



# KEK電子陽電子入射器におけるPodmanを用いたArchiver Applianceの導入

○佐武 いつか, 佐藤 政則, 王 迪, 草野 史郎<sup>A</sup>, 工藤 拓弥<sup>A</sup>, 櫻井 雅哉<sup>B</sup>  
高エネルギー加速器研究機構, <sup>A</sup>三菱電機システムサービス株式会社, <sup>B</sup>関東情報サービス株式会社

## 概要

- KEK入射器では、データ取得ソフトウェアとして Archiver Appliance(AA)を採用し、17万件以上の制御対象を効率的に収集・保存している。
- 今回、システムの再現性向上、導入作業の簡素化、運用効率の向上を目的として、AAをPodmanによるコンテナ化に取り組んだ。
- 本発表では、これまでの仮想マシン環境での運用経験をもとに、コンテナ環境でAAを構築した際に得られた知見について報告する。

## 仮想マシン構築と管理ツール

- AAコンテナは、Proxmox VE上に構築した仮想マシン上(Rocky Linux 9.5)で稼働している。仮想マシンへのリモート操作にはSSH クライアントとして MobaXtermを用いた。

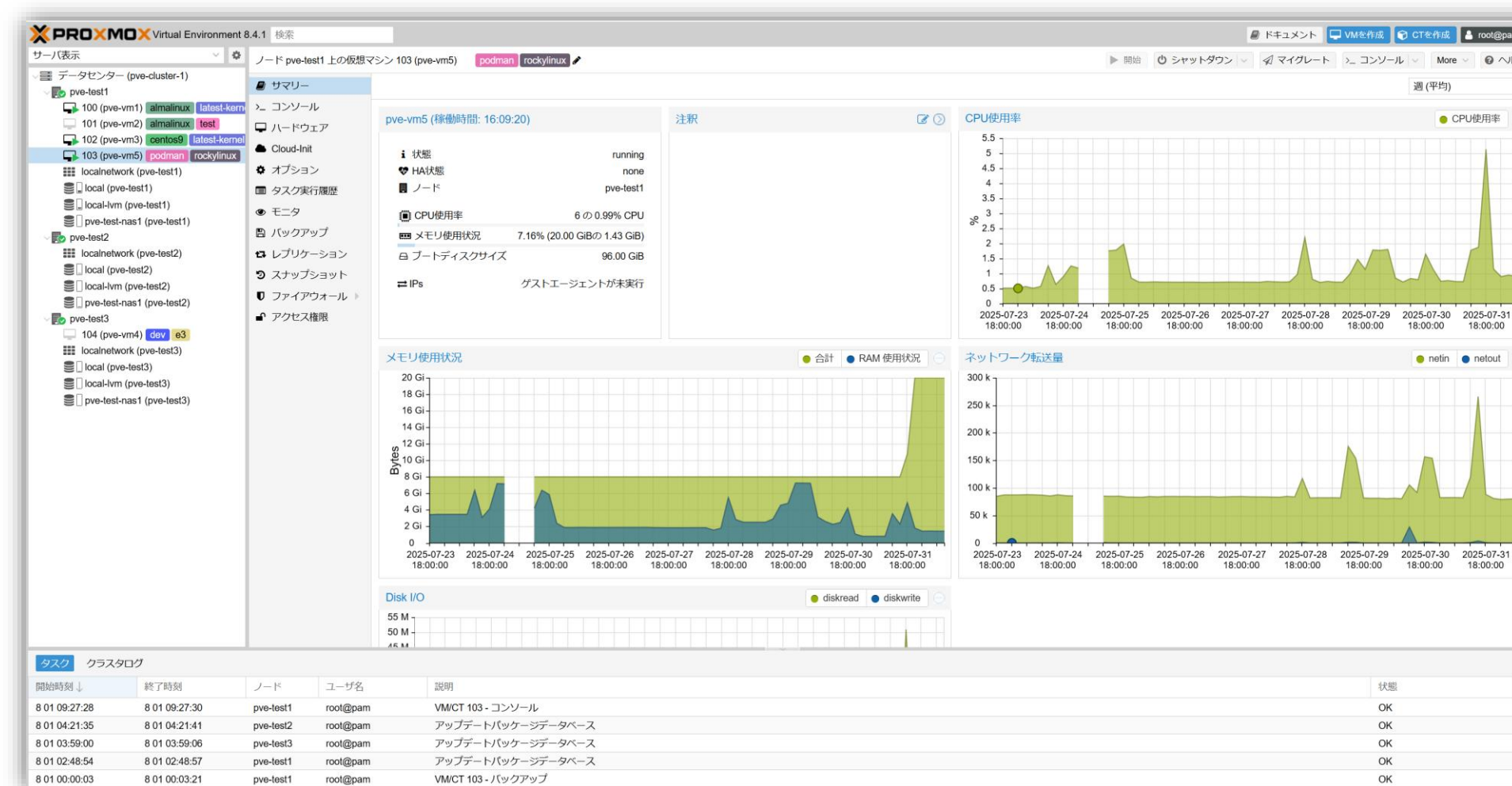


Figure 1: Summary view of virtual machines on Proxmox VE.

