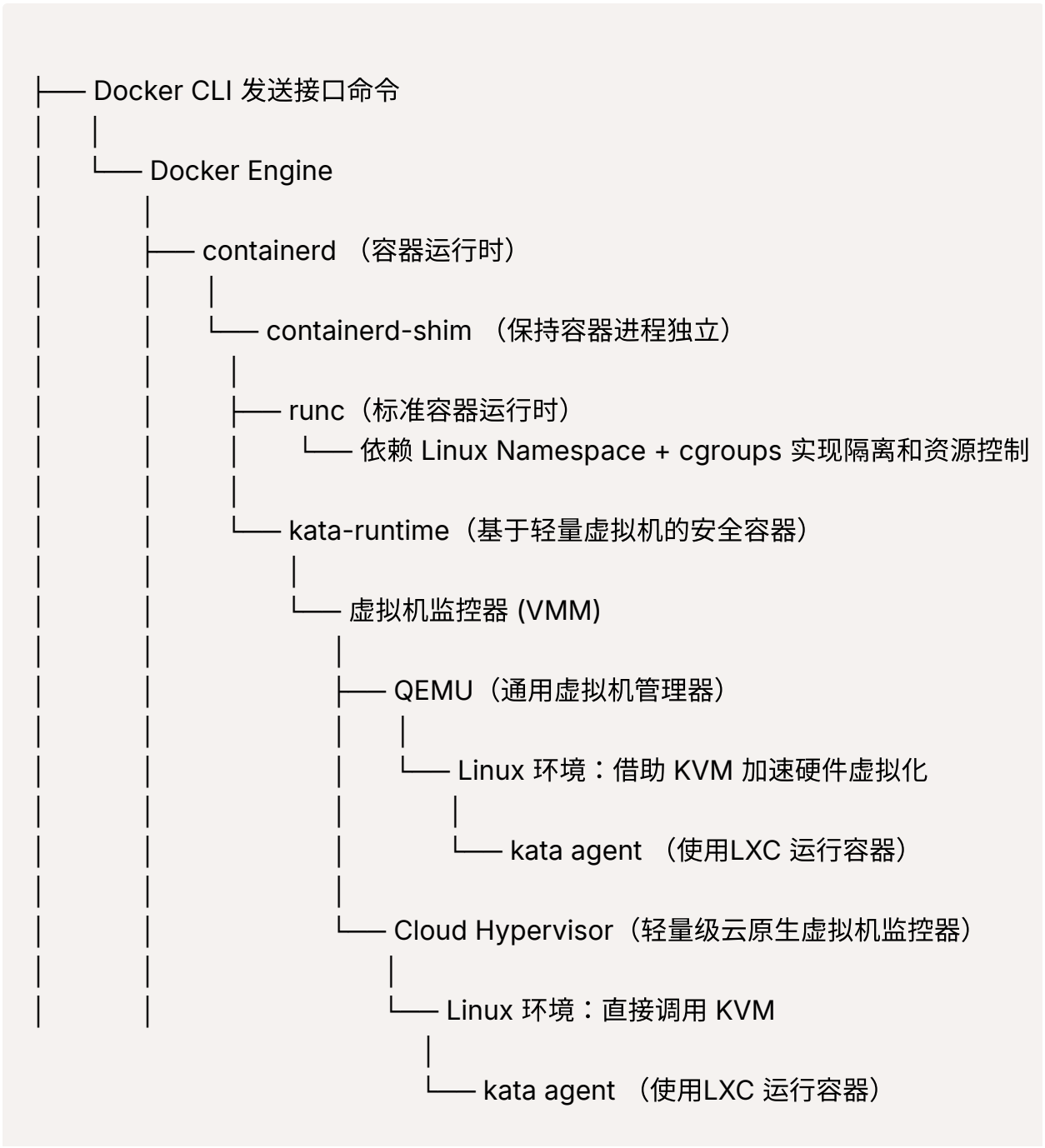
容器化技术与虚拟化技术

技术架构



```
Kubelet
  │
  └── containerd （CRI容器运行时接口）
      │
      └── [同上路径]
```

