Kontrollsystem für HAMSTER

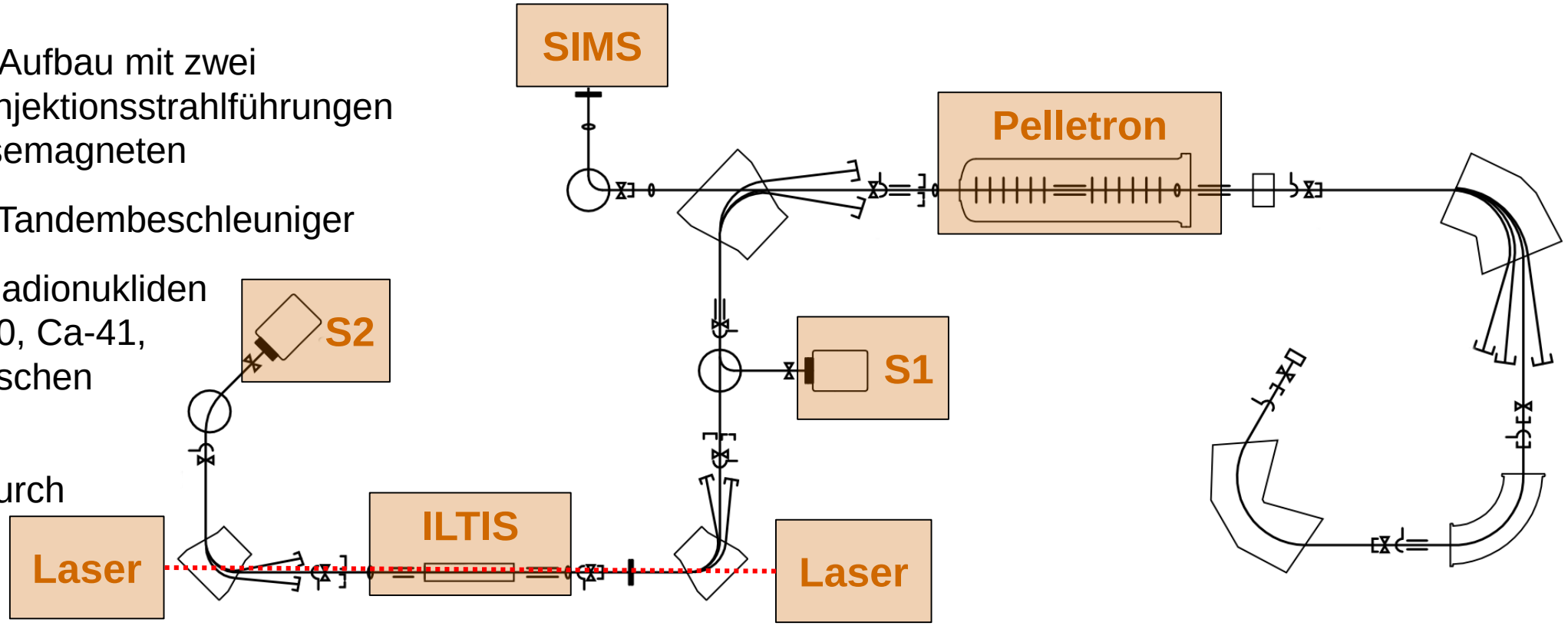
Umsetzung des übergeordneten Kontrollsystems für eine neue AMS-Anlage basierend auf EPICS und quelloffenen Tools in Docker Containern.