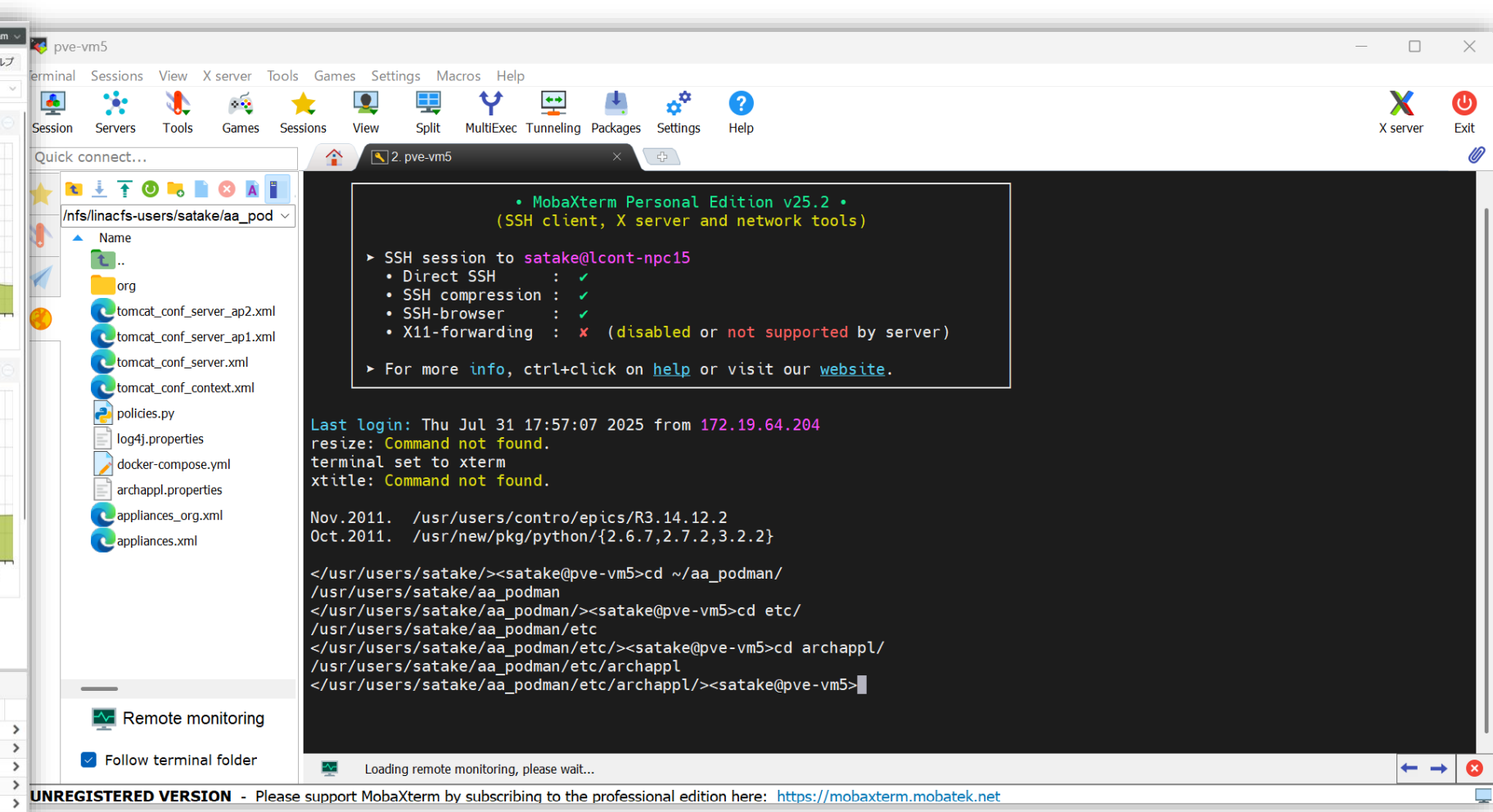


Figure 2: Remote Access to the VM Environment with MobaXterm.

## AAコンテナの構成

### Rootless Podman 実行

- rootless Podman が必要とするファイル操作 (chownなど) が NFS 上では許可されない。
- rootless Podmanが使用する仮想的なUID/GIDの情報はNFSサーバ側では認識されず、操作が拒否される。

➡ Podman が使用する実体データ (コンテナイメージ・一時領域) をローカルディスク上に退避し、rootless 実行を実現した。

### NFS 環境での rootless 実行対応手順

- 1) サブUID/GID範囲の割り当て
- 2) ローカルディスク上にディレクトリ作成
- 3) Podman のストレージ設定ファイル作成

### ネットワーク構成とポート管理

- **--network host** を指定し、コンテナがホストと同一のネットワークスタックを使用する。
- 複数のAAインスタンスを同一ホスト上で運用する場合、異なるポートを割り当てる。Tomcatの設定ファイル (server.xml) をAAコンテナごとに用意し、-v オプションでマウントして切り替えることで、Dockerfileは共通のまま構成管理をおこなう。

ユーザー名前空間：プロセスごとに異なるUID/GIDの対応関係を持てる仕組み

Bridgeモード：コンテナがホストのネットワークとは分離。CAブロードキャストがコンテナに届かない。

Table 1: Configuration of AA Container

Base OS	Debian Bookworm Slim
Application Environment	tomcat:9.0.80-jdk21-openjdk-slim-bullseye
AA	2.0.10
Java	OpenJDK 21
Web Server	Apache Tomcat 9.0.80
Database	MySQL 8.4.5 (Connector-J 9.3.0)