1. Docker 和 containerd 的关系

- Docker 最初使用的是 **单体架构**，后来将运行时部分抽离成了 `containerd`。
- 现在的 Docker 架构：
  - 构建镜像 → Docker
  - 管理容器生命周期 → containerd
  - 实际运行容器 → 由 containerd 调用 `runc`（或其他兼容的 runtime）

2. containerd 和 kata-container 的关系

- containerd 默认使用 `runc` 启动容器。
- 如果你想获得更强隔离（如在多租户环境），可以配置 `kata-runtime`（Kata Containers 提供）作为 containerd 的 runtime。
- `kata-runtime` 实际是使用虚拟机（QEMU 或 Cloud Hypervisor）来运行容器。

3. Kata Containers 和 QEMU / Cloud Hypervisor

- Kata Containers 本质是将容器运行在轻量虚拟机中。
- 它支持多种 VMM（虚拟机监控器）后端：
  - 默认是 QEMU（兼容性广）
  - 也支持 Cloud Hypervisor（专为云场景优化，更轻量，启动快）

4. QEMU 和 Cloud Hypervisor 的关系

- 都是运行虚拟机的工具：
  - QEMU：功能强大、支持多平台、仿真能力强，但启动稍慢、资源开销大。

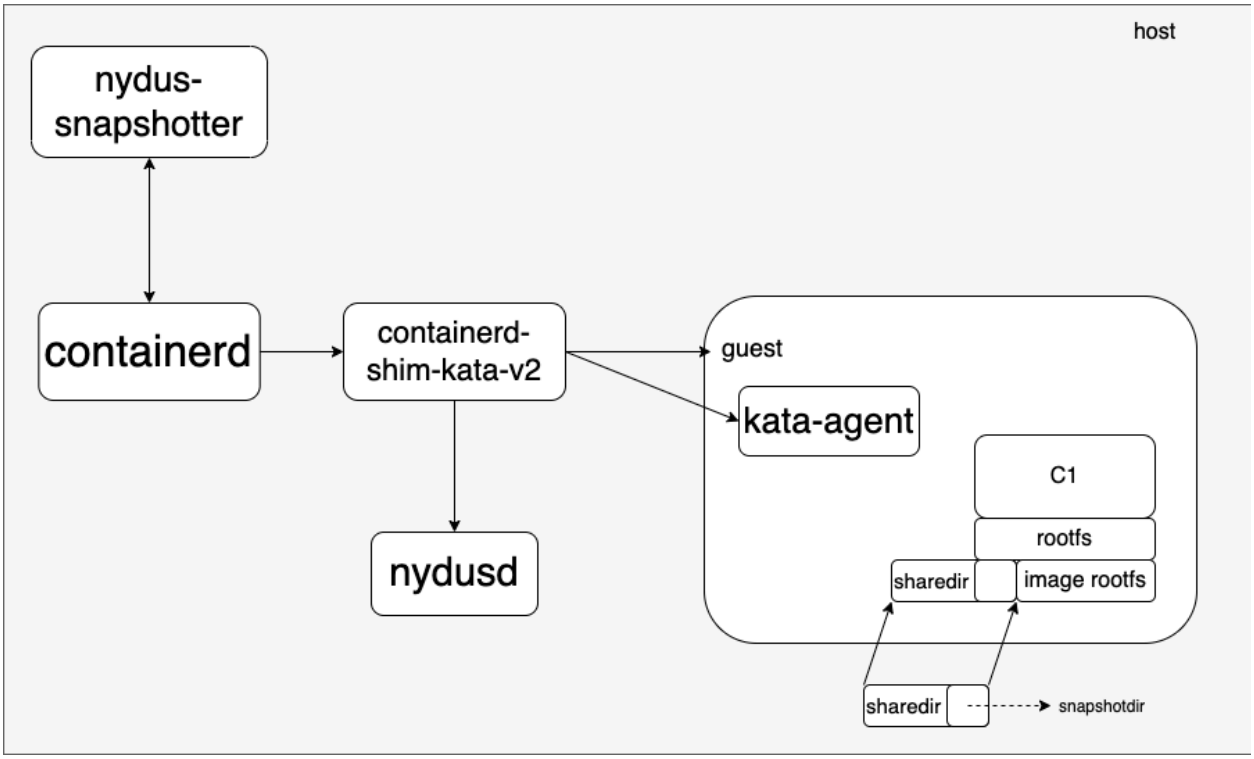
- Cloud Hypervisor：仅支持 x86-64/aarch64 Linux/Windows，针对云原生场景优化，启动更快，占用更少。