SEI 2025

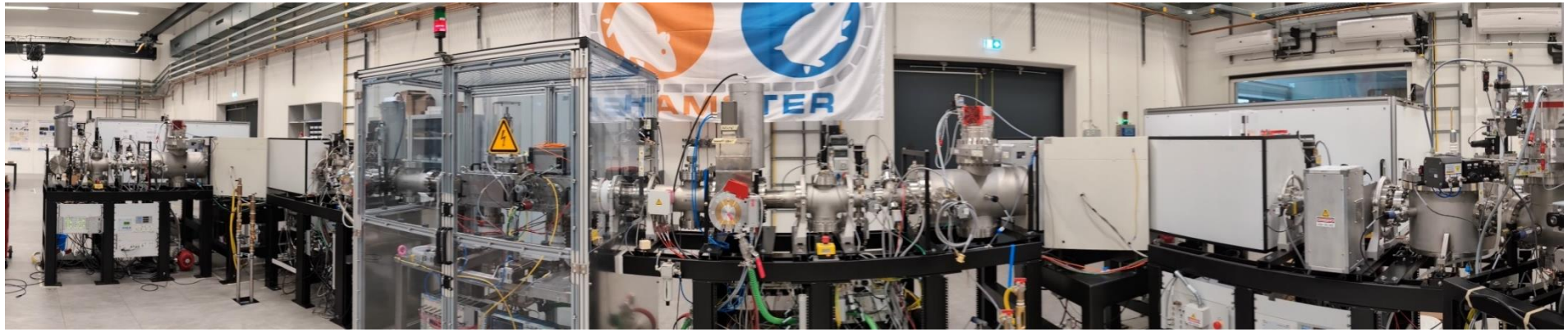
M. Meyer

Helmholtz Accelerator Mass Spectrometer Tracing Environmental Radionuclides

- Typischer AMS-Aufbau mit zwei Niederenergie-Injektionsstrahlführungen und zwei Analysemagneten
- 1-MV Pelletron-Tandembeschleuniger
- Detektion von Radionukliden z.B. Al-26, Be-10, Ca-41, Cl-36 in geologischen Proben
- Bereitstellung durch NEC Corp.

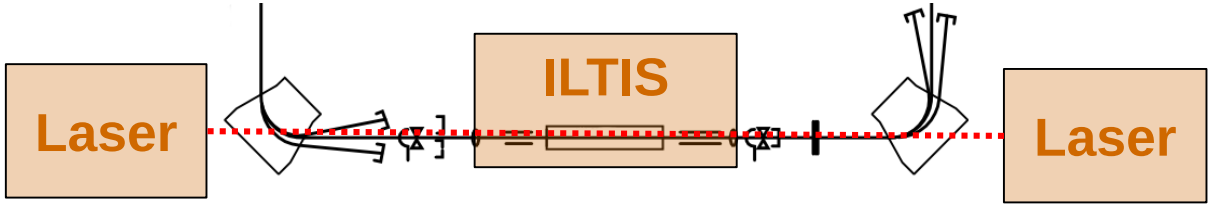


- Integration:
  - Secondary Ion Mass Spectrometry (SIMS)
  - ILTIS
  - Sicherheitssystem

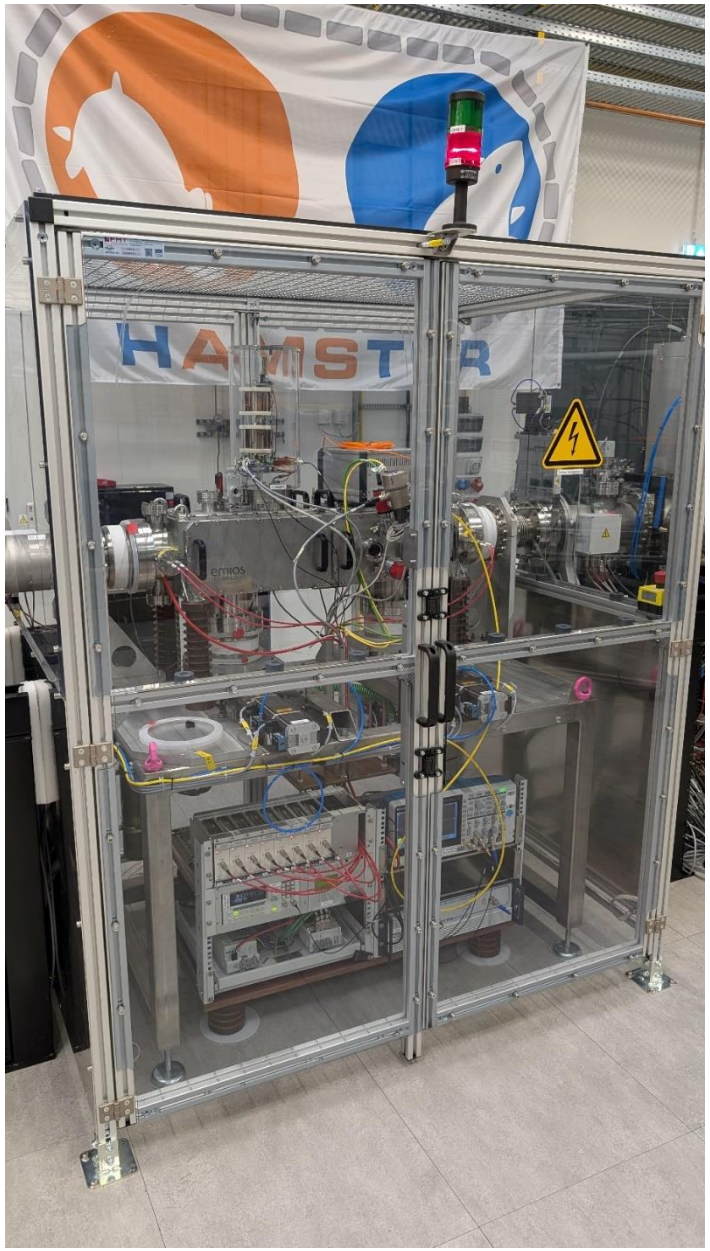


Ion Linear Trap for Isobar Suppression (ILTIS)