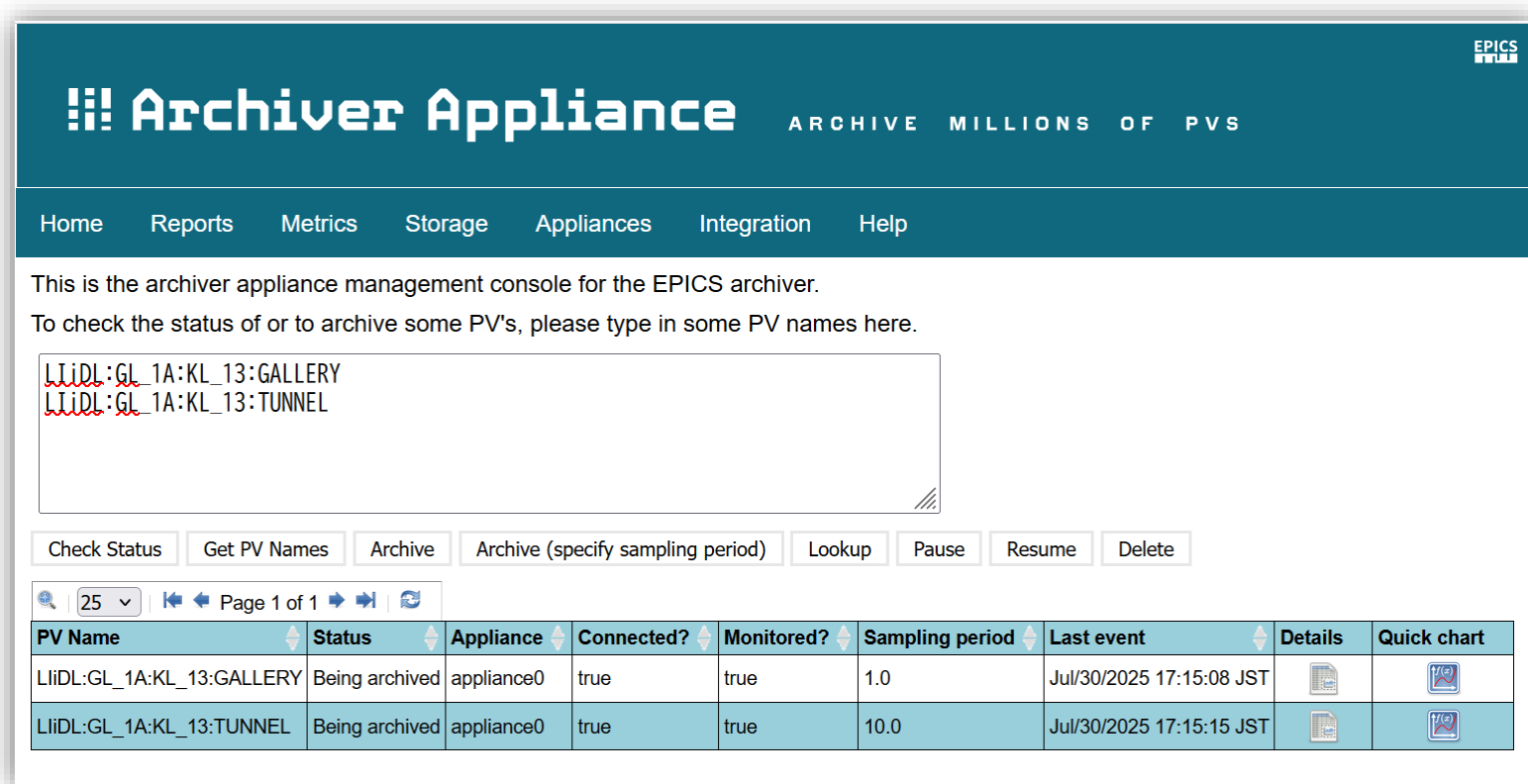


Figure 3: Screenshot of management GUI interface of AA container.