5. 基础概念 — Hypervisor（虚拟机监控器）

虚拟机监控器（Hypervisor） 是实现虚拟化的关键软件层，它在硬件和虚拟机之间调度资源。根据部署位置不同分为两类：

- **Type 1（裸机型）Hypervisor**：直接运行在物理硬件上，比如 KVM、MSHV、VMware ESXi、Xen。
- **Type 2（托管型）Hypervisor**：运行在操作系统之上，比如 VirtualBox、VMware Workstation。

Kata Container



Kata Container 项目的本质是在兼容OCI标准的要求下，使用VMM技术，将符合OCI标准的镜像运行在VM中，最终达到使用vm kernel ，对容器实现内核级隔离，并还能保持和容器创建速度接近。

在DBaaS项目中使用Kata容器的难点

1. 排查问题链路变长，当运行在kata container中的程序出现问题时，就必须登陆到vm中进行排查，相比于在物理机上自己排查难度增大。
2. 运行在虚拟机中的性能损耗，和朱立宏老师了解当时使用kata container进行测试，后续放弃该方案的原因，主要就是测试发现运行在kata container中upsql的性能相比运行在物理机docker容器的性能损失最大达到40%，这是主要放弃kata方案的原因。

总结

Kata Container 是一个提供内核级别安全的容器runtime方案，但是引入VM 虽然能提高容器的隔离能力，但是也大大增加了运维和管理的成本，在如此复杂的架构下，我们觉得我们很难驾驭如此复杂的软件。

Cloud Hypervisor

Cloud Hypervisor 是一个专为**现代云原生场景设计的高性能、轻量级虚拟机监视器（VMM）**，其技术定位聚焦于通过精简架构、安全性和效率优化，满足云计算基础设施对敏捷性、资源密度及安全隔离的核心需求。

Cloud Hypervisor作为面向云原生设计的轻量级VMM（虚拟机监视器），在启动速度、资源开销和安全性上具有显著优势，但其技术定位也带来了一些相比QEMU的明显缺失和不足。以下是针对Linux环境（如Rocky Linux/RHEL/openEuler）的详细对比分析：

一、功能完整性缺失

1. **热迁移与快照支持不足**：
   - **实时热迁移**：QEMU支持完整的实时迁移（Live Migration），可在业务无感知状态下迁移虚拟机；Cloud Hypervisor仅提供基础迁移功能，复杂场景（如大内存实例或网络密集型负载）的迁移成功率和稳定性较低13。
   - **快照管理**：QEMU支持完整快照链（增量/差异快照），而Cloud Hypervisor的快照功能处于“部分支持”状态，缺乏高级回滚和一致性保障机制1。

2. **高级设备管理能力弱**：
    - **设备热插拔**：虽然支持基础设备热插拔，但缺乏QEMU的精细化控制（如PCIe设备动态重配）。
    - **传统硬件兼容性**：仅支持`virtio`半虚拟化设备（如virtio-blk/net），无法模拟传统硬件（如IDE硬盘、e1000网卡），导致旧版OS或专用设备驱动兼容性差15。

