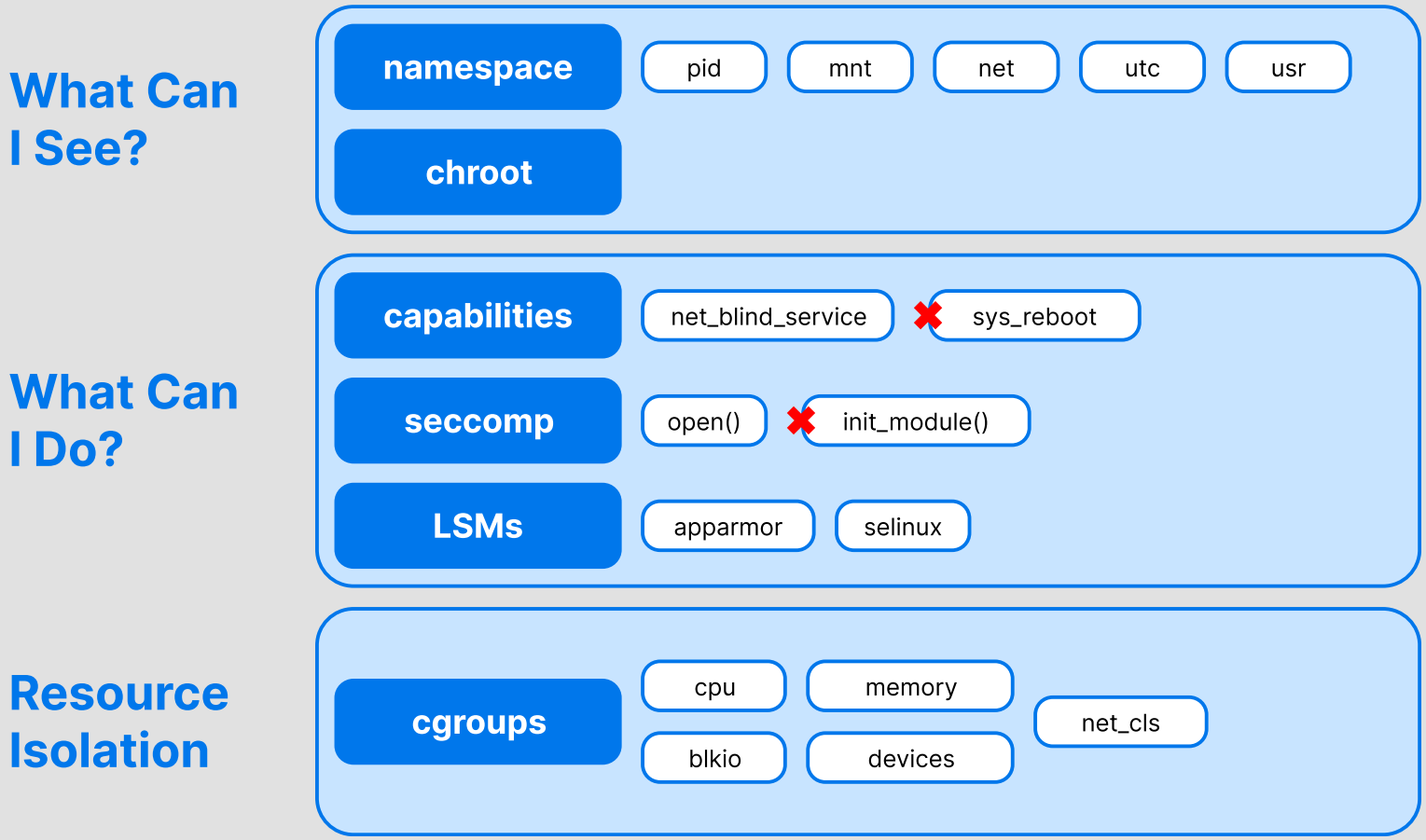
01

phd 2X pt

Поверхность атаки и побеги из контейнеров

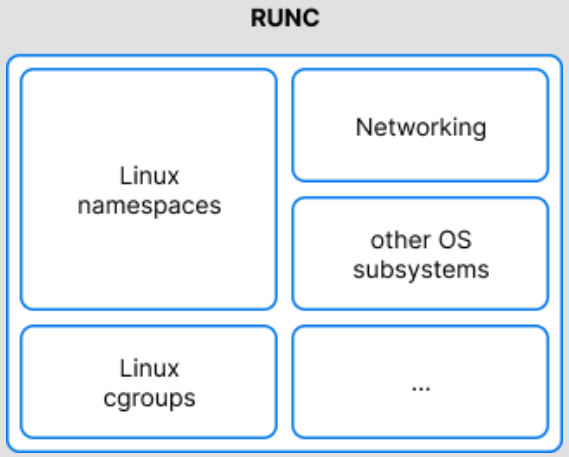
Linux контейнер

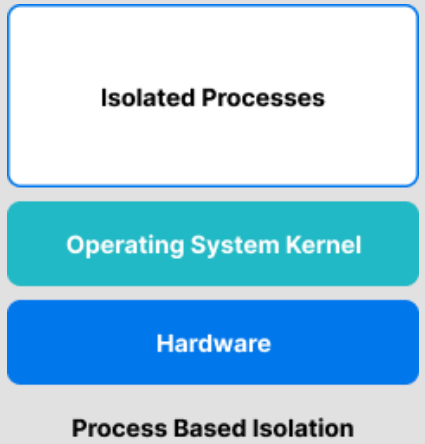
Container = Linux process + cgroup + namespaces (pid, user, uts, ipc, net, mnt, ...) + pivot_root + image

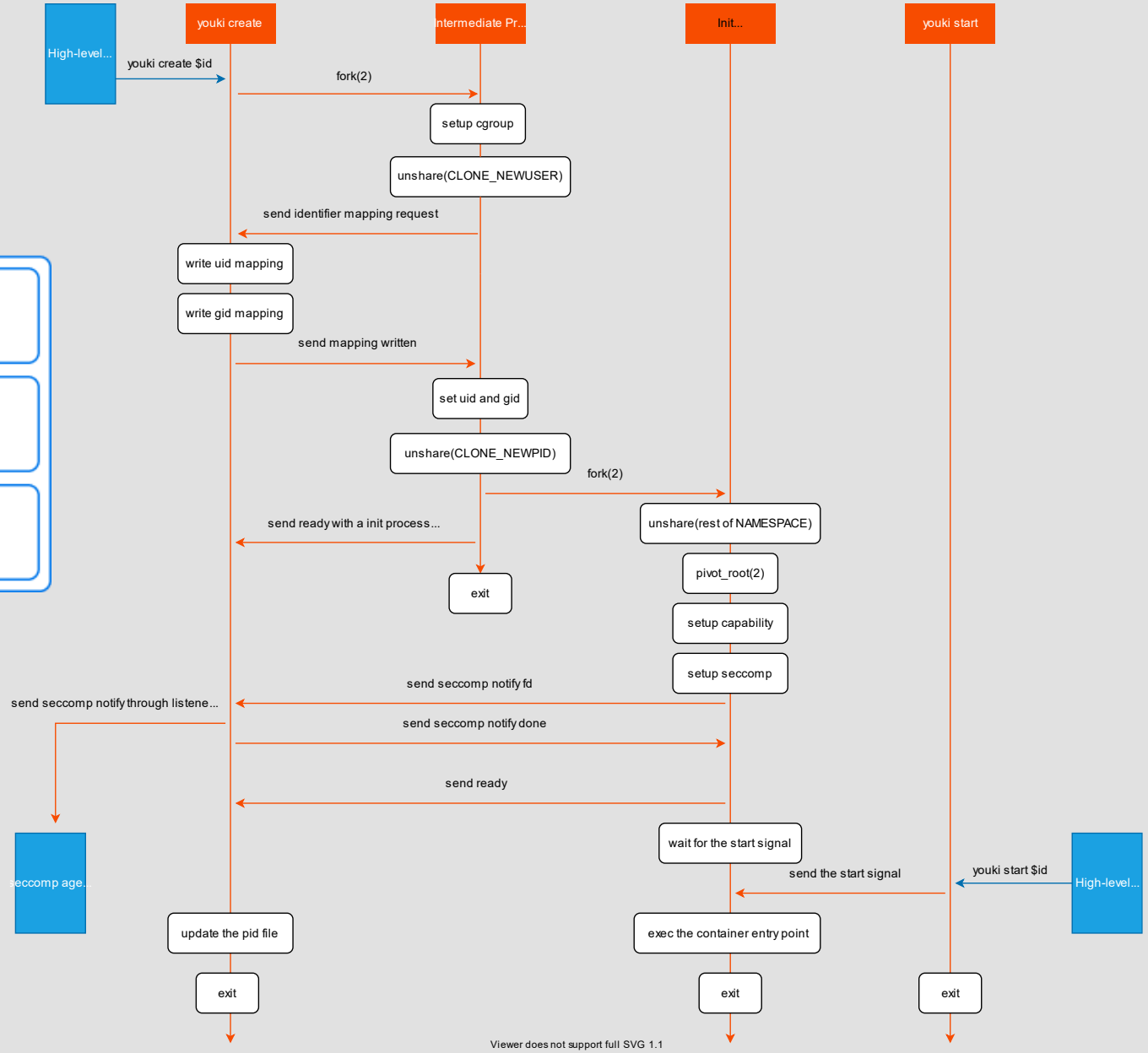


Native containers

- runc
  - Go
- crun
  - C
- Youki
  - Rust







Взгляд с хоста

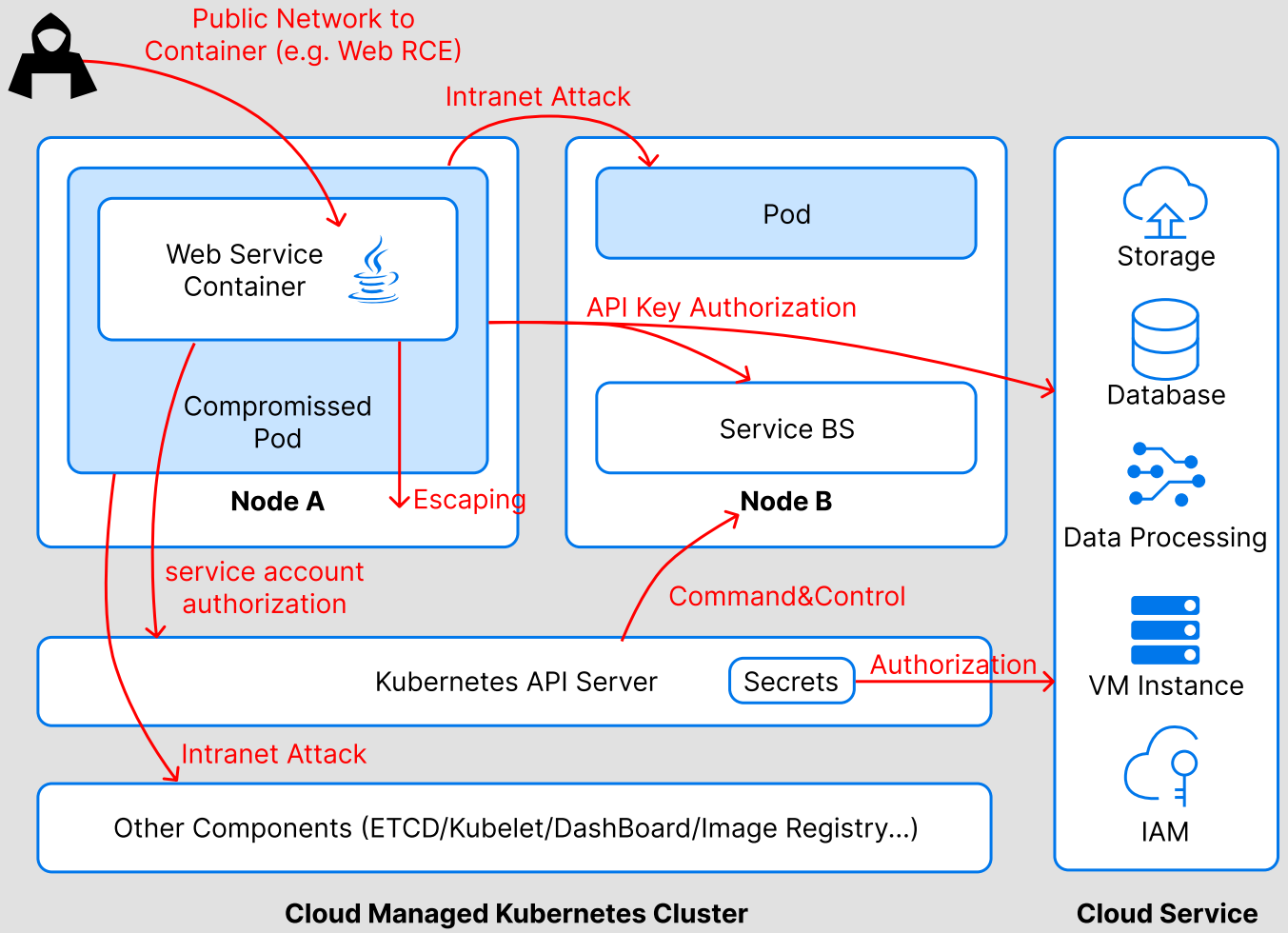
runc

```
   8554 ?        Sl   414:20 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id d4c22c8f5248619f
   8602 ?        Ss     0:00  \_ /pause
2407498 ?        Ss   141:25  \_ /usr/bin/tini -- /usr/local/bin/docker-entrypoint -c /etc/filebeat.
2407509 ?        Sl   11158:26      \_ filebeat -c /etc/filebeat.yml -e
   8700 ?        Sl   400:29 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id 487fb543a56f1455
   8721 ?        Ss     0:00  \_ /pause
   9699 ?        Ssl  302:20  \_ /bin/node_exporter
2332096 ?        Sl   363:31 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id 331319558688f52c
2332117 ?        Ss     0:00  \_ /pause
2228218 ?        Ss     0:00  \_ nginx: master process nginx -g daemon off;
2228254 ?        S      0:00      \_ nginx: worker process
2228255 ?        S      0:00      \_ nginx: worker process
```

```
2337545 ?        Sl    17:43 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id f9ea
2337564 ?        Ss     0:00  \_ /pause
2189220 ?        Ssl    7:07  \_ sensor -v --readiness
2347410 ?        Sl    17:56 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id cb88
2347434 ?        Ss     0:00  \_ /pause
2347464 ?        Ssl    2:20  \_ /app -port=80
2347792 ?        Sl    40:32 /usr/bin/containerd-shim-runc-v2 -namespace k8s.io -id a45d
2347810 ?        Ss     0:00  \_ /pause
1292416 ?        Ssl  118:06  \_ /src/server
```

