Data Center Observability Blueprint

Document Control

Version	Date	Changes Made
1.0	20.04.2025	Initial version Addition of graphical additions, reference architecture and example architectural design
1.1	19.12.2025	Removal of tool specific mentions (ELK, Telegraf, Netbox). Replacing alerting examples in Appendix A with Change management & Visualization requirements.
1.1.1	07.01.2026	Update references to ApeiroRA components in Appendix B & C, include AperioRA refernces in abstract

Table of Contents

ABSTRACT

This document outlines a blueprint for an open-source data center monitoring and observability platform, based on Apeiro Reference Architecture.
Designed to address the evolving operational challenges faced by modern data center operators. Recognizing the increasing demands for efficiency, real-time insight, and scalable infrastructure management, the proposed solution targets the granular monitoring of individual racks through Telemetry devices, including Embedded Monitoring Units, intelligent Power Distribution Units, and environmental sensors.

It emphasizes real-time data collection and analysis, SNMP-based monitoring at the socket level, and comprehensive environmental oversight. Integration with industry-standard tools such as OpenTelemetry, CMDB, data lakes, and ticketing systems ensures streamlined incident response, enhanced interoperability, and sustainability.

By overcoming common limitations like scalability constraints and vendor lock-in, this approach delivers improved reliability, performance, and operational visibility for data centers of varying scale and complexity.

1. Introduction

Modern data centers must prioritize operational efficiency, real-time monitoring, and scalability to manage increasing complexity. Key components like Embedded Monitoring Units, intelligent PDUs, and environmental sensors are vital for reliable power distribution and maintaining optimal conditions. However, current DCIM solutions often face issues such as limited scalability, vendor lock-in, and compatibility challenges, which can hinder proactive infrastructure management.

This blueprint presents an open-source monitoring platform designed for data center operators, focused on individual racks with telemetry devices (management points, PDUs, sensors).

Key features include:

- Real-time operational data collection and analysis
- SNMP-based monitoring of EMUs and PDUs at the socket level
- Environmental tracking for temperature and humidity
- Integration with CMDB, data sinks and ticketing systems

The following sections will address core operational monitoring and control challenges in modern data centers.

1.1 Challenges

Modern data centers are rapidly evolving due to technological, regulatory, sustainability, and accountability pressures. They now operate as transparent, compliant ecosystems serving various stakeholders beyond IT. As a result, advanced observability platforms are needed to meet both technical and strategic organizational goals.

Key challenges in the data center environment include:

- **Customization Requirements:** Standardized solutions may not address the distinct needs of various organizations. Customization is often implemented to ensure infrastructure aligns with business objectives.
- **Integration:** Managing and integrating multiple components can introduce complexity and require specialized expertise.
- **Legacy Systems:** Integrating legacy systems with modern infrastructure may cause compatibility or performance issues.
- **Resource Allocation:** Efficiently allocate resources to meet workload needs and reduce contention.
- **Deployment Complexity:** Manage operations across varied, multi-site data centers.
- **Cost-Effectiveness:** Utilizing an open-source and cost-efficient platform over proprietary alternatives.

Monitoring embedded units, PDUs, and environmental sensors presents additional challenges, such as:

- **Scalability:** Monitor all Embedded Monitoring Units and PDUs across distributed sites.
- **Interoperability:** Integrate with existing CMDB and incident management tools without depending on specific vendors.
- **Accurate Insights:** Deliver timely data to reduce downtime and optimize performance.
- **Environmental Metrics:** Measure factors like temperature and humidity.

These factors indicate the necessity for revised approaches. The subsequent section introduces the intent and strategic direction of this blueprint.

1.2 The Need for Observability

Traditionally, data center monitoring has been treated as a technical function, an operational necessity for detecting faults, measuring resource usage, and maintaining uptime. However, the **expectations placed on modern data center operations have evolved significantly.**

Organizations now face demands such as:

- **Timely performance visibility** across globally distributed, heterogeneous environments (including co-location and edge sites).
- **Cross-functional integration** between facility, infrastructure operations teams, application operations teams, capacity planning, procurement, finance, and sustainability teams.
- **Compliance with stringent regulatory frameworks** requiring not only energy efficiency but also auditability, transparency, and continuous reporting.
- **Adaptability to rapid deployment cycles and automation pipelines** enabled by Infrastructure as Code **(IaC),** containerization, and cloud-native paradigms.