🖥️ 二、硬件兼容性与生态局限性

1. **异构架构支持不足**：
    - QEMU支持全CPU架构模拟（x86、ARM、PowerPC等），而Cloud Hypervisor仅优化支持**x86_64**，ARMv8处于实验阶段，其他架构（如RISC-V）几乎无支持46。
    - 在国产化场景（如openEuler+昇腾/RISC-V）中，QEMU已被深度集成，Cloud Hypervisor需额外适配3。
2. **GPU/AI加速支持薄弱**：
    - QEMU成熟支持**GPU直通**（vGPU/NVIDIA GRID）和虚拟化方案（如vGPU分片）；Cloud Hypervisor虽支持基础直通，但缺乏虚拟化分片和AI框架（如CUDA）的深度优化310。

🛠️ 三、管理与调试工具链不完善

1. **运维工具生态差距**：
    - **图形化管理**：QEMU可通过`virt-manager`提供完整GUI，Cloud Hypervisor仅有基础CLI工具（`ch-remote`），依赖第三方集成1。
    - **监控与诊断**：QEMU支持`libvirt`+Prometheus等全链路监控，Cloud Hypervisor的监控接口（如Metrics API）功能简陋，缺乏性能剖析工具13。
2. **调试能力不足**：
    - QEMU提供完整的设备模拟调试、内存泄漏检测和QEMU Trace跟踪工具；Cloud Hypervisor仅支持基础GDB，缺乏深度调试能力，问题定位效率低16。

📦 四、生产环境成熟度与稳定性短板

1. **企业级特性缺失**：

- **高可用保障**：QEMU支持集群化部署（如OpenStack Nova调度），配合热迁移实现故障恢复；Cloud Hypervisor无原生HA方案，依赖上层编排35。
- **热升级能力**：QEMU需重启或迁移VM以更新Hypervisor；Cloud Hypervisor虽支持模块热替换，但实际生产中稳定性未经验证29。

2. **资源弹性限制**：
   - 大内存VM启动优化不足：QEMU通过**并发内存清零**可将700GB内存VM启动时间从270秒降至22秒；Cloud Hypervisor仍需完整预分配内存，启动延迟显著。