Поверхность атаки

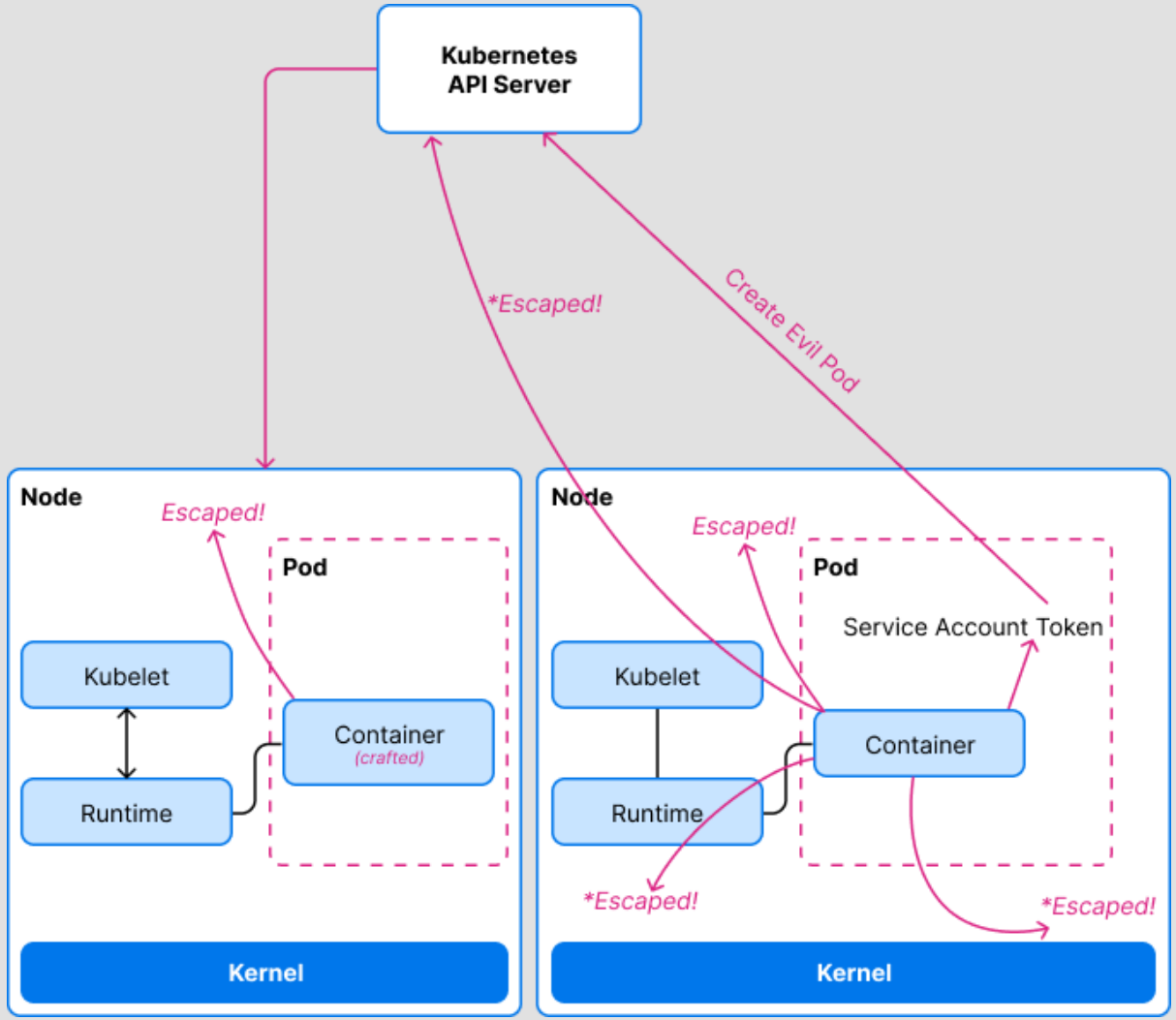
1. С public network в Pod
2. С Pod на другие Pods/Services
3. С Pod на Node (Escape)
4. С Pod на Master Node Components
5. С Pod на API Server
6. С API Server на другие Pods/Nodes
7. С K8s кластера на Cloud Service



Побеги из контейнеров

Вектора в K8s

- Bad Pods
- Создание Bad Pods
- Уязвимости container runtime
- Уязвимости ядра
- Уязвимости стороннего ПО в Kubernetes
- Уязвимости компонентов Kubernetes
- В теории: hardware attacks Exploiting Speculative Execution



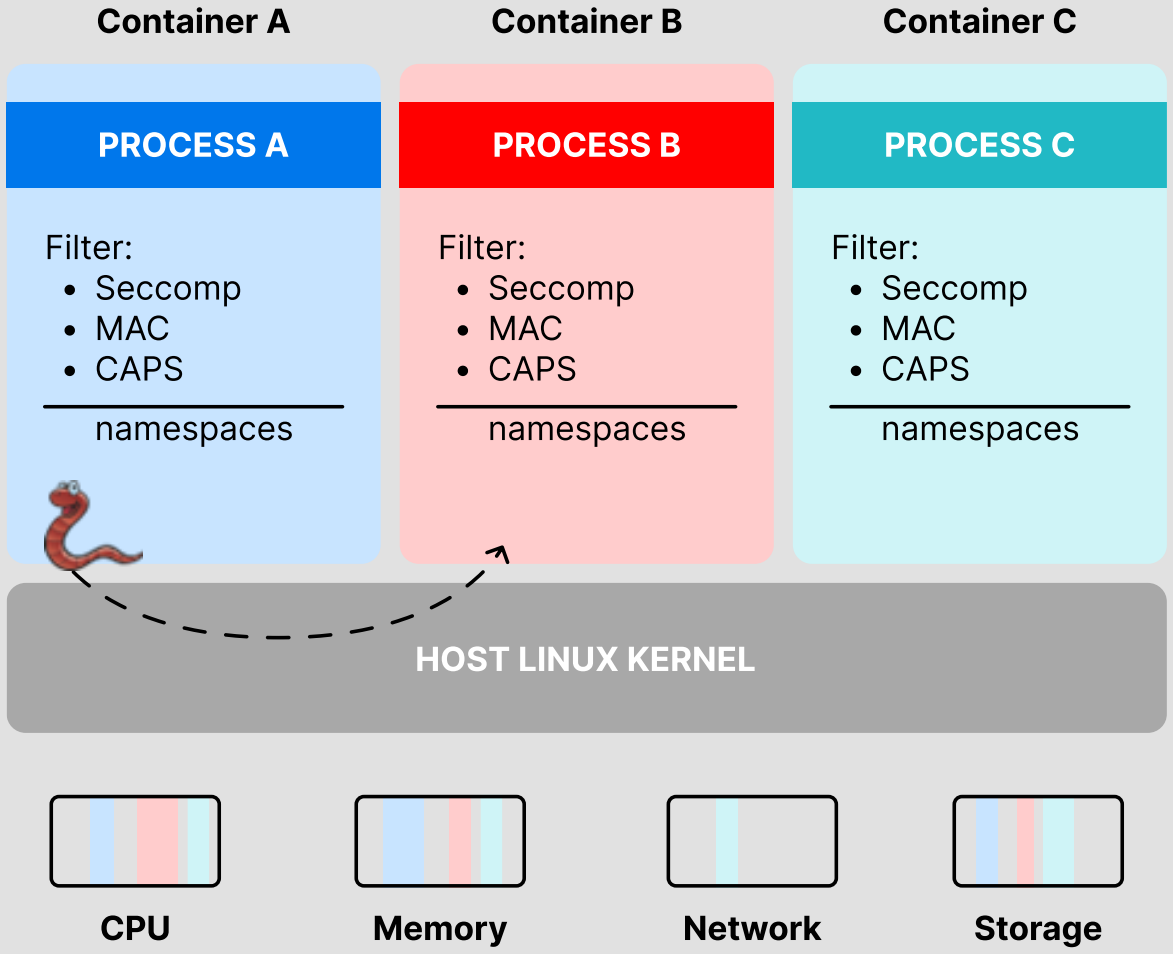
ZeroNights 2021. “Container escapes: Kubernetes edition”

Уязвимости ядра

Все контейнеры разделяют между собой ядро хостовой ОС.

Примеры уязвимостей для побега:

- CVE-2023-3389
- CVE-2023-0461
- CVE-2022-0847
- CVE-2022-0492
- CVE-2022-0185
- CVE-2021-22555
- CVE-2021-31440
- CVE-2020-14386
- CVE-2020-8835
- ...



Защита

Необходимо:

- усложнить запуск кода атакующего
- уменьшить поверхность атаки
- обновлять ядро хостовой ОС

Механизмы и подходы защиты:

- seccomp
- AppArmor
- SeLinux
- SecurityContext
- Rootless containers
- Tiny/Slim/Distroless images
- OS host machine hardening
- Alternative Runtimes

Применимо к native containers

02

phd 2X pt

RuntimeClass в K8s

Kubernetes конструктор

halvarflake @halvarflake · 21 ч.
Dear Kubernetes followers: If you run a Kubernetes cluster across AZ boundaries, what do you use for networking? In general, if you run Kubernetes on Cloud machines, what is your networking layer? Calico? Flannel? Cilium? Something else?

12 5 19

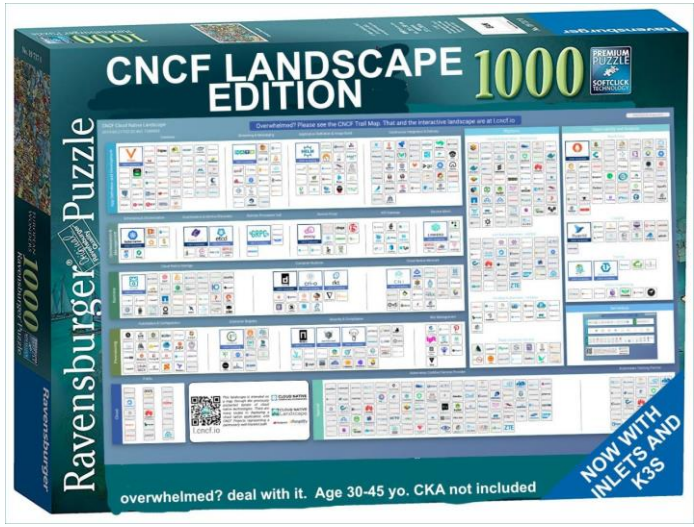
Показать другие ответы

Ian Coldwater 📦💥 @IanCol... · 21 ч.
В ответ @halvarflake
Each Kubernetes cluster is 110% a unicorn, sorry

2 3

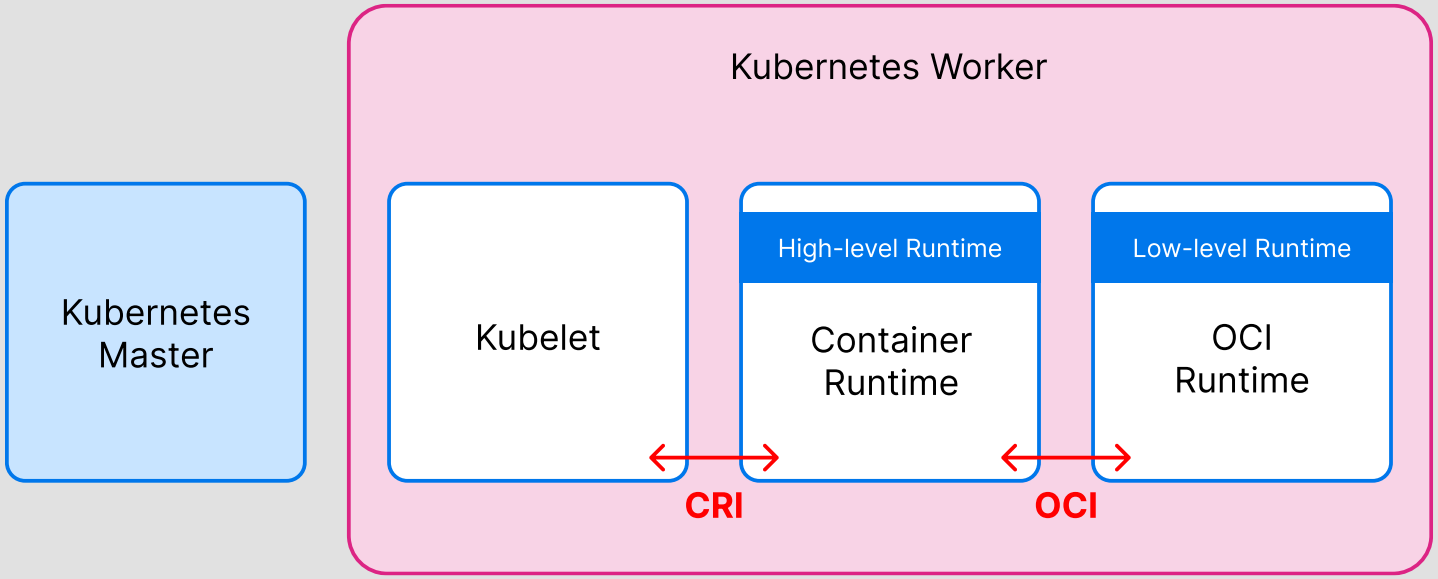
Ian Coldwater 📦💥 @IanCol... · 21 ч.
A special unique snowflake of a sparkling distributed attack surface ✨