- MySQLコンテナ：Quadletを用いた systemd による管理
- AAコンテナ：独自に作成した systemd サービスユニット

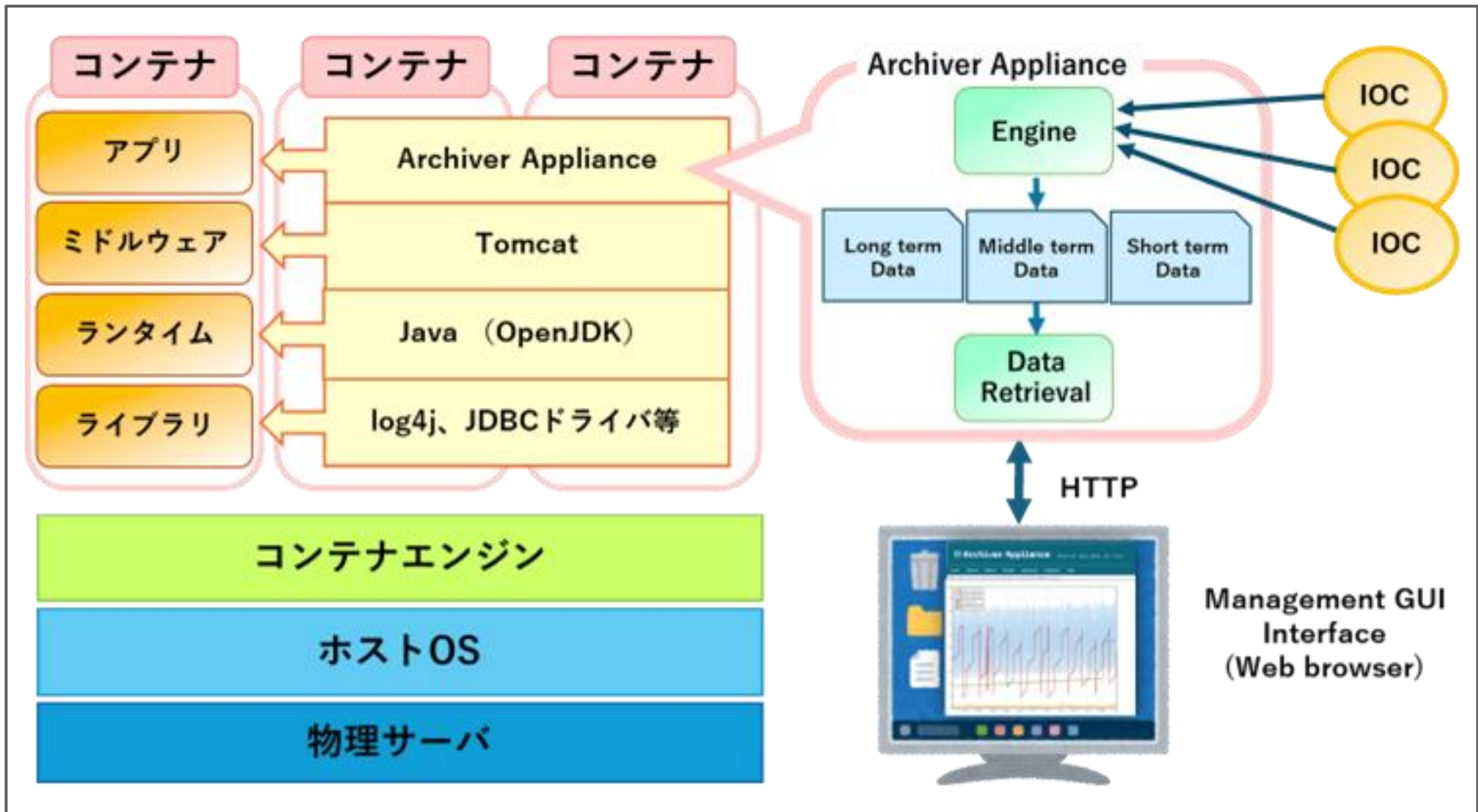


Figure 4: Overview of the AA container system architecture.