总结

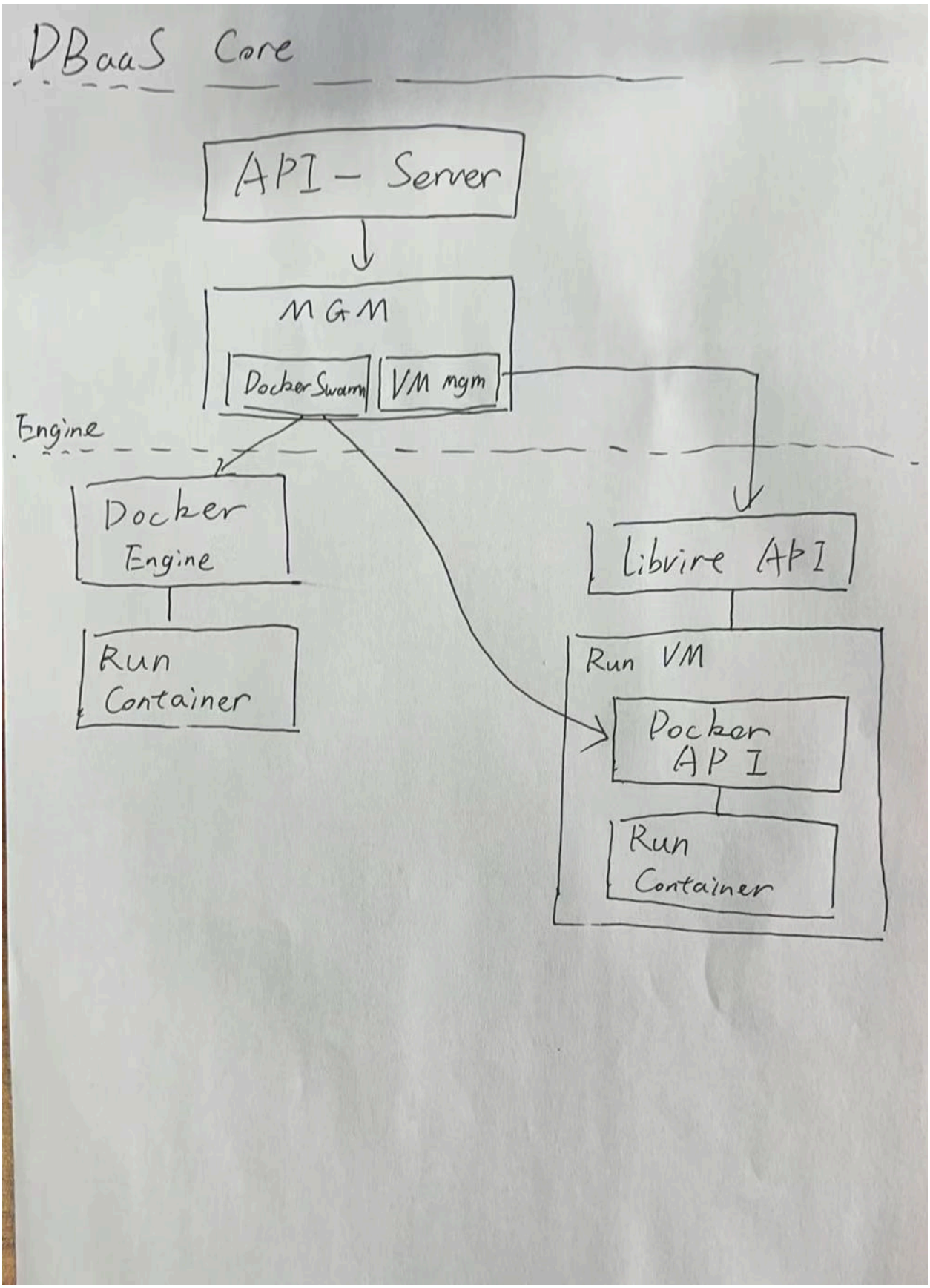
Cloud Hypervisor是
云原生轻量化场景的革新者，他的使用场景主要是秒级启动、低资源开销的简单的无状态服务容器场景，面对数据库这样对启动要求不高，资源开销较大的场景，并不时候，并且他还缺失如VM热迁移，完整的VM快照管理等重要的运维功能。