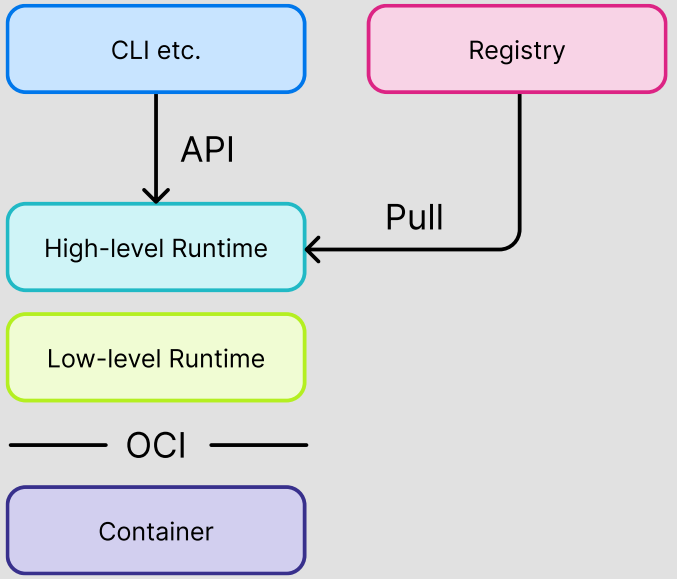
4



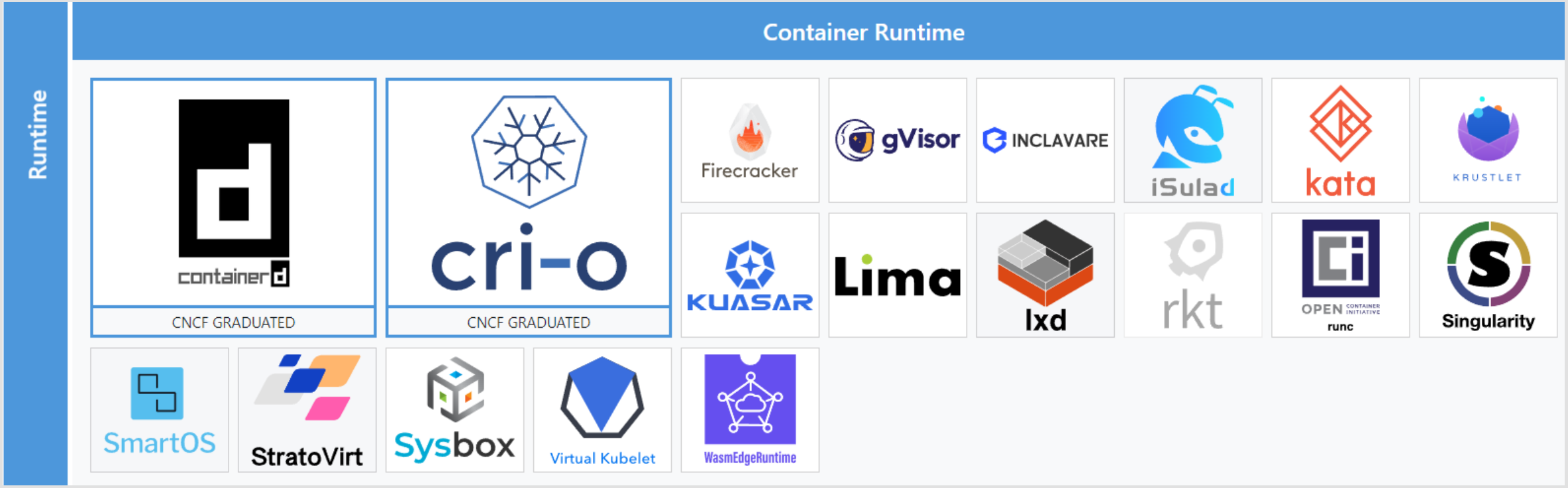
Типы Container Runtimes

- High-level Runtime
- Low-level Runtime
- CRI - Container Runtime Interface
- OCI - Open Container Initiative





Runtimes в K8s



https://landscape.cncf.io/

RuntimeClass и runtimeClassName

```yaml
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: gvisor
handler: runsc
```

```yaml
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: wasmedge
handler: wasmedge
```

```yaml
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: gvisor-kvm
handler: runsc-kvm
```

```yaml
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: kuasar-vmm
handler: kuasar-vmm
```

```yaml
kind: RuntimeClass
apiVersion: node.k8s.io/v1
metadata:
  name: kata-qemu
handler: kata-qemu
overhead:
  podFixed:
    memory: "160Mi"
    cpu: "250m"
scheduling:
  nodeSelector:
    katacontainers.io/kata-runtime: "true"
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: gvisor-nginx
  name: gvisor-nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: gvisor-nginx
  template:
    metadata:
      labels:
        app: gvisor-nginx
    spec:
      runtimeClassName: gvisor
      topologySpreadConstraints:
      - maxSkew: 1
        topologyKey: kubernetes.io/hostname
        whenUnsatisfiable: DoNotSchedule
        labelSelector:
          matchLabels:
            app: gvisor-nginx
      containers:
        - name: nginx
          image: nginx
          imagePullPolicy: IfNotPresent
      restartPolicy: Always
```

Non-containers

“Non-container” containers

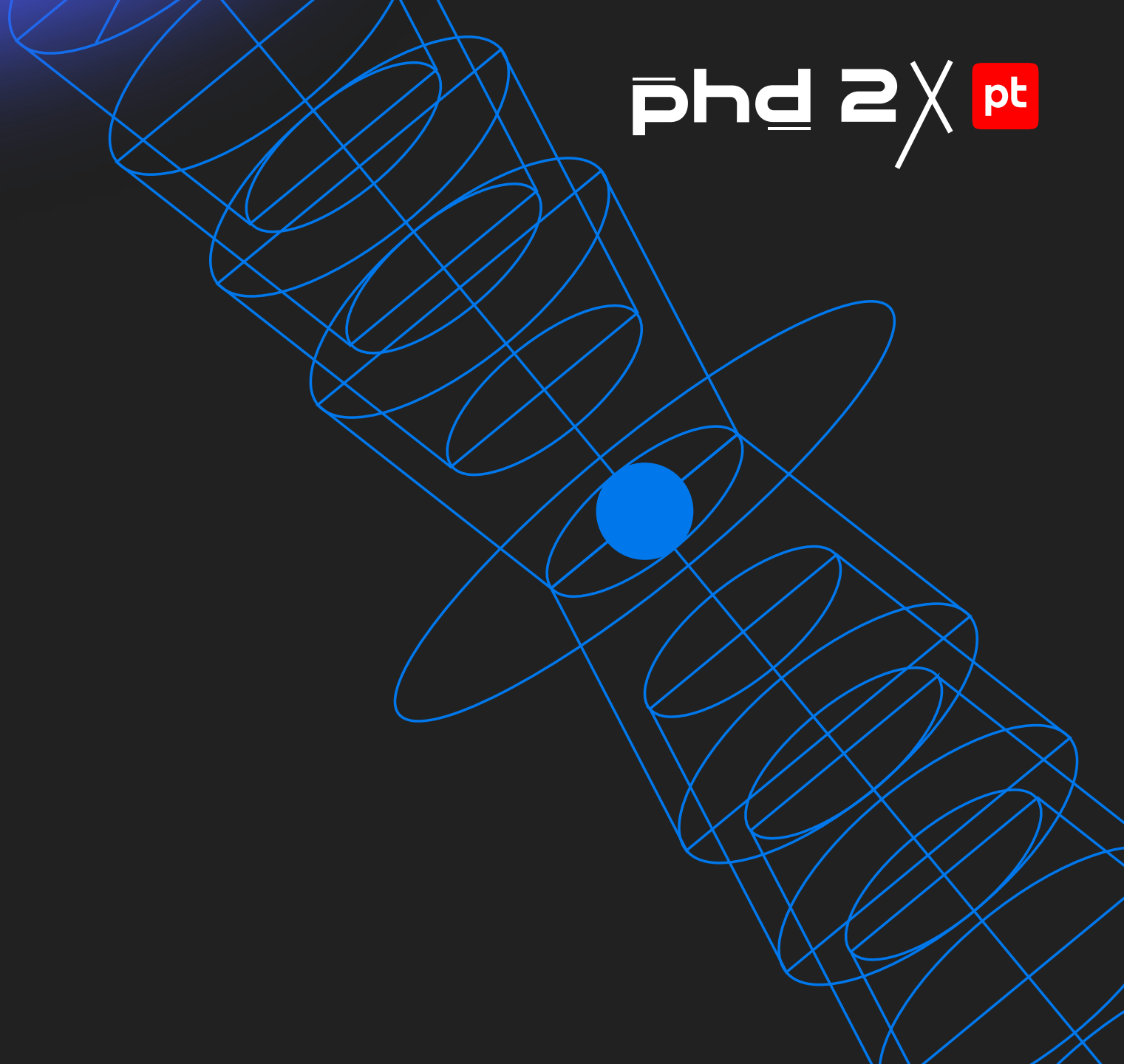
NTT

- “Containers” are not well defined

- Almost anything can be called a “container runtime” when it accepts OCI formats 😅

03

phd 2X pt



Изоляция

Изоляция и другие подходы

Isolation
- Дополнительный уровень изоляции от ядра хостовой ОС
- WASM
  - WasmEdge, Wasmtime, Wasmer, ...
- Sandbox/App kernel
  - gVisor, Quark
- MicroVM
  - Kata containers

Detection
- Идентификация нежелательного действия
- Сторонние решения в k8s

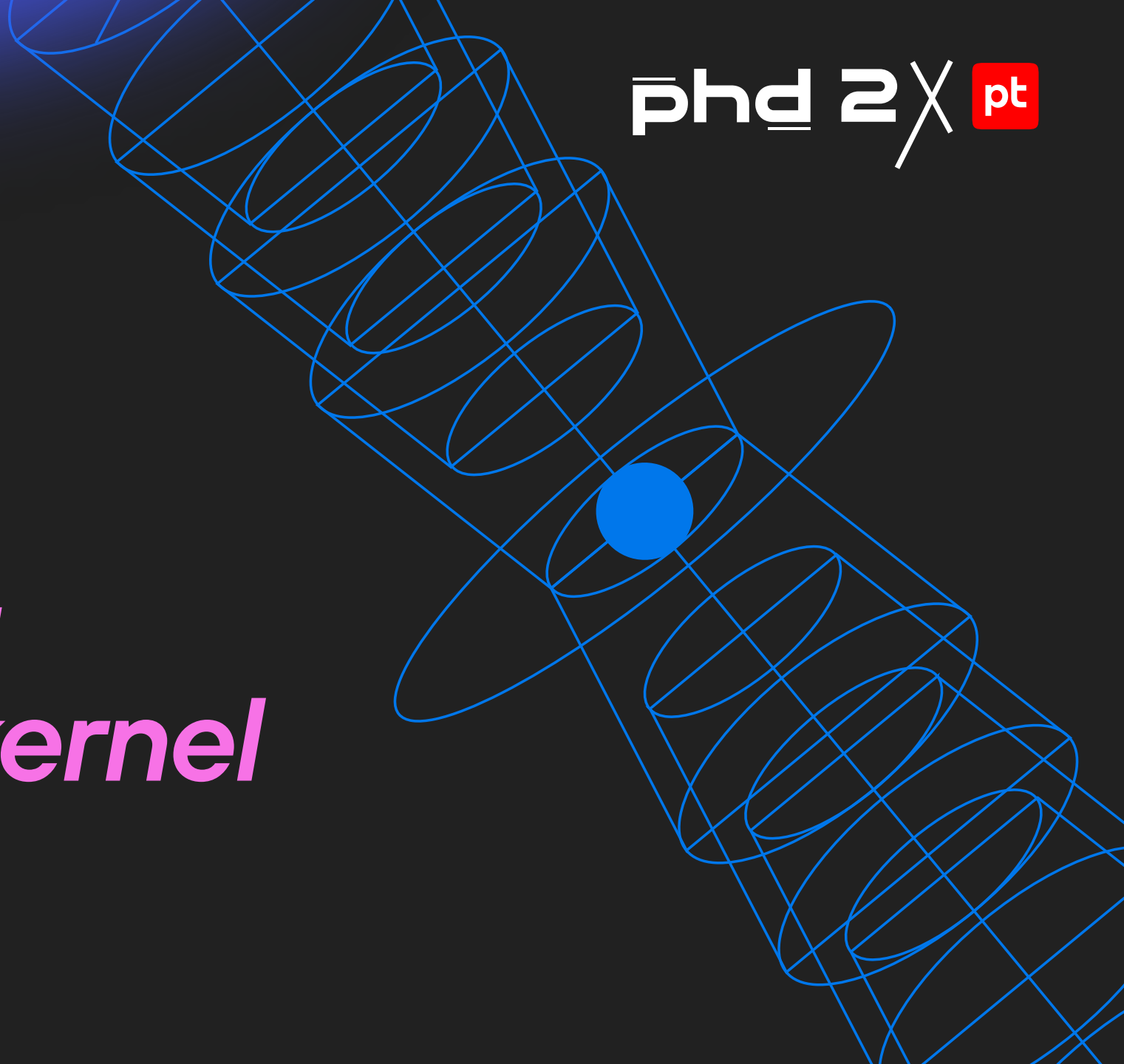
Prevention
- Невозможность выполнения нежелательного действия
- Seccomp
- AppArmor
- SeLinux
- LSM
- iptables/eBPF

Mitigation
- Усложнение эксплуатации уязвимостей
- Stack cookies
- W^X
- ASLR
- SMEP
- SMAP
- ...