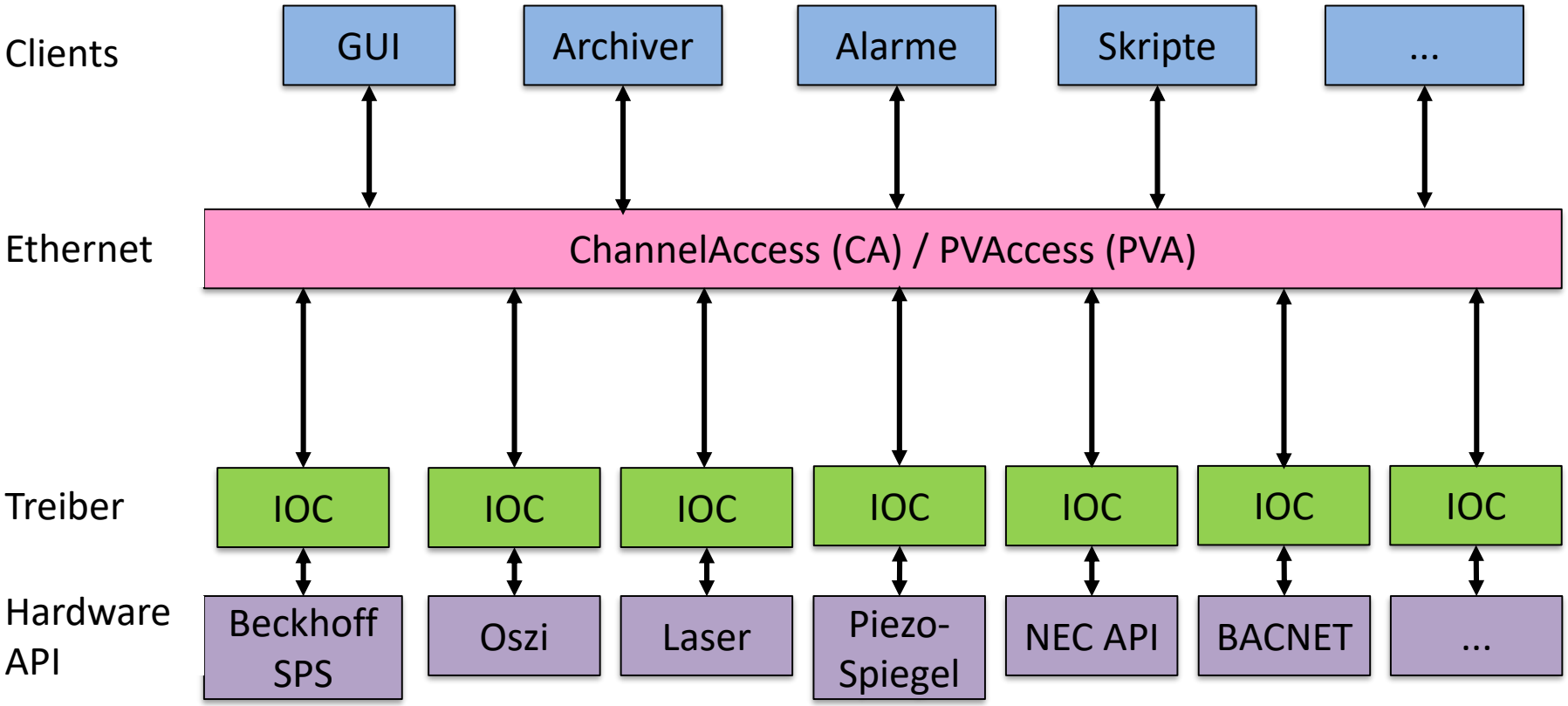
- **Ziel**: Unterdrückung von Isobaren des nachzuweisenden Isotops
- **Methode**:
  - Abbremsen der Ionen für maximale Wechselwirkungsdauer (durch HV und Puffergas)
  - Ladungsneutralisierung der Isobaren-Ionen durch ein Photon mit gleich oder größerer Energie der Elektronenaffinität des Ions (Laser-Photodetachment)



- **Herausforderungen:**
  - Stark begrenzter Raum für Elektronik
  - Netzwerk und RS232-Verbindungen werden auf POF-Leitung getunnelt
  - Personensicherheitssystem für Hochspannung und Laser
  - Einheitliches Nutzer-Interface für ILTIS, Laser, Spiegelmotoren, ...
  - Begrenzte Personalressourcen