Observability, defined as the ability to infer the internal state of a system from its telemetry (logs, metrics, traces) is no longer a tooling upgrade. It has become a foundational capability for enterprise resilience, regulatory compliance, cost optimization, and environmental responsibility.

1.3 Purpose of Blueprint

The goal of this blueprint is to deliver a targeted, composable, scalable, and modular monitoring platform to:

- Provide visibility into Embedded Monitoring Units and their connected devices.
- Provide timely insights into power consumption, environmental conditions, balance, function and device health.
- Simplify the management and reporting of operational data using open-source tools and technologies.
- Automate deployment and management using
- Centralize data processing with **data lake** for enhanced analytics.
- Provide integration with **various systems (i.e. Ticketing)**.

By implementing this solution, organizations will be able to:

- Reduce operational risks through proactive monitoring and alerts.
- Streamline incident management workflows via integration with Ticketing platforms.
- Lay the foundation for future observability enhancements, expanding beyond Embedded Monitoring Units and PDUs…

Before diving into the technical framework, it is critical to understand who this solution serves and how it influences organizational roles and responsibilities.

1.3.1 Primary Objectives

This blueprint aims to achieve the following objectives:

- Establish a unified architectural framework for integrating telemetry across power, environmental, and asset management domains within data center environments, utilizing Aperio Reference Architecture components
- Translate legal and regulatory obligations (e.g., Directive 2023/1791, Delegated Regulation 2024/1364) into technical design requirements and system capabilities.
- Define a reference observability pipeline, including data ingestion, normalization, enrichment, visualization, and alerting using open-source tools.
- Support dynamic, real-time observability of data center components such as PDUs, sockets, phase loads, environmental sensors, and embedded monitoring units.
- Enable integration with CMDB and Incident Management / Ticketing platforms to ensure traceability, auditability, and automation of responses to anomalies or regulatory triggers.
- Provide implementation guidance for scalable deployment across distributed data center infrastructure, with applicability to co-location facilities and edge environments.

1.3.2 How to Read This Document

This document is structured as both a strategic reference architecture and a technical implementation guide. It is intended to support a wide range of stakeholders with varying levels of technical expertise, functional responsibility, and regulatory involvement. As such, the document has been designed to be modular, navigable, and role-aware, allowing readers to engage with the content based on their individual perspective and operational focus.

To maximize clarity and usability, this section offers guidance on how to navigate and extract value from the blueprint according to stakeholder function. Readers are encouraged to approach the document non-linearly, focusing on the chapters that are most relevant to their responsibilities while consulting foundational sections as needed for context.

1.3.3 Recommended Reading Paths by Audience

Audience	Recommended Sections	Purpose
Infrastructure & Platform Engineers	Chapters 2, 4, 6 Appendices	Understand architecture, telemetry sources, and deployment models
Sustainability & Compliance Officers	Chapters 1.1, 3.6.5, 4.5, 6.4	Map observability platform to EU directives, ESG KPIs, and reporting needs

Data Center Operations & Capacity Planners	Chapters 2, 3, 4.2–4.3, 6.2, 6.3	Enable proactive planning, incident prevention, and power/environmental visibility
Service Management & Incident Teams	Chapters 3.6.3, 4.6, 6.3–6.4	Understand alert correlation, CMDB linkages, and incident resolution workflows
Executive & Strategic Leadership	Chapters 1.1–1.2, 3.6.6–3.6.8	Evaluate business value, risk reduction, and alignment with transformation strategy
External Partners & Regulatory Bodies	Chapters 1.3–1.4, 4.6, 6.5	Assess regulatory alignment, system openness, and cross-entity interoperability

1.3.4 Reading Recommendations

- Readers unfamiliar with observability concepts or data center operations are advised to start with **Chapter 2**, which introduces key principles, layers of observability, and distinctions from traditional monitoring practices.
- Those evaluating the blueprint for alignment with **EU regulations or internal ESG frameworks** should consult **Chapter 1.3** for legal mappings and **Chapter 4.5** for data retention and audit considerations.
- Readers involved in actual **design and deployment** of observability platforms should focus on **Chapter 4 (architecture)** and **Chapter 6 (implementation strategy)**, which provide a technically actionable pathway for rollout.
- Use cases, alert thresholds, and real-world implementation logic can be found in the Appendices, which serve as reference material for practitioners.