Reaction
- Завершение процесса/контейнера/Pod постфактум после нежелательного события и/или дамп файловой системы и/или памяти
- CRI интерфейс (stop/remove)
- Kubelet Checkpoint API

Вебинар Luntry. “Runtime Security: на вкус и цвет все фломастеры разные”

03.1

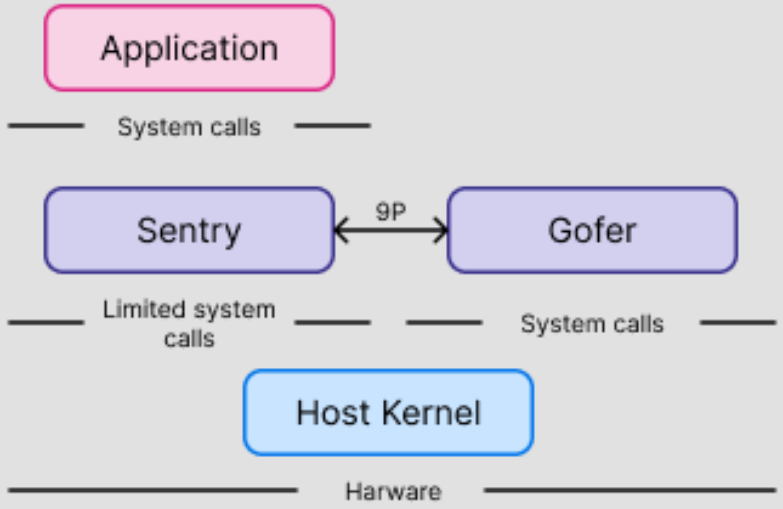


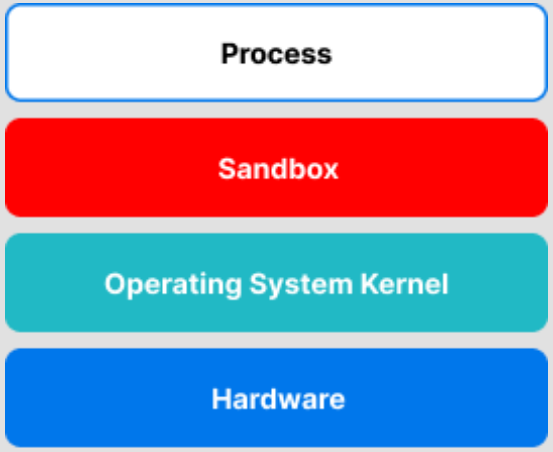
Sandbox или Application kernel

Sandbox

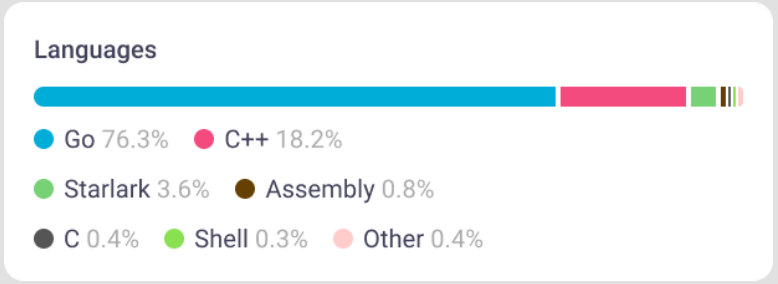
gVisor - Application Kernel for Containers.

- Использует OCI совместимый runtime под названием runsc.
- Создан как дополнительный механизм защиты против эксплуатации уязвимостей ядра недоверенным user-space кодом
- Похож на механизм используемый в User-Mode Linux (UML).





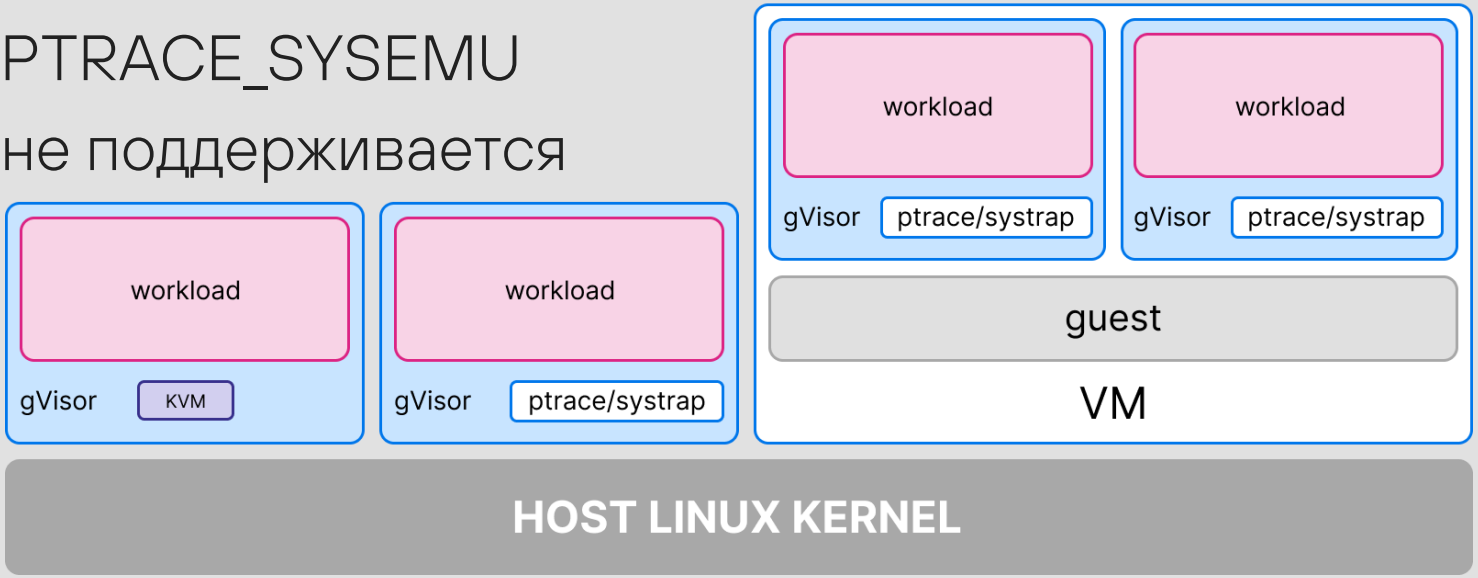
Sandbox Based Isolation



Реализация gVisor

- KVM
  - Требуется вложенная виртуализация в виртуальной машине
- systrap
  - На базе seccomp's SECCOMP_RET_TRAP
  - По умолчанию с середины 2023
- ptrace
  - На базе PTRACE_SYSEMU
  - Больше не поддерживается

```
{
    "runtimes": {
        "runsc-kvm": {
            "path": "/usr/local/bin/runsc",
            "runtimeArgs": [
                "--platform=kvm"
            ]
        },
        "runsc-systrap": {
            "path": "/usr/local/bin/runsc",
            "runtimeArgs": [
                "--platform=systrap"
            ]
        }
    }
}
```



Взгляд с хоста

gVisor release-20240501.0


```
70342 ?        Sl     0:00 /usr/local/bin/containerd-shim-runsc-v1 -namespace k8s.io -address /run/containerd
70356 ?        Ssl    0:00  \_ runsc-gofer --log-format=json --panic-log=/var/log/pods/default_gvisor-nginx-5
70361 ?        Ssl    0:39  \_ runsc-sandbox --log=/run/containerd/io.containerd.runtime.v2.task/k8s.io/74e82
70387 ?        Ss     0:00  |   \_ [exe]
70406 ?        S      0:00  |   \_ [exe]
70416 ?        SN     0:00  |   |   \_ [exe]
70474 ?        S      0:00  |   \_ [exe]
70479 ?        SN     0:00  |   |   \_ [exe]
70487 ?        S      0:00  |   \_ [exe]
70488 ?        SN     0:00  |   |   \_ [exe]
70489 ?        S      0:00  |   \_ [exe]
70490 ?        SN     0:00  |   |   \_ [exe]
70491 ?        S      0:00  |   \_ [exe]
70492 ?        SN     0:00  |   |   \_ [exe]
70493 ?        S      0:00  |   \_ [exe]
70494 ?        SN     0:00  |   |   \_ [exe]
70497 ?        S      0:00  |   \_ [exe]
70500 ?        SN     0:00  |   |   \_ [exe]
70498 ?        S      0:00  |   \_ [exe]
70499 ?        SN     0:00  |   |   \_ [exe]
70501 ?        S      0:00  |   \_ [exe]
70502 ?        SN     0:00  |       \_ [exe]
70407 ?        Sl     0:00  \_ runsc --root=/run/containerd/runsc/k8s.io --log=/run/containerd/io.containerd.
```

Таксономия проблем безопасности

gVisor

Security issue taxonomy

We distinguish the following type of issues, listed from most to least severe:

- Issues that go **beyond the sandbox boundary**:
  - **Container escapes**: Issues that allow arbitrary code to run on the host machine.
    - gVisor's purpose is to prevent these.
  - **Data exfiltration** from the host: Issues that allow reading arbitrary files or file metadata from the host (other than those intended to be visible to the sandbox).
  - **Sandbox-to-sandbox lateral movement**: Issues that allow arbitrary code execution in a different sandbox on the same host.
  - **Denial-of-service attacks** that affect **the host kernel** (i.e. trigger a host kernel panic).
  - **Denial-of-service attacks** that affect **other sandboxes on the same host**.
    - This excludes things like causing CPU starvation when a sandbox is running without resource constraints.
- Issues that **remain confined to a single sandbox**:
  - **Denial-of-service attacks** that affect a single sandbox and are **triggerable remotely** (e.g. by sending a specially-crafted network packet).
  - **Privilege escalation within the sandbox** (e.g. being able to do what in-sandbox `root` would be able to do from an in-sandbox non-`root` user).
  - **Denial-of-service attacks** that affect a single sandbox and are **triggerable from user code** running in that sandbox.
  - **Data integrity issues** relative to Linux behavior.
    - gVisor aims to be bug-for-bug compatible with Linux. While most compatibility issues are not security issues, it is conceivable that some compatibility issues may manifest as persistent data corruption; for example, differences in I/O syscall implementations may cause a database program to end up storing invalid data.

While all of the above are security issues, we generally only assign CVEs for issues that go beyond the sandbox boundary. Since gVisor is a container security platform, its main security focus is on preventing a user workload from "getting out of the box", relative to issues that remain

SECURITY.md

Особенности gVisor

Checkpoint and Restore*

Checkpoint and Restore

gVisor can **checkpoint and restore containers**. Use it to cache warmed-up services, resume workloads on other machines, snapshot execution, save state for forensics, or branch interactive REPL sessions.

Runtime Monitoring

Runtime Monitoring

Observe runtime behavior of your applications by streaming application actions (trace points) to an external **threat detection engine** like Falco and generate alerts.