Experimental Physics and Industrial Control System (EPICS)



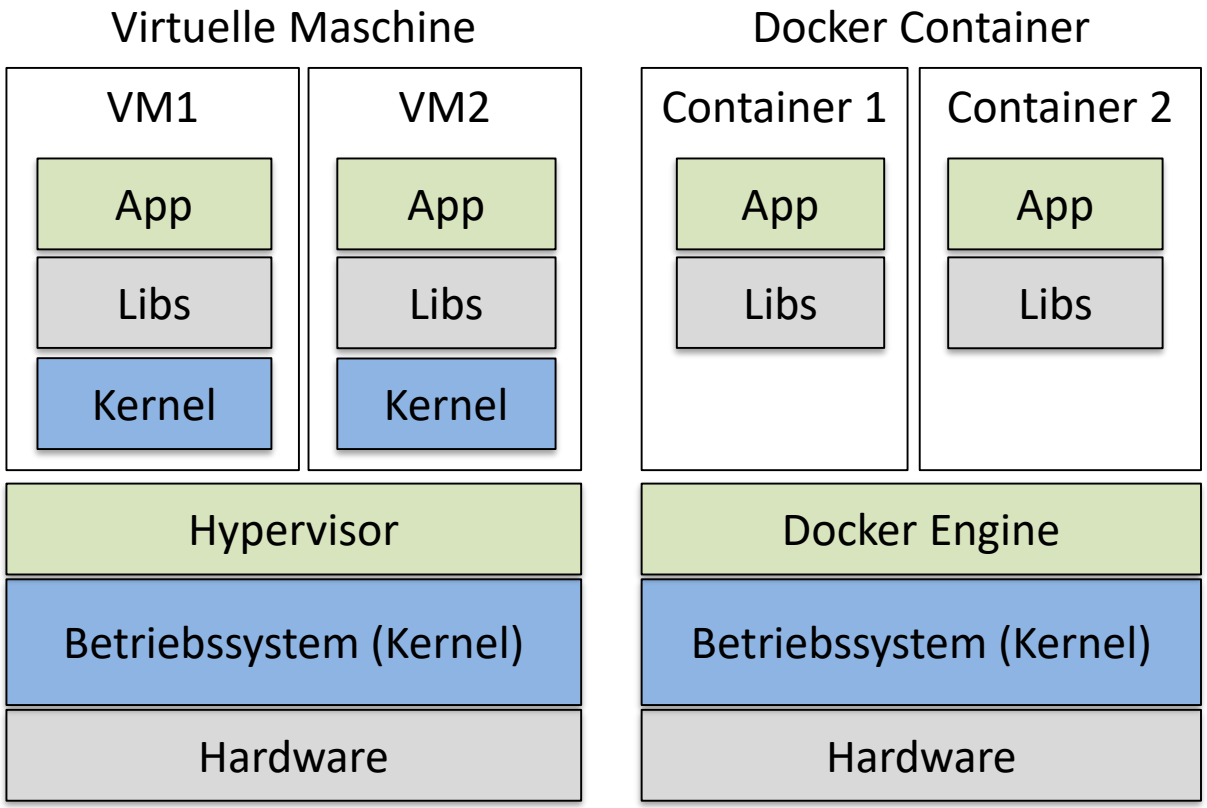
- Verteiltes Kontrollsystem
- Jeder kann mit jedem kommunizieren
- Open Source
- Flexibel

- Fokus auf Linux
- Nicht deterministisch
- Nicht zertifiziert
- Keine Cyber-Security

→ Viele verschiedene Tools mit eigenen Abhängigkeiten und Schnittstellen

Verwendung von Docker Containern

- Erstellung von Docker-Images für alle benötigten (Mikro-)Dienste, Datenbanken und IOCs
- Verwendung des Helmholtz-Gitlab als Container-Registry und Versionsverwaltung