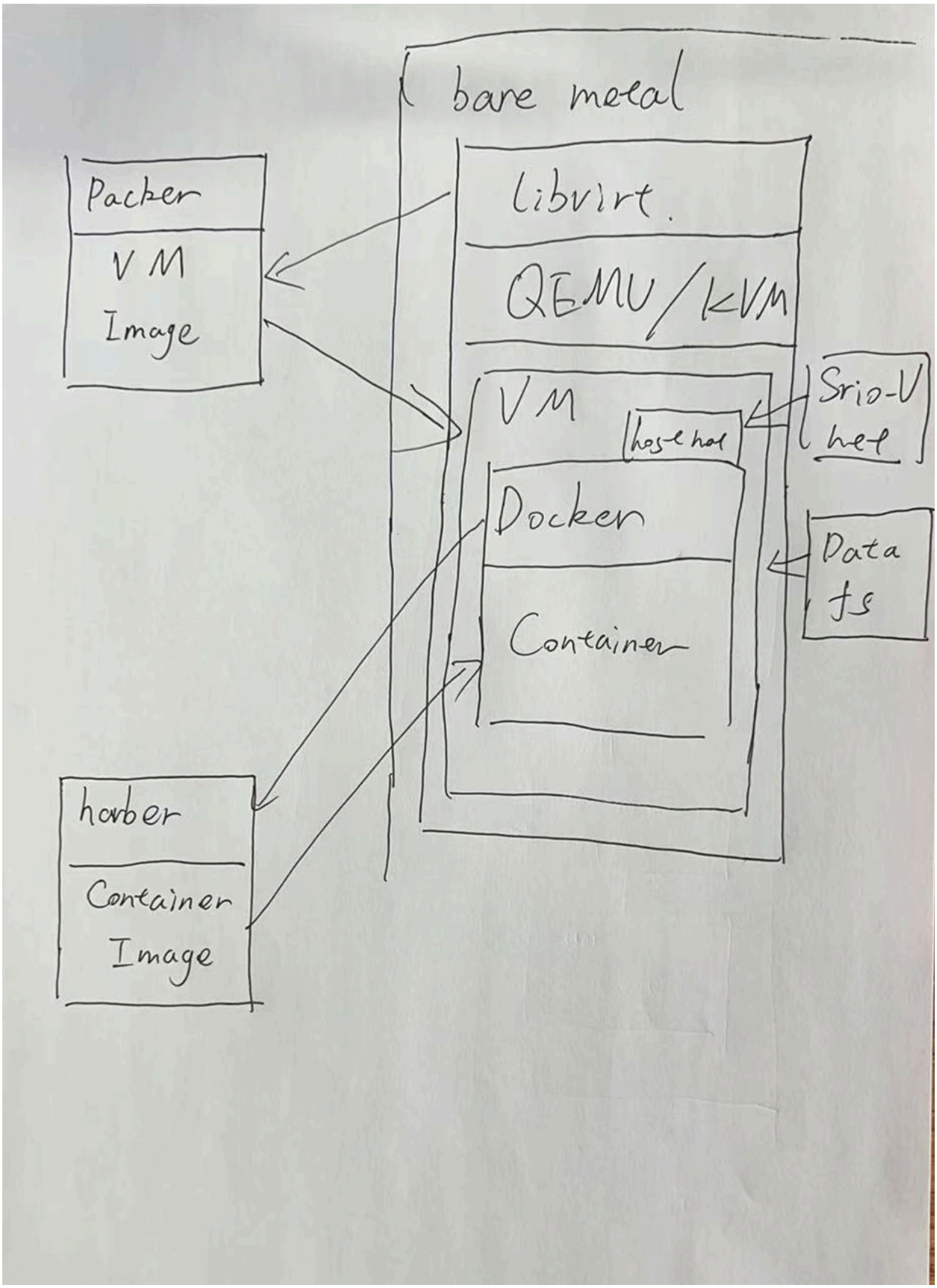
研究后思考

回到需求本身，我们希望在DBaaS项目中有所变化，找到能够引起客户继续投资DBaaS的技术点。由kata container 和Cloud Hypervisor项目启发，并结合现有的DBaaS 设计模型，我觉得可以通过将DBaaS平台引入扩展 VM 管理能力为技术点，解决虚拟机的内核级隔离和强大的VM技术（热迁移，快照管理），实现新的DBaaS平台。

新架构中会引入对于物理机管理VM的功能，主要通过LibVirt实现，底层使用QEMU/KVM实现。并在VM 中运行Docker ，并纳入现有MGM的管理范围，并标记是虚拟机，运行指定范围的容器，以满足安全容器的需求，在物理机上实现虚拟机运行容器和直接运行容器，多重运行方式。并且后续还可以使用VM技术中的热迁移及快照技术，进一步加强运维能力。

新架构


DBaaS Core
API - Server
MGM
Docker Swarm
VM mgm
Engine
Docker Engine
Run Container
Libvirt API
Run VM
Docker API
Run Container


bare metal
Packer
VM
Image
Libvirt
QEMU/KVM
VM
Srio-V
net
Docker
Data
fs
Container
harber
Container
Image