* Только через raw команды к runsc или частичная поддержка Docker

Мысли о gVisor

Большой вопрос совместимости с приложениями

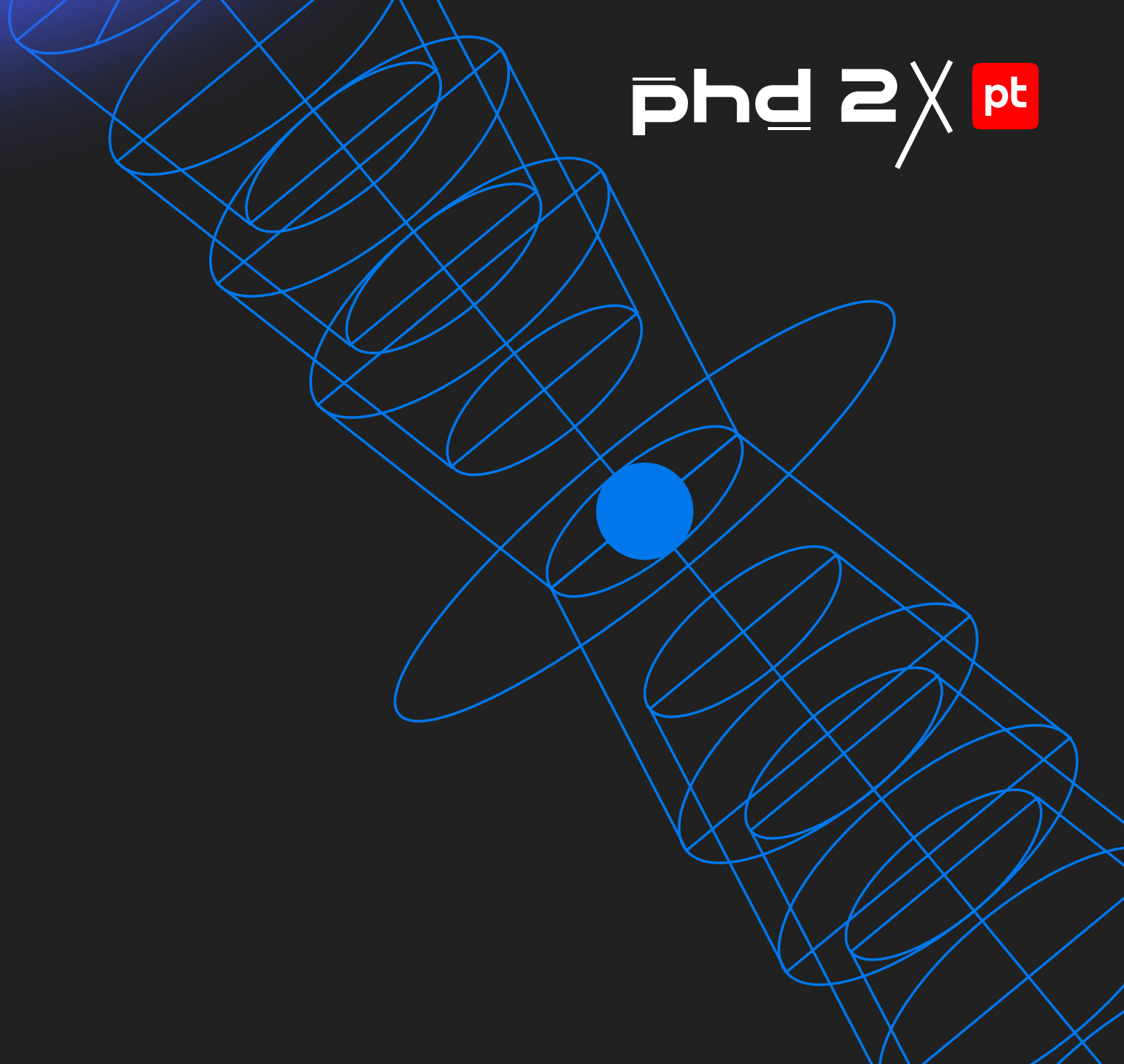
Application

The application is a normal Linux binary provided to gVisor in an OCI runtime bundle. gVisor aims to provide an environment equivalent to Linux v4.4, so applications should be able to run unmodified. However, gVisor does not presently implement every system call, `/proc` file, or `/sys` file so some incompatibilities may occur. See Compatibility for more information.

gVisor implements a large portion of the Linux surface and while we strive to make it broadly compatible, there are (and always will be) unimplemented features and bugs. The only real way to know if it will work is to try. If you find a container that doesn't work and there is no known issue, please file a bug indicating the full command you used to run the image. You can view open issues related to compatibility here.

03.2

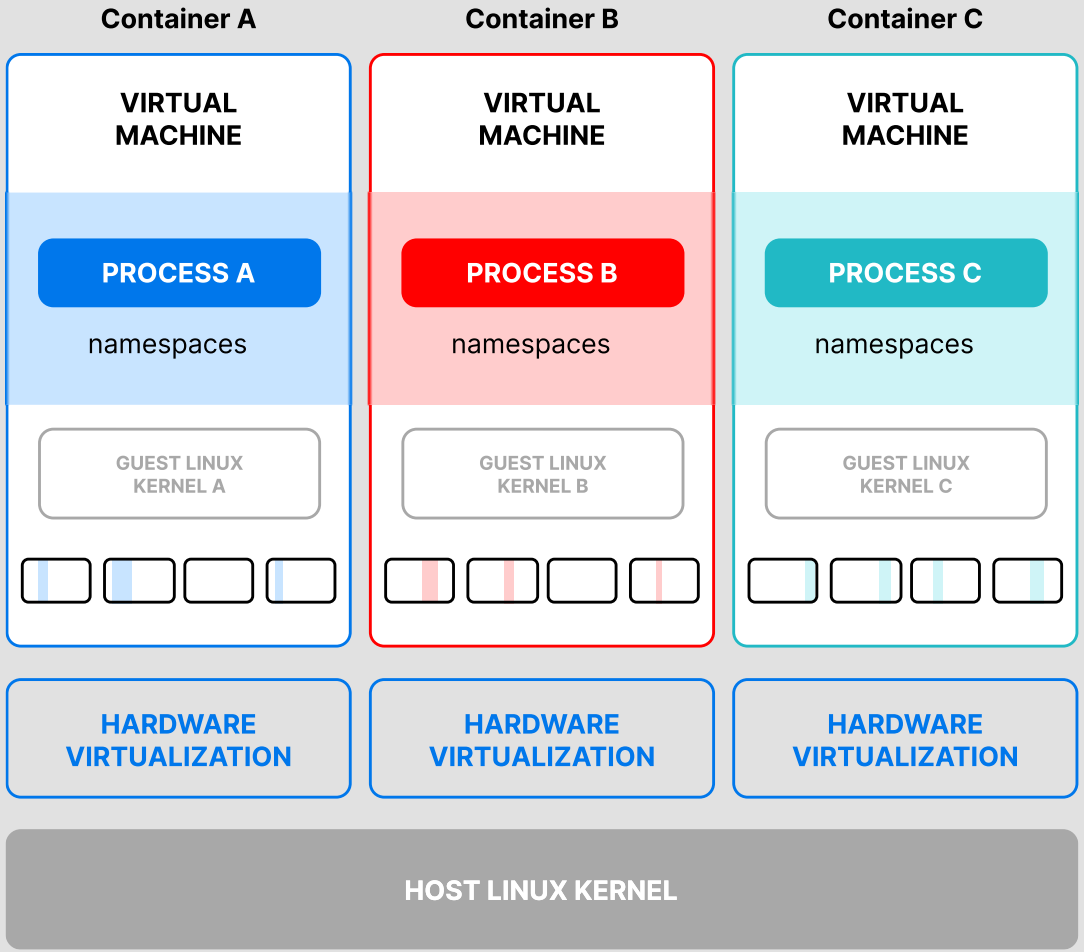
phd 2X pt

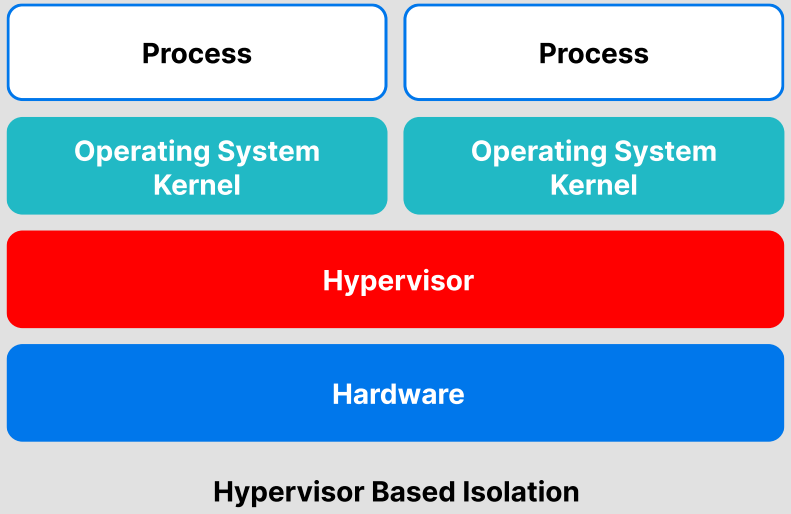


microVM

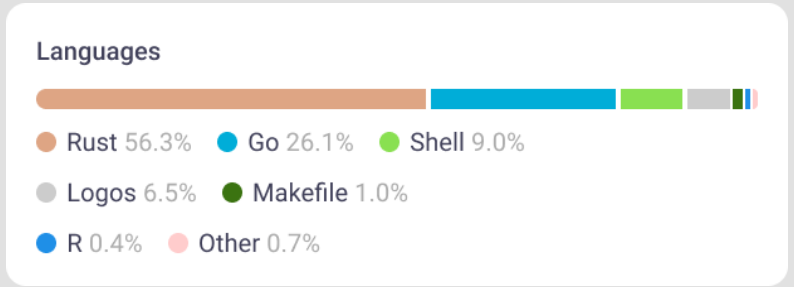
microVM

Kata containers - lightweight Virtual Machines (VMs).

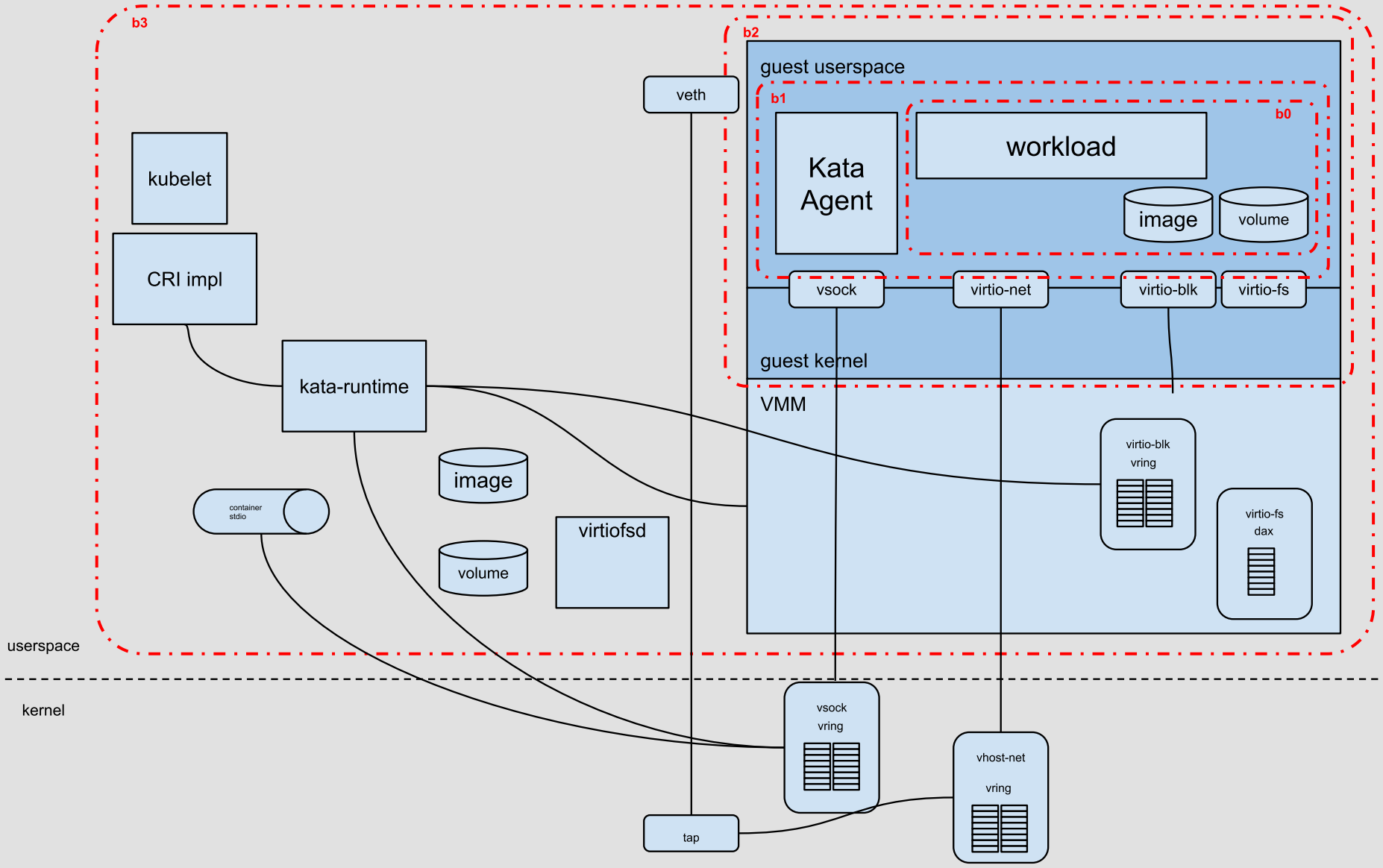




Hypervisor Based Isolation



Реализация Kata

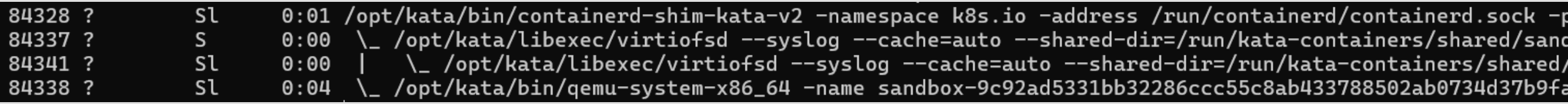


Выбор hypervisor

Hypervisor	Written in	Architectures	Type	Summary	Features	Limitations	Container Creation speed	Memory density	Use cases	Comment
ACRN	C	.x86_64	Type 1 (bare metal)	Safety critical and real-time workloads			excellent	excellent	Embedded and IOT systems	For advanced users
Cloud Hypervisor	rust	aarch64 , x86_64	Type 2 (KVM)	Low latency, small memory footprint, small attack surface	Minimal		excellent	excellent	High performance modern cloud workloads	
Firecracker	rust	aarch64 , x86_64	Type 2 (KVM)	Very slimline	Extremely minimal	Doesn't support all device types	excellent	excellent	Serverless / FaaS	
QEMU	C	All	Type 2 (KVM)	Lots of features	Lots		good	good	Good option for most users	
Dragonball	rust	aarch64 , x86_64	Type 2 (KVM)	Built-in VMM, low CPU and memory overhead	Minimal		excellent	excellent	Optimized for most container workloads	out-of-the-box Kata Containers experience
StratoVirt	rust	aarch64 , x86_64	Type 2 (KVM)	Unified architecture supporting three scenarios: VM, container, and serverless	Extremely minimal (MicroWM) to Lots (StandardWM)		excellent	excellent	Common container workloads	StandardVM Type of StratoVirt for Kata is under development

Взгляд с хоста

Kata 3.4.0



Модель угроз

kata containers

Особенности Kata

- Создание собственных минимальных kernel и rootfs

osbuilder

Introduction

The Kata Containers runtime creates a virtual machine (VM) to isolate a set of container workloads. The VM requires a guest kernel and a guest operating system ("guest OS") to boot and create containers inside the guest environment.