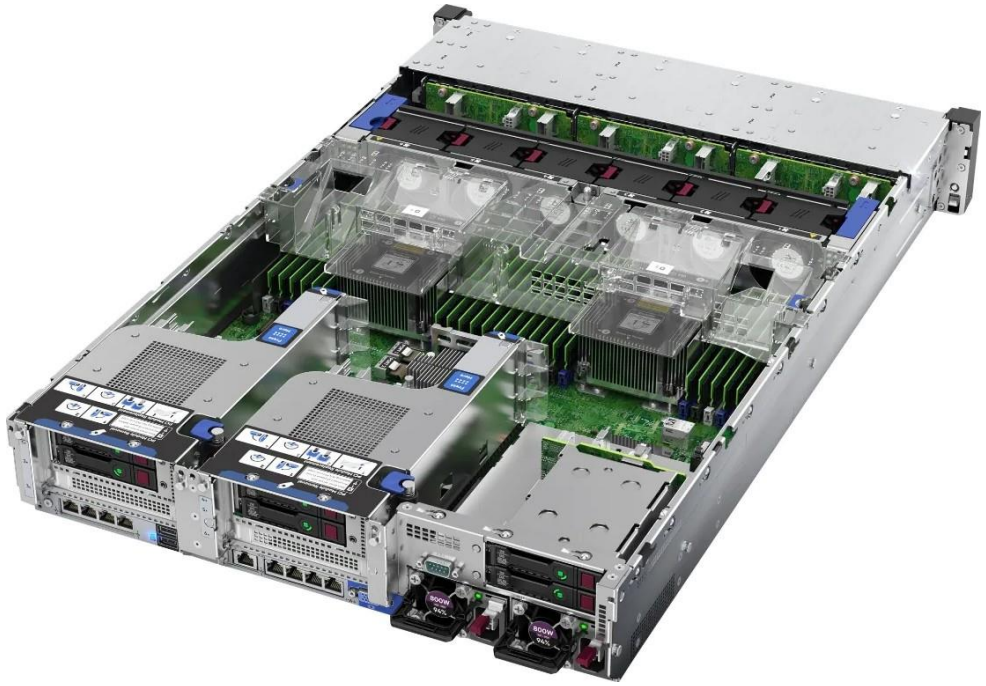


Docker Image

```
1  FROM maven:3.9.6-eclipse-temurin-17 AS build
2  ENV DEBIAN_FRONTEND=noninteractive
3  RUN apt-get update && apt-get install git -y --no-install-recom
4  WORKDIR /phoebus-olog
5  ARG OLOG_VERSION=v5.0.2
6  RUN git clone --depth 1 --branch=${OLOG_VERSION} https://github
7  RUN mvn clean install \
8      -DskipTests=true \
9      -Dmaven.javadoc.skip=true \
10     -Dmaven.source.skip=true \
11     -Pdeployable-jar
12
13 FROM eclipse-temurin:17
14 RUN useradd -ms /bin/bash olog
15 RUN apt update && apt install -y --no-install-recommends curl
16 COPY --from=build /phoebus-olog/target /olog-target
17 RUN chown olog:olog /olog-target
18 USER olog
19 WORKDIR /olog-target
20 CMD ["java", "-jar", "service-olog*.jar"]
```

Server als Docker-Host

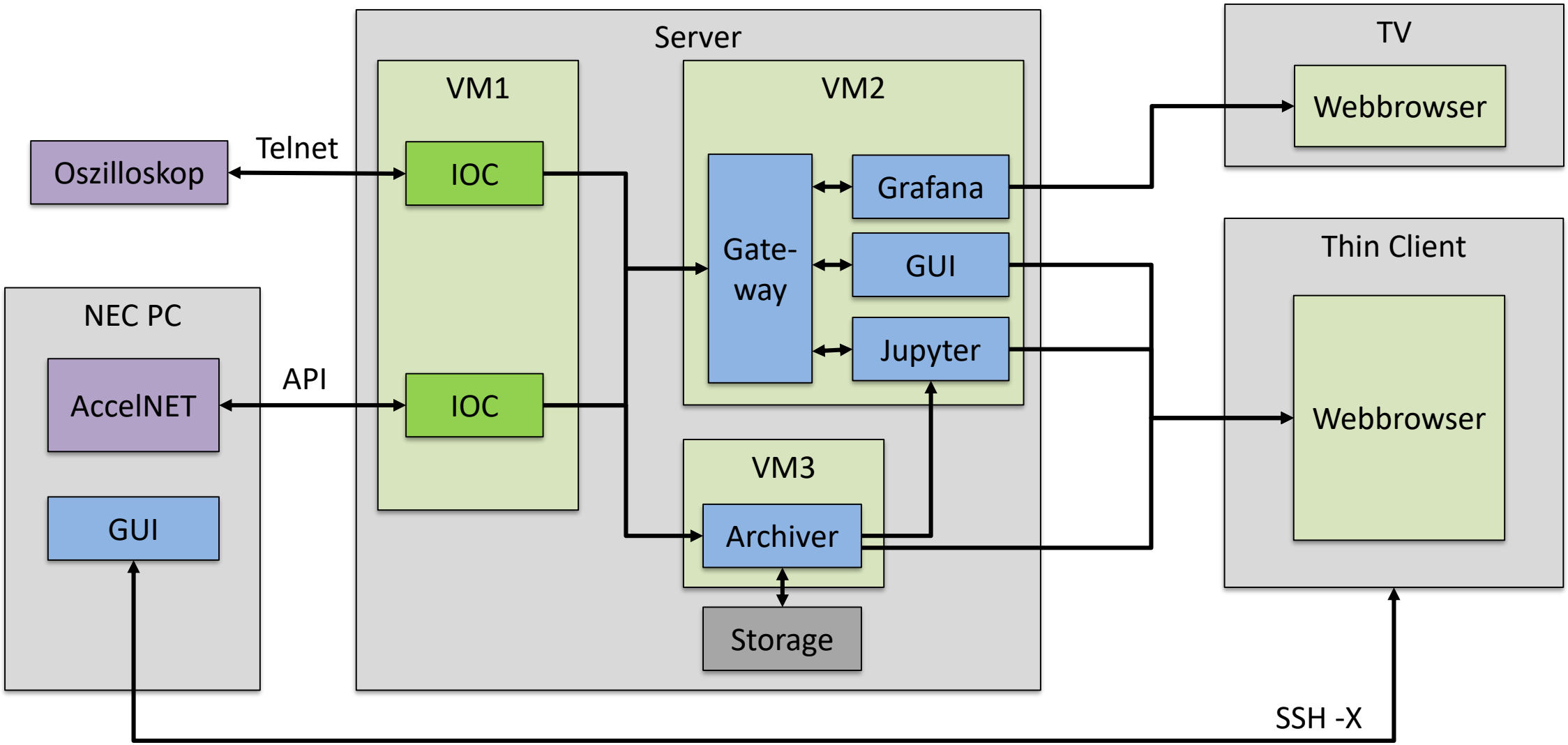
- Host für aktuell 7 virtuelle Maschinen, in denen mehrere Docker Container laufen
  - Minimale Einrichtung jeder VM: git, docker, ntp und nfs/smb
  - 13 IOCs und 45+ Dienste
- Ausgelegt auf 10.000 Variablen mit 1 Hz Änderungsrate
- Anbindung an das Backend im Rechenzentrum (Backup, Firewall)