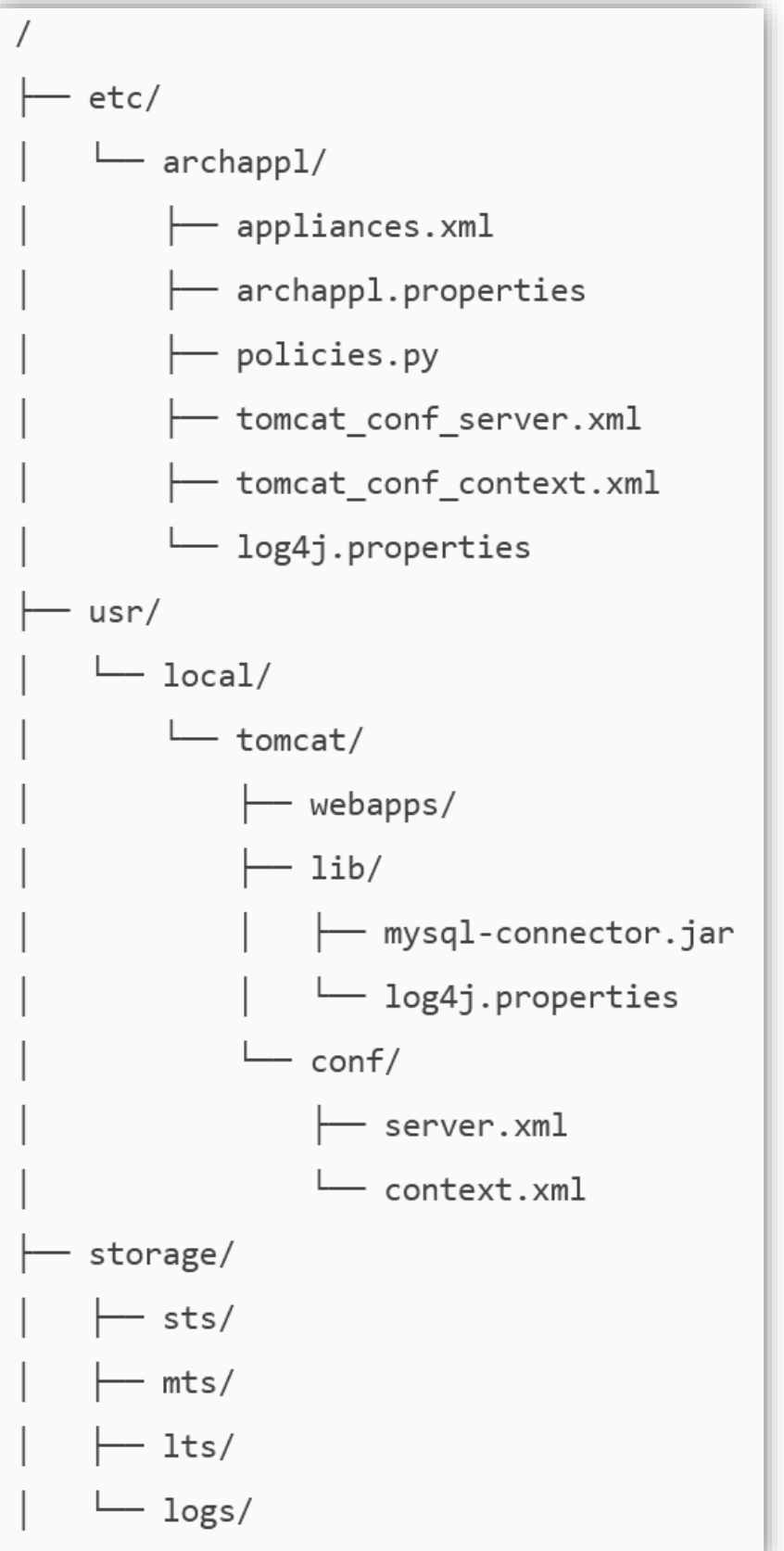


Figure 5: Internal directory layout of AA container

## 従来との比較

Table 2: Comparison of System Architectures: Physical, VM, and Container

	物理サーバー	仮想マシン (VM)	コンテナ
構築の容易さ	ソフトウェア依存関係の管理が煩雑	ゲストOSごと構築、比較的容易	Dockerfileにより迅速かつ一貫した構築が可能
環境の再現性	手動設定が多く再現が困難	スナップショット等で再現可能	イメージから高い再現性を確保
リソース効率	高性能だがリソースの有効活用が難しい	OS単位のオーバーヘッドが大きい	軽量でホストリソースを効率的に使用可能
起動時間	BIOS・OS起動含め時間を要する	OS起動が必要	数秒以内で即時起動可能
可搬性	ハードウェアに依存	ハイパーバイザ依存	OS非依存で高い可搬性
運用の柔軟性	構成変更には停止や手動作業が必要	VM単位の操作が可能	コンテナ単位で柔軟に変更・再構築可能
拡張性	ハードウェア制約が大きい	VM増設で拡張可能	軽量のインスタンスを容易に増減でき、オーケストレーションによる自動拡張も可能



Figure 6: Layered architecture comparison of virtual machine and container.

- ✓ **コンテナ**はホストOSのカーネルを共有しながらアプリケーションを個別の環境として隔離して動作させるため、**より軽量で高速な展開**が可能。
- ✓ Dockerfileを用いた環境定義により、**OSバージョンや依存ライブラリの差異に左右されず一貫性のある構築**が可能である。

## 課題と展望

- KEK入射器の制御システムにおいて、AAのコンテナ化を実施し、Docker のマルチステージビルドを活用して軽量・高セキュリティなコンテナイメージを構築した。
- MySQL との連携や環境変数による動的設定も導入し、AAが正常に動作することを確認した。
- 従来の物理サーバや仮想マシンによる構築と比較して、迅速な環境構築、高い可搬性、柔軟な運用といった利点が明確に示された。
- 今後は、加速器運転中の実負荷を想定したメモリやCPUリソースの最適化、Zabbixによる監視体制の整備に加え、Kubernetesなどによる運用自動化・スケラビリティ向上についても検討を進める。