This repository contains tools to create a guest OS disk image.

- Наличие VM templating для быстрого запуска

What is VM templating

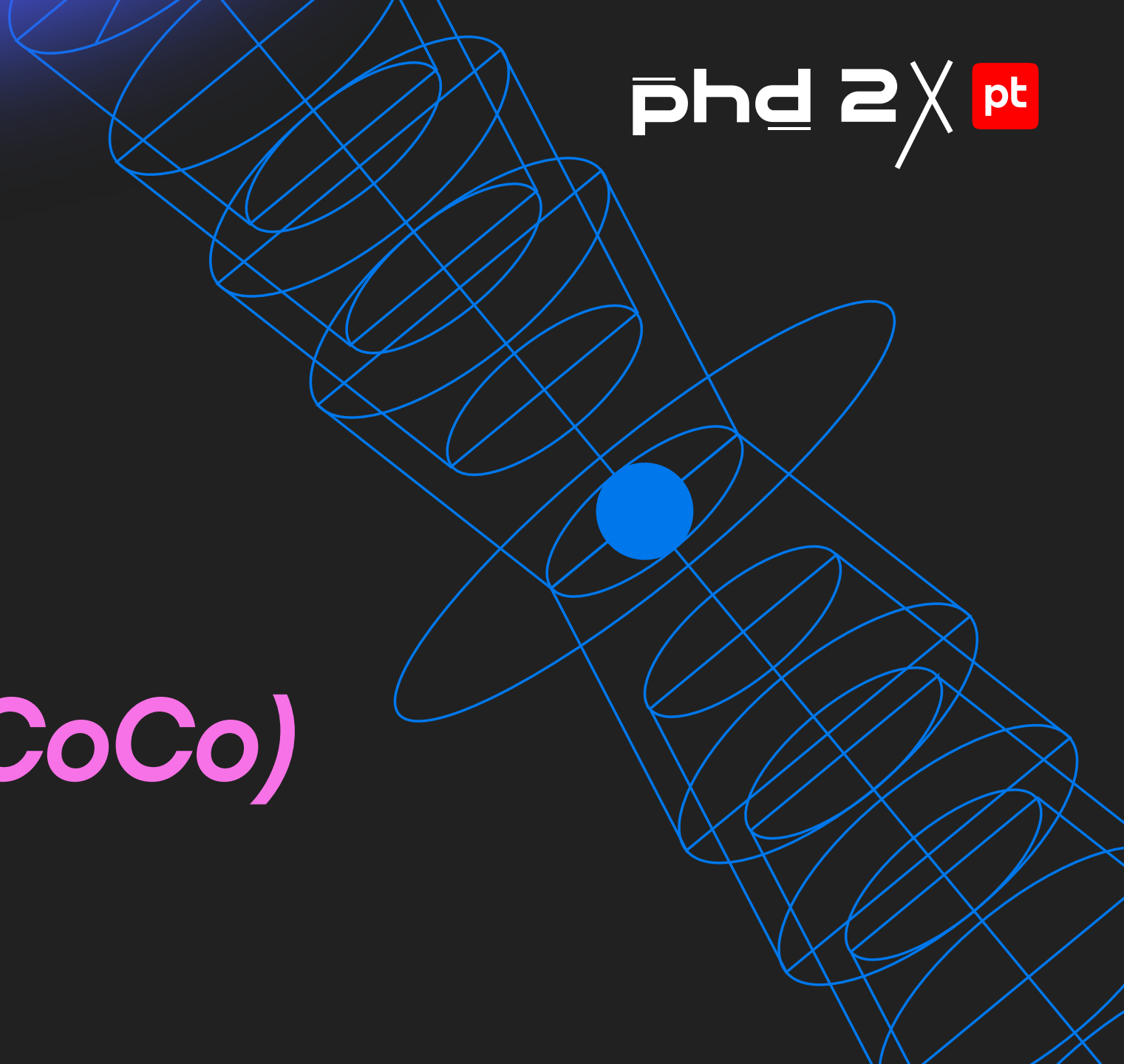
VM templating is a Kata Containers feature that enables new VM creation using a cloning technique. When enabled, new VMs are created by cloning from a pre-created template VM, and they will share the same initramfs, kernel and agent memory in readonly mode. It is very much like a process fork done by the kernel but here we *fork* VMs.

Мысли о Kata

- Ограничения
  - Выполнение команд, сеть, управление ресурсами, работа со storage, разделение ресурсов с хостом, ...
- Требование к hardware по виртуализации
- Вопрос observability
  - Появление слепых зон в инфраструктуре

03.2.1

phd 2X pt

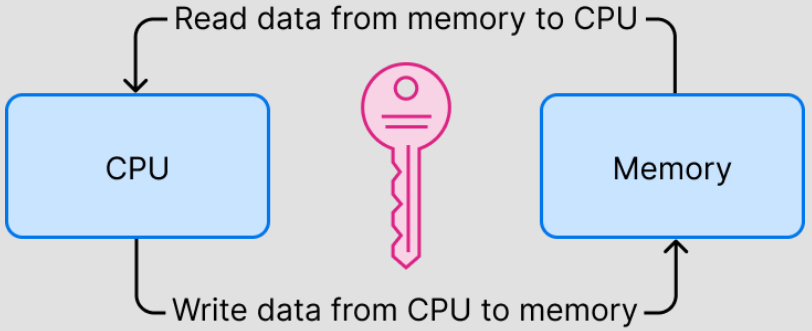


Confidential Containers (CoCo)

Confidential Containers (CoCo)

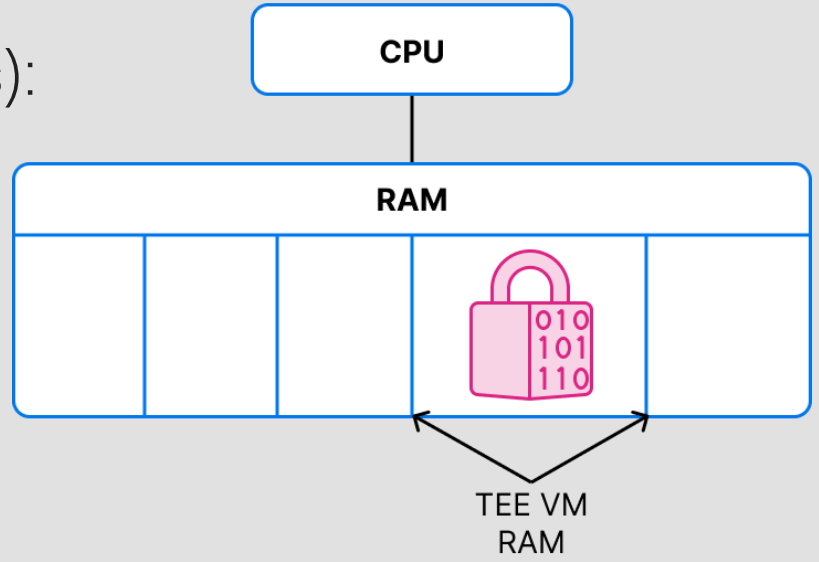
Confidential computing защищает данные в момент ее использования:

- CPU registers
- CPU caches
- RAM



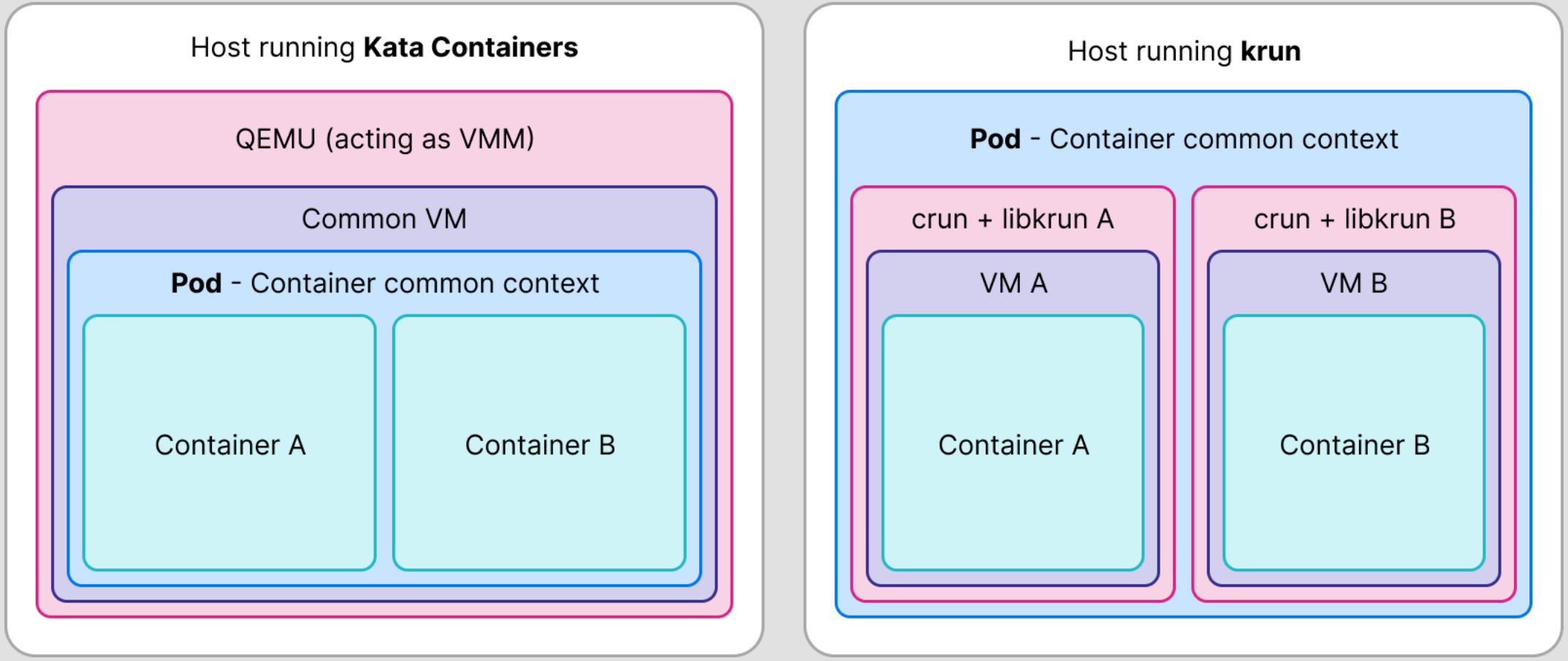
Основано на Trusted Execution Environments (TEEs):

- AMD SEV
- Intel SGX
- Intel TDX



Реализация CoCo

krun Runtime vs. Kata Containers

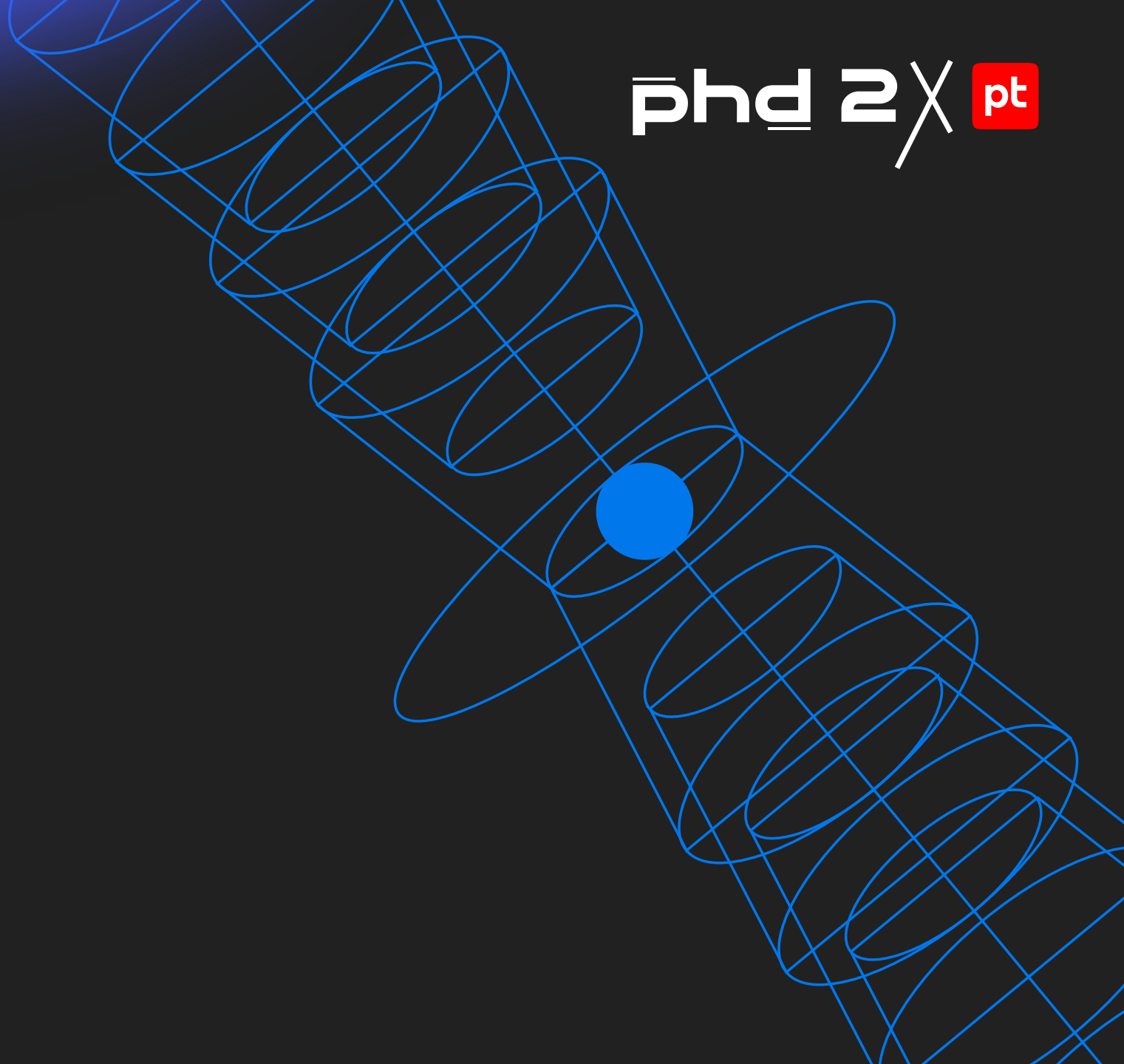


Мысли о CoCo

- Защита от нарушителя с доступом к железу
- Требования к процессору и способу запуска
- Ограничения на возможности приложений
- Необходимость в специально подготовленном образе
  - Можно конвертировать существующие в специальный формат “confidential workload”
- Взаимодействие с attestation server
- Вопрос observability
  - Появление слепых зон в инфраструктуре