- Eine schwächere Workstation oder mehrere Embedded-PCs sind auch möglich!
- Umstellung von VMWare ESXI zu Proxmox VE

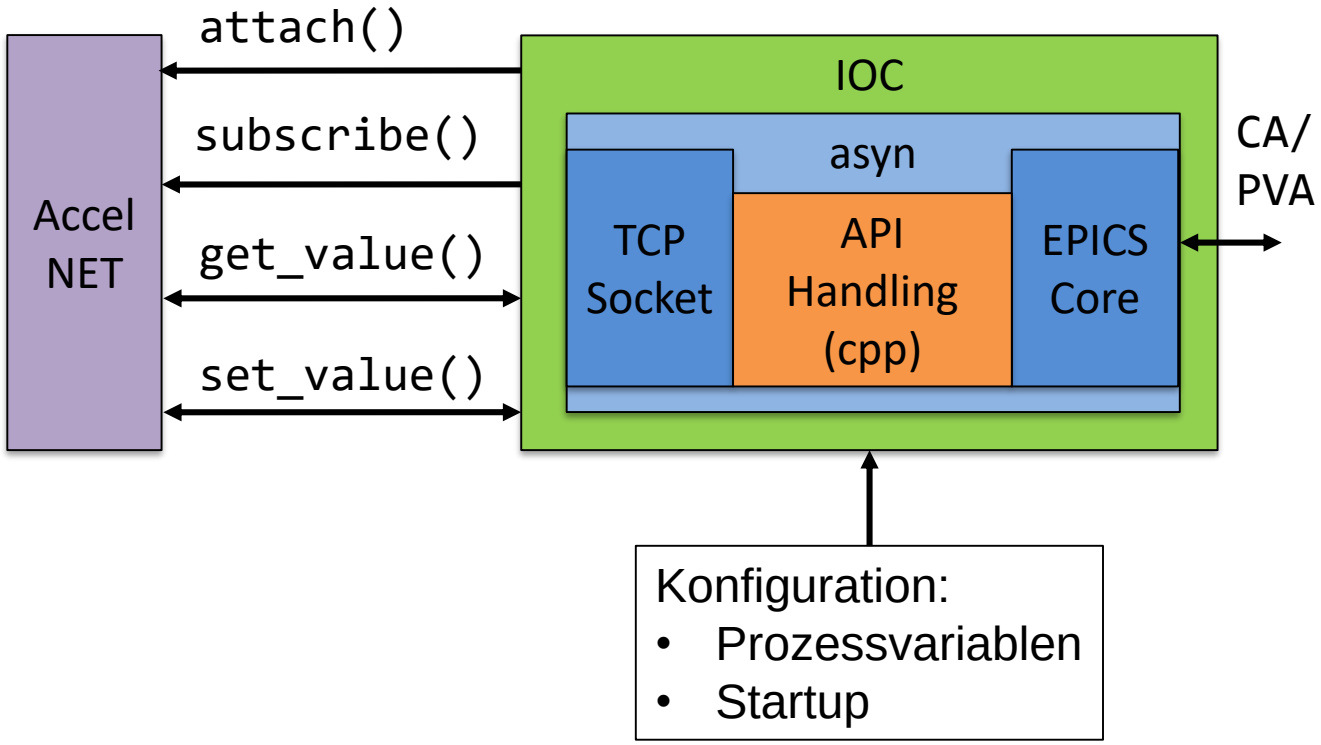
- Im Kontrollraum nur Thin-Clients mit Remote-Verbindungen zum Server via Webbrowser

Grundlegender Aufbau und Kommunikationswege



Anbindung von Hardware an EPICS

- IOC ist die Schnittstelle zwischen Hardware und den EPICS Netzwerkprotokollen
- Konzentration auf den Device-Treiber
- Für viele Protokolle gibt es fertige Device-Treiber (Modbus, OPC-UA, Telnet, USB-TMC, GigE, ...)




```
record(ai, "$(P)-IRd")
{
    field(DESC, "FaradayCup Current Read")
    field(DTYP, "asynFloat64")
    field(SCAN, "I/O Intr")
    field(INP,  "@asyn($(PORT),0,1)$(P)-IRd")
    field(PREC, "2")
    field(EGU,  "A")
    info(NEC,   "$(NEC)|CR")
}
```

→ Der Gerätehersteller muss eine offene Schnittstelle bereitstellen

Infrastructure as Code mit Docker Compose

- Konfiguration der Container (eigene + externe) über Dateien auf dem Host
- Zusammenspiel zwischen Containern: Datenbanken, Netzwerk
- Versionierung

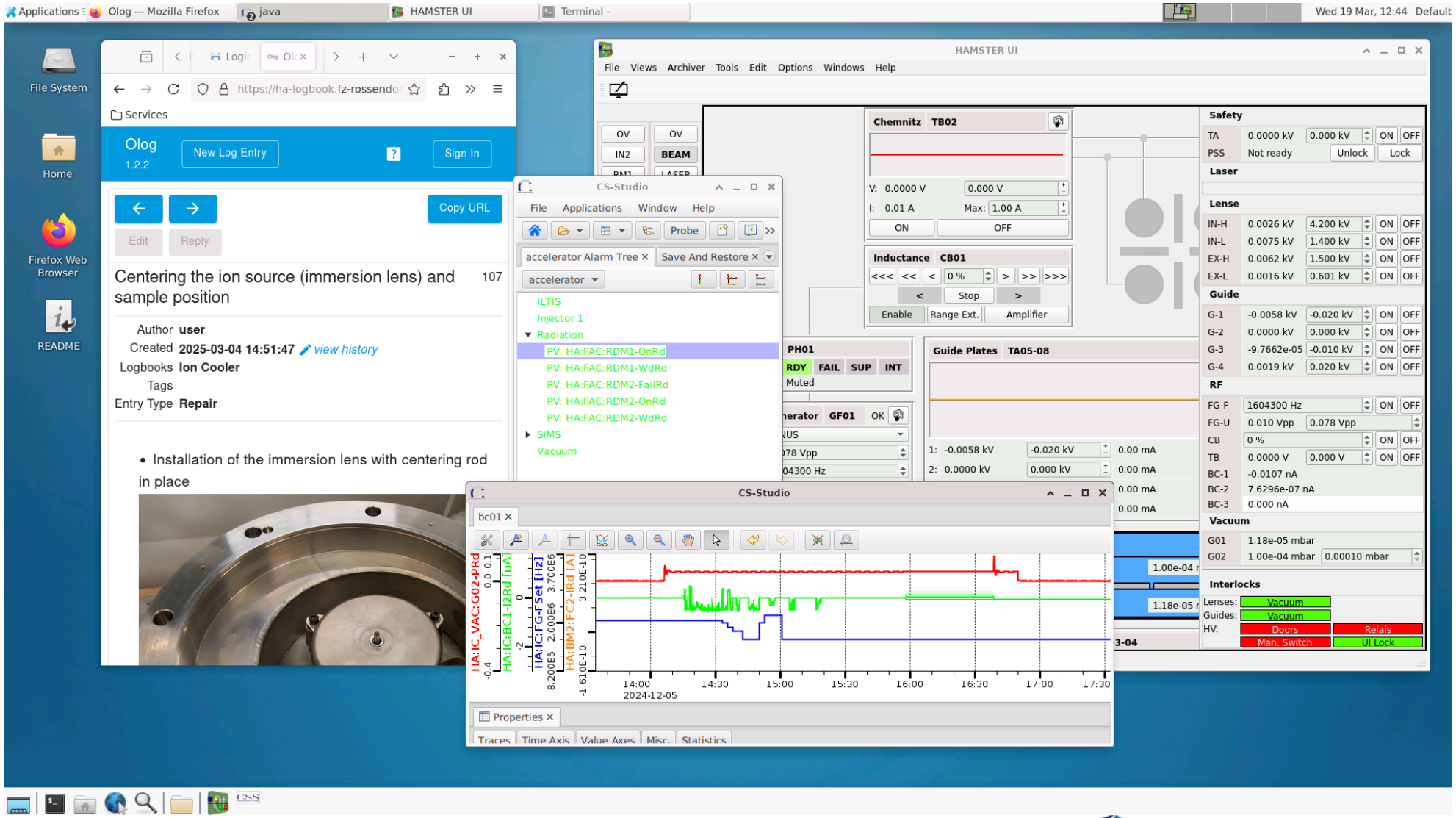
```
services:
  accelnet:
    image: registry.hzdr.de/fwf/libraries/epics/accelnet-ioc:latest
    stdin_open: true
    tty: true
    environment:
      - PUID=1000
      - PGID=1000
    volumes:
      - ./config/accelnet/boot:/usr/local/epics/accelnet-ioc/iocBoot/iocaccelnet
      - ./config/accelnet/db:/usr/local/epics/accelnet-ioc/db
    restart: unless-stopped
    network_mode: host
```