03.3

phd 2X pt

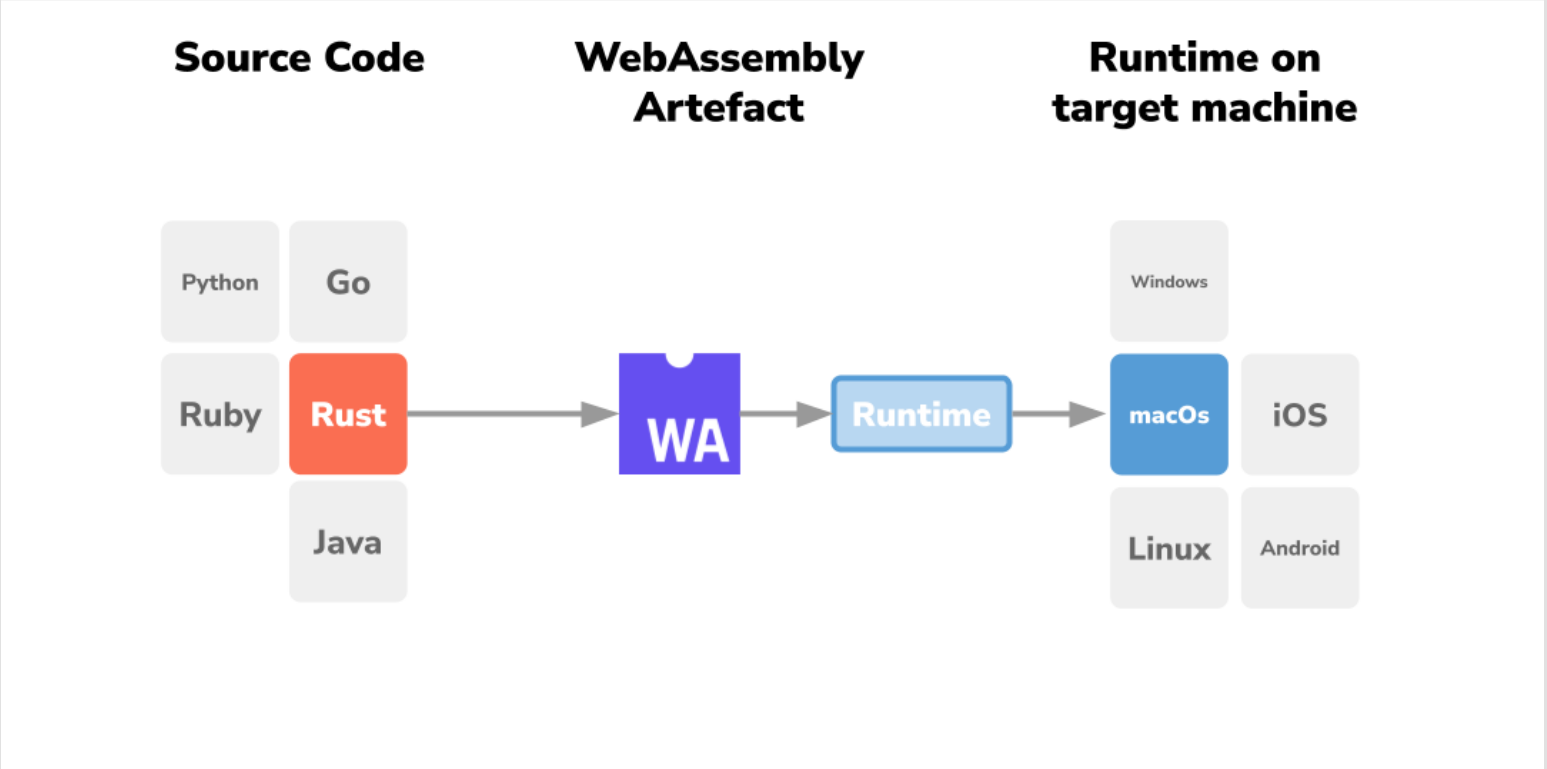


WASM

WASM

- WebAssembly

WebAssembly (abbreviated *Wasm*) is a binary instruction format for a stack-based virtual machine. Wasm is designed as a portable compilation target for programming languages, enabling deployment on the web for client and server applications.



Реализация WASM runtimes

- WasmEdge,
- Wasmtime,
- Wasmer,
- > 30+

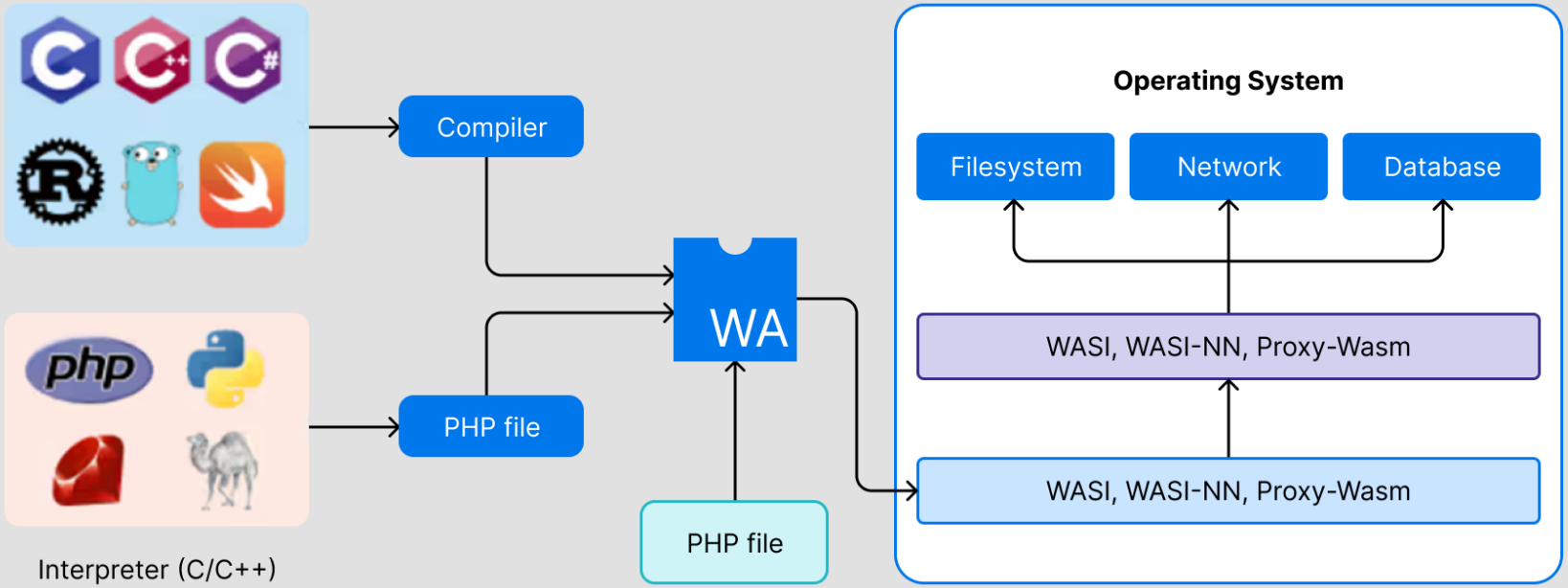
WasmEdgeRuntime

BYTECODE ALLIANCE

Wasmer

WASI и WASIX

- WASI - WebAssembly System/Standart Interface
  - Разрабатывается с 2019
  - Предоставляет низкоуровневое API для реализации POSIX-like слоя
  - Должен быть применим ко всем ОС
  - Много чего еще не реализовано, не стабильно
- WASIX = WASI ABI + дополнительный набор non-invasive syscall



WasmEdge в K8s

WasmEdgeRuntime

1. Замена Kubelet
   - Krustlet
     - Kubelet написанный на Rust
     - Только WASM нагрузка на Node
2. Использовать containerd shim
   - RuntimeClass
     - Только WASM нагрузка в Pod
3. Заменить на свой OCI Runtime
   - crun
     - module.wasm.image/variant или run.oci.handler: wasm-smart annotation
     - Уживается с Sidecars
     - Необходима замена runc
4. Просто встроить Runtime в образ
   - wasmedge/slim-runtime
     - Универсально
5. Как плагин для вашей программы

Kubernetes Use Cases

- Kubernetes + containerd + crun
- Kubernetes + CRI-O + crun
- Kubernetes + Containerd + Runwasi
- OpenYurt
- SuperEdge
- KubeEdge
- Kind
- Knative
- OpenFunction
- Kwasm
- WasmEdge DockerSlim

Взгляд с хоста

wasmedge-shim 0.4.0

WasmEdgeRuntime

```
93163 ?        Sl     0:00 /usr/local/bin/containerd-shim-wasmedge-v1 -namespace k8s.io -id 9043f24a235f751f41c46da7c539
93224 ?        Ss     0:00  \_ /usr/local/bin/containerd-shim-wasmedge-v1 -namespace k8s.io -id 9043f24a235f751f41c46da7
```

```
C:\Users\d.evdokimov\Documents>kubectl.exe --kubeconfig=phd.kubeconfig exec --stdin --tty wasmedge-5c8bc7c9b-trdm7 -- /bin/bash
error: Internal error occurred: error executing command in container: failed to exec in container: failed to create exec "f32c69
 /containerd.task.v2.Task/Exec is not supported: not found
```

Особенности WASI

WASI Design Principles

Capability-based security

WASI is designed with capability-based security principles, using the facilities provided by the Wasm component model. All access to external resources is provided by capabilities.

There are two kinds of capabilities:

- Handles, defined in the component-model type system, dynamically identify and provide access to resources. They are unforgeable, meaning there's no way for an instance to acquire access to a handle other than to have another instance explicitly pass one to it.
- Link-time capabilities, which are functions which require no handle arguments, are used sparingly, in situations where it's not necessary to identify more than one instance of a resource at runtime. Link-time capabilities are *interposable*, so they are still refusable in a capability-based security sense.

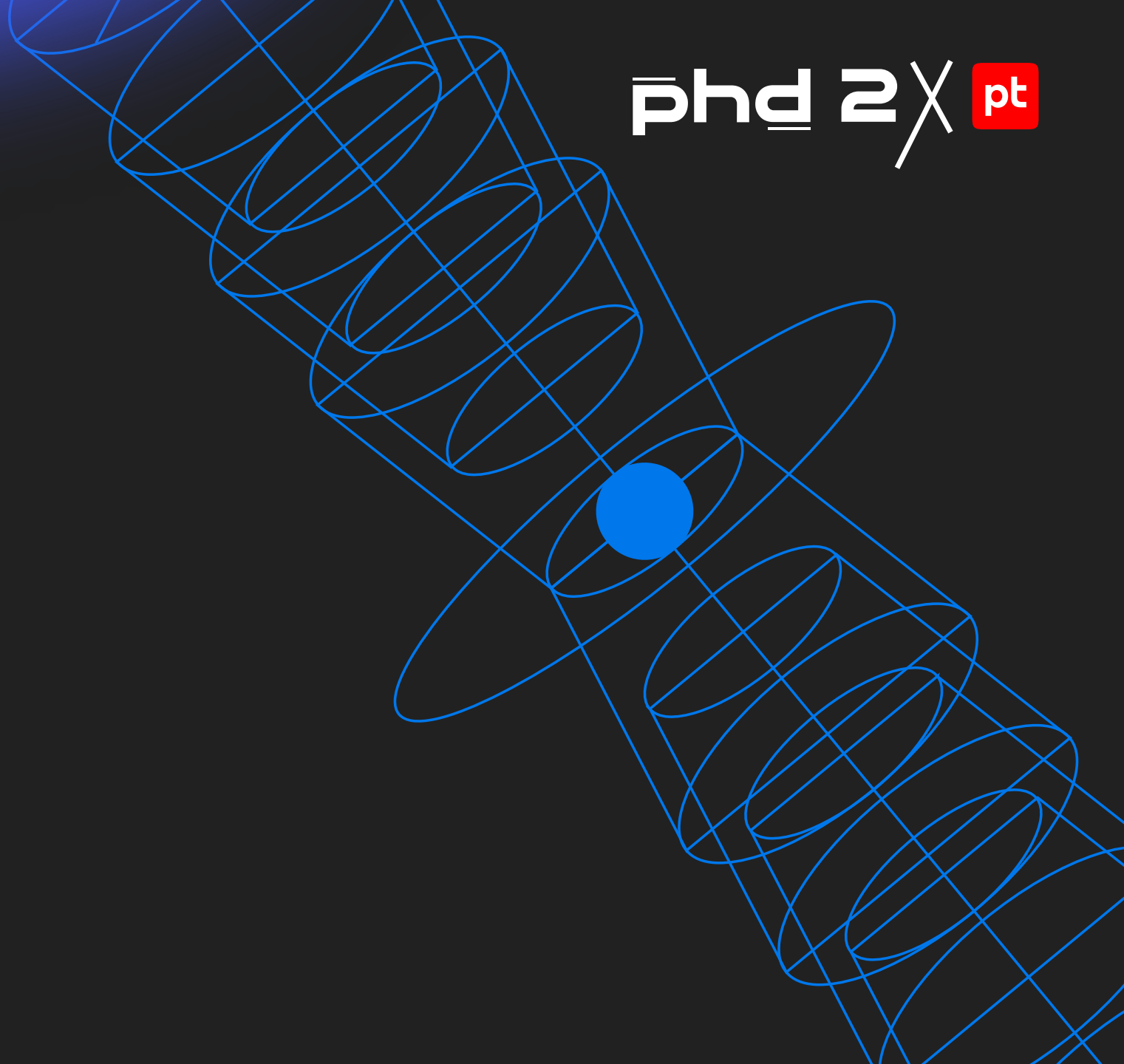
WASI has no *ambient authorities*, meaning that there are no global namespaces at runtime, and no global functions at link time.

Мысли о WASM

- Зоопарк WASM runtimes
- Сырое состояние WASI
- Сложности с инструментами отладки и т.д.
- Сложности сочетания с контейнерами не на WASM
  - Несколько контейнеров в Pod
- Вопрос observability
- Большой потенциал для изоляции и безопасности

03.4

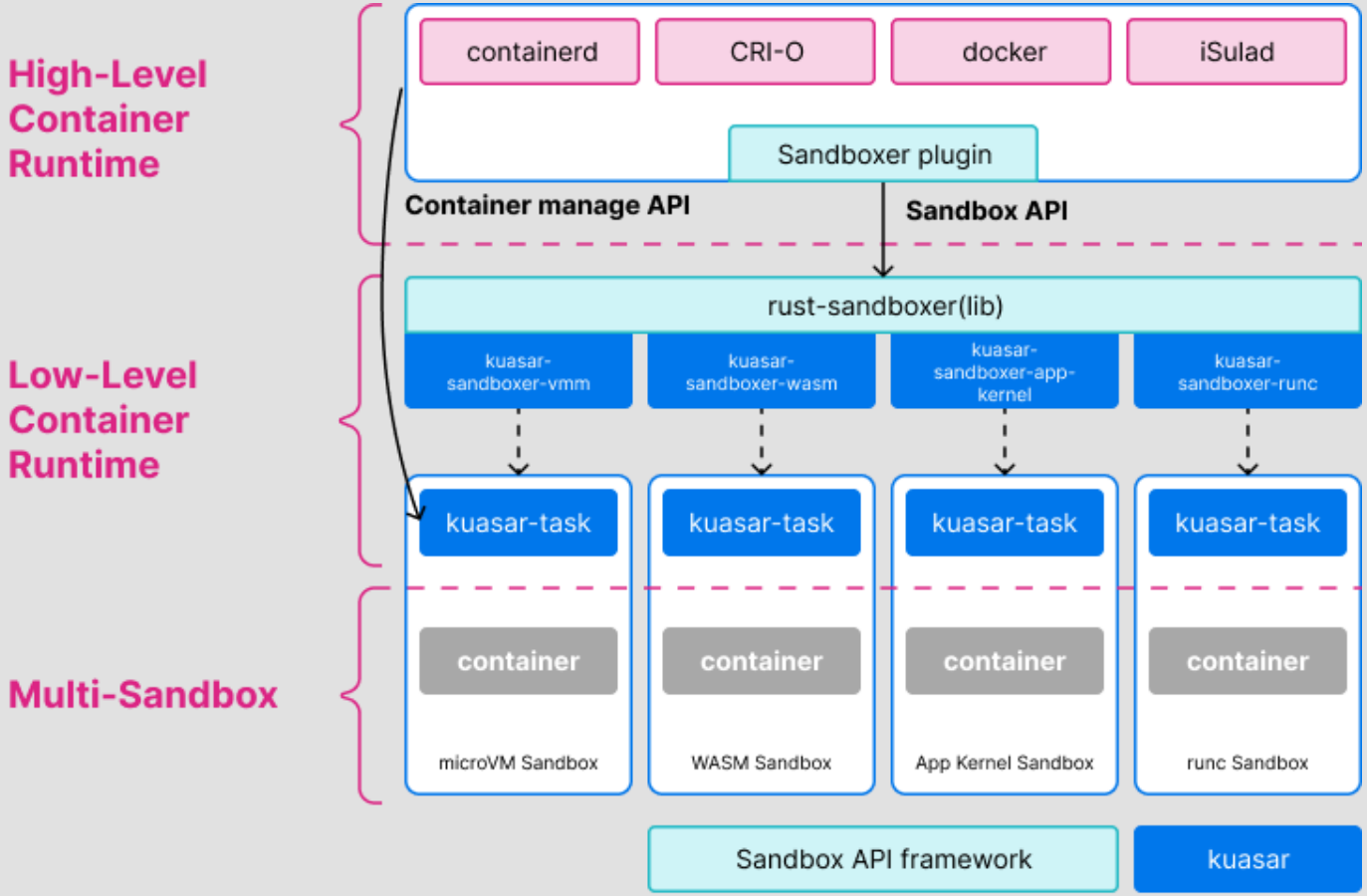
Sandbox API



Sandbox API

Kuasar - An Efficient Multi-Sandbox Container Runtime.

Kuasar = High-level container runtime + sandbox plugins + sandbox API



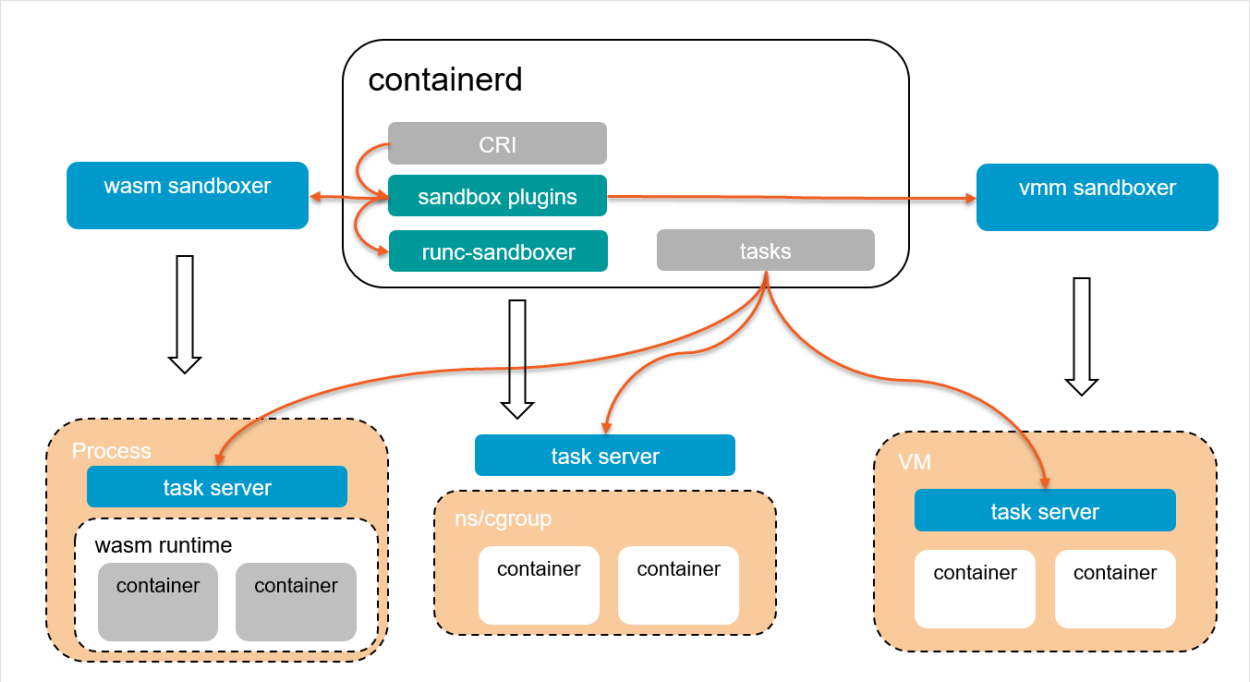
Sandboxer	Sandbox	Status
MicroVM	Cloud Hypervisor	Supported
	QEMU	Supported
	Firecracker	Planned in 2024
	StratoVirt	Supported
Wasm	WasmEdge	Supported
	Wasmtime	Supported
	Wasmer	Planned in 2024
App Kernel	gVisor	Planned in 2024
	Quark	Supported
runC	runC	Supported

Languages

Rust 92.9% Shell 6.0% Other 1.1%

Мысли о Sandbox API

- Благая идея все унифицировать, объединить и понизить порог входа
- Очень сырое текущее состояние
  - Proposal: Sandbox API
    - https://github.com/containerd/containerd/issues/4131
  - [Proposal] Introduce "sandboxer" plugin
    - https://github.com/containerd/containerd/issues/7739



04

phd 2X pt

Заключение

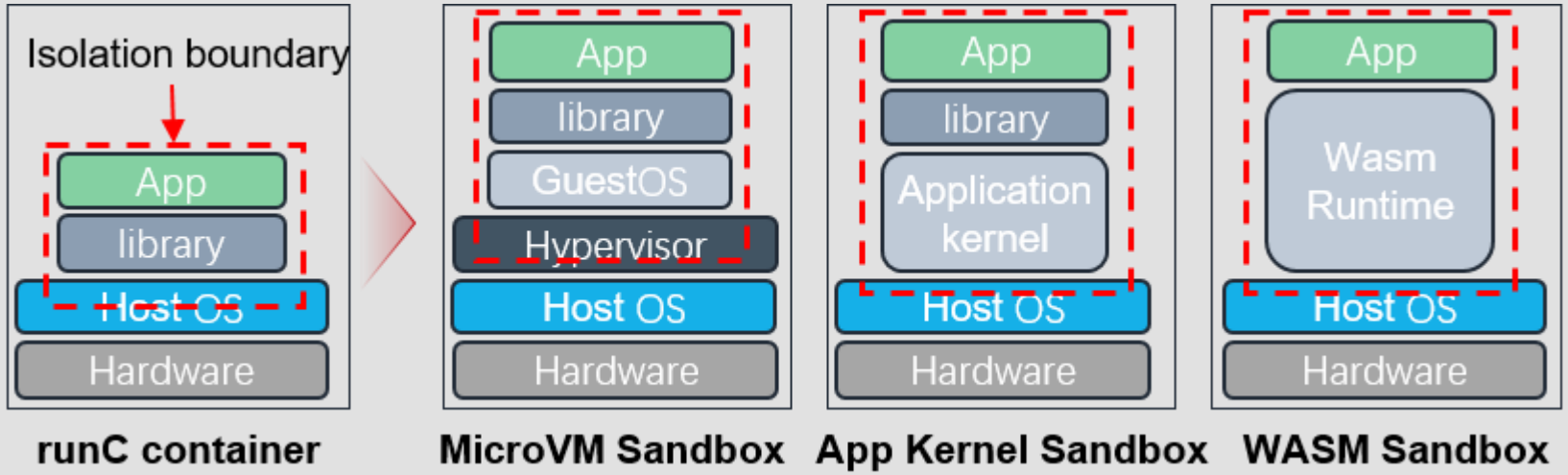
Containers VS Sandbox VS microVM

1. Специфика приложения
2. Требования к hardware
3. Вопрос observability

Sandbox Classification	Advantage dimension			Vendor		
	efficient	secure	General	AWS	GCP	Azure
runC container	√		√	√	√	√
microVM		√	√	firecracker	√	√
App Kernel	√	√			gVisor	
WASM	√	√				wasmtime

Заключение

- Для защиты от побегов из native containers через уязвимости ядра есть множество механизмов
- Высокая изоляция:
  1. Защищает только от 1 вектора побега из контейнера!
  2. Как правило выдвигает требования к железу
  3. Как правило лишает прозрачности происходящего в контейнерах
  4. В некоторых реализациях накладывает ограничения на приложение
  5. В некоторых реализациях для K8s в очень сыром состоянии



Полезные ссылки

1. “The internals and the latest trends of container runtimes”, Akihiro Suda
2. “Container escapes: Kubernetes edition”, Dmitriy Evdokimov
3. “Kata Containers: Security and Containers Without Compromise”, Ildiko Vancsa
4. “Confidential Containers with the Crun-Krun Container Runtime”, Tyler Fanelli


phd 2
Positive Hack Days Fest
от ■ positive technologies
Email: de@luntry.ru
Twitter: @evdokimovds
@Qu3b3c
Channel: @k8security
Site: www.luntry.ru
Спасибо!