→ **Das komplette Deployment ist in Textdateien festgehalten!**

GUI in einem Headless-Desktop Container

- Shared Desktop über VNC im Webbrowser
- Tools und Bibliotheken installiert
- Alle externen Dienste vor-konfiguriert

- Einheitliche GUI
- Editierbar und versioniert



Weitere Dienste

- Verzeichnisdienst für verfügbare Prozessvariablen
- Alarm-System mit Quittierung und Logging
- Jupyter-Notebooks für Skripting und Datenauswertung
- SMB-Storage zum Messdatenaustausch

- Elektronisches Laborbuch
- Dokumentations-Wiki
- Grafana

- State-Machines für Überwachungen und übergreifende automatische Reaktionen (z.B. Stromausfall)
- Bluesky für automatisierte Messungen

- ... und was sich die Nutzer noch wünschen werden.

Durchführung von Updates

- Das Aktualisieren der Dienste ist aktuell nicht automatisiert
  - Prüfen fremder und Anpassen eigener Änderungen hinsichtlich Kompatibilität
- Images werden automatisch in der Gitlab CI gebaut und in die Registry hochgeladen
- Container können vor dem Deployment lokal getestet werden

- Der Austausch des Containers erfolgt in wenigen Sekunden

```
docker compose pull olog
docker compose down olog && docker compose up –d olog
```

- Bei Fehlern, tauscht man zurück auf die vorherige Version des Image
  - Image-Tags sind wichtig!
  - Dateien außerhalb des Containers müssen im Backup sein

Vor- und Nachteile des Container-Deployments

- **Vorteile**
  - Getrennte Laufzeitumgebungen (Update von Containern ohne Abhängigkeitskonflikte)
  - Host-Hardware flexibel (von Embedded-PC bis Cluster)
  - Container auch für die Entwicklung nutzbar
  - Übergreifende Konfiguration in Text-Dateien (volle Versionskontrolle)
  - Images können mit anderer Konfiguration für andere Projekte wiederverwendet werden

- **Nachteile**
  - Linux stark empfohlen
  - Einarbeitung in die Arbeit mit Containern notwendig
  - Häufig Probleme mit Dateiberechtigungen
  - Die (manuelle) Update-Wartung selbst bleibt bestehen

Zusammenfassung

- Deployment von IOCs und (Mikro-)Diensten in Docker Containern
- Mehr Konfigurieren als Programmieren
  - Infrastructure as Code für Images und Compose
  - Trotz Menge der Dienste für eine Person beherrschbar
- Stabilität sehr gut
- Updates erfolgreich durchgeführt

- EPICS bietet für unterschiedliche Geräte ein einheitliches Interface
- Die EPICS Community ist großartig!

- Nächste Schritte:
  - Einbinden von weiteren Geräten → Steigung der Anzahl an Prozessvariablen
  - GUI vervollständigen
  - Automatisierte Messungen und Optimierungen

Vielen Dank!