1.3.5 Document Use in Practice

This document may be used in multiple contexts:

- As an **architectural guide** for internal observability platforms across data centers and co-location facilities.
- As a **compliance readiness framework**, showing how data flows, reporting outputs, and controls align with regulatory requirements.
- As a **stakeholder alignment** guide during cross-functional workshops, procurement evaluations, or design reviews.
- As a **knowledge base** for onboarding technical teams or partners contributing to observability deployment and lifecycle management.

Readers are encouraged to annotate, extend, or adapt this blueprint to suit their local environment, technology stack, or regulatory jurisdiction. All design principles outlined herein are intended to be composable, reusable, and modular.

1.4 Regulatory Considerations (Regulatory-Driven Framing)

This section outlines how European and national regulations affect current data center operations. The resulting legal requirements and sustainability objectives serve as design considerations for the monitoring and observability platform described in this blueprint.

Infrastructure teams typically consider uptime, cooling, and space as primary factors. Recently, energy consumption, carbon emissions, and public accountability have also been identified as key considerations.

- **Data centers use ~3–4%** of total electricity in Europe.
- The European Union now **requires reporting** of energy efficiency and environmental impact for data centers of certain sizes.
- **Failure to comply may lead to legal risk**, fines, or blocked expansion in countries like Germany and the Netherlands.
- **Clients and investors**— sustainability is now a competitive advantage.

Observability now serves as both a compliance tool and a business enabler.

A simplified workflow:

1. **EU Regulation** (e.g., PUE reporting)
2. **Requires data** (IT power vs. total power)
3. **Data collected** via SNMP or Redfish from PDUs
4. **Data processed** to align with OpenTelemetry model
5. **Stored** in a data lake
6. **Visualized** through dashboards
7. **Reports generated** for EU, audits, and executives

This system streamlines compliance, enhances visibility, and delivers operational insights—all framed by primary regulatory texts and mandates.

1.4.1Market and Operational Challenges in the Data Center Sector

Across Europe, data center operators from cloud providers to enterprise infrastructure teams face converging pressures including:

Category	Challenge Description
Operational Complexity	Growing footprints of global data centers, co-location cages, and edge sites leads to inconsistent monitoring and blind spots.
Legacy vs. Modernization	Many DCs operate mixed environments with legacy infrastructure that lacks telemetry support.

Energy Reporting Gaps	Operators often cannot measure or report energy consumption at sufficient granularity for audit or compliance.
Cooling & PUE Monitoring	Temperature and humidity data is inconsistently tracked, and PUE calculation often lacks real-time accuracy.
Siloed System Ownership	IT, facilities, finance, and compliance teams use separate tooling, hindering coordinated monitoring efforts.
Sustainability Proof Gaps	Reporting frameworks (ESG, EU CSRD) require traceable power and carbon data - current tooling often lacks in this area.

These are no longer just best-practice shortcomings - many are now subject to legal and regulatory enforcement, particularly under the revised **Energy Efficiency Directive (EU 2023/1791)** and its delegated implementation regulation **(EU 2024/1364).**

1.4.2 Regulatory Requirements (as per Apr.2025)

Below is a mapping of specific, enforceable obligations to architectural or design responses.

Regulation	Legal Requirement	Blueprint Response / Mapping
Directive (EU) 2023/1791 (EED – Article 12)	Data centers with installed IT power ≥ 500 kW must report energy efficiency indicators by Sept 15, 2024.	Telemetry from PDUs + Assent Metadata → data Lake → exportable KPIs
Delegated Regulation (EU) 2024/1364 (Annex I)	Operators must report: PUE, temperatures, waste heat reuse, renewable share, energy reuse factor, water usage.	Floor- and socket-level monitoring, ambient temp/humidity, data lake enrichment
Regulation (EU) 2019/424 – Ecodesign	Applies minimum efficiency standards for servers/storage. Compliance needed to procure/operate such devices.	CMDB-based mapping of equipment type and model + firmware version tracking
Climate Neutral Data Centre Pact (self-regulation)	Commit to 100% renewable use by 2030, energy reuse, clean water use.	Renewable source tagging per site, alerting if usage exceeds brown energy thresholds

Sources:

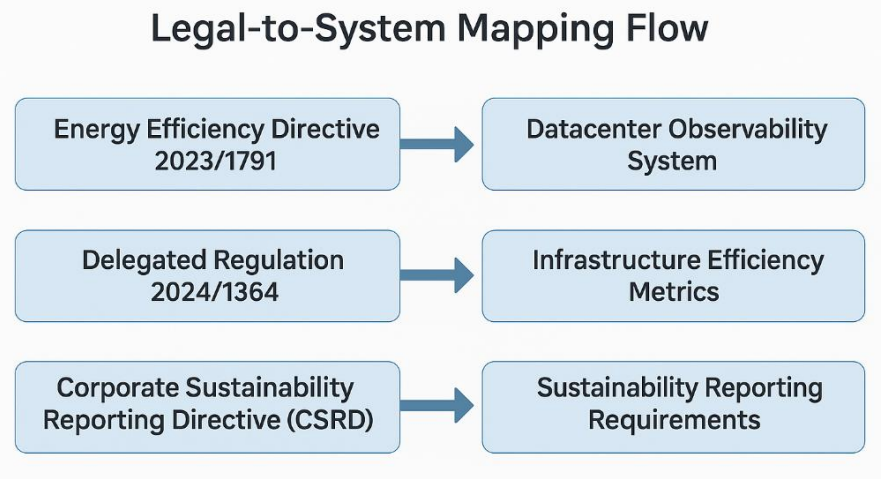
- DIRECTIVE (EU) 2023/1791
- DELEGATED REGULATION (EU) 2024/1364
- ECODESIGN REGULATION (EU) 2019/424
- GERMANY EEG 2023

(subject to update as per legal proceedings – no claim for long term accuracy)

1.4.3 Formalized Requirements Inferred (excerpt)

Each KPI or mandate in the regulatory documents logically implies an infrastructure or observability feature. Below are examples of traceable causality:

Legal Mandate (Verbatim)	**Inferred Platform Requirement**
"Operators shall report their Power Usage Effectiveness (PUE)" (Reg. 2024/1364, Annex I)	Measurement of IT energy vs. total facility energy → Derived from rack PDU data vs. site-wide metering
"…temperature set point in IT spaces and external air temperature"	In-rack and whitespace temperature monitoring, normalized and reported at time-synced intervals
"Share of electricity from renewable sources"	Tagging of sites by energy sourcing metadata, with integration from procurement or power provider contracts
"Data center waste heat utilization potential"	Telemetry on inlet/outlet air differential temperatures, airflow patterns, or BTU calculations
"Annual water usage for cooling"	Integration with environmental and facility sensors where cooling towers or liquid-based systems are used
"Reporting KPIs to the central EU database annually (Article 12, EED)"	Reporting/export API from **Data Lake** or intermediate dashboard layer, structured as per delegated regulation schemas



1.4.4 Advanced EU Sustainability and Taxonomy Compliance Mapping

As data centers become both critical infrastructure and regulated environmental actors, the European Union has introduced overlapping legal frameworks and voluntary pledges that redefine observability from a purely operational tool into a sustainability-enabling control system.

This section provides a glimpse of how advanced EU legislation - beyond baseline energy efficiency directives, shapes architectural decisions, metadata design, and audit workflows within the observability platform.

Where Section **1.4.2** presented formal reporting requirements, and Section **1.4.3** mapped legal text to inferred observability needs, this section translates strategic sustainability mandates, including the **Corporate Sustainability Reporting Directive (CSRD),** the **EU Taxonomy Regulation**, and the **Climate Neutral Data Centre Pact (CNDCP)** into design implications. It also addresses forward-looking obligations such as energy reuse, water efficiency, and Scope 3 readiness.

The Corporate Sustainability Reporting Directive (Directive (EU) 2022/2464)

CSRD introduces a dual-reporting obligation known as double materiality:

- **Financial materiality:** How environmental risks (e.g., energy costs, outage-prone infrastructure) affect the organization's performance.
- **Impact materiality:** How the organization's operations affect climate, resources, and society (e.g., carbon emissions, water draw, waste heat).

To support CSRD-aligned reporting, observability platforms must enable quantifiable, traceable, and timestamped metrics that reflect both categories.

These include:

CSRD Topic	**Observability Feature**

Scope 2 emissions	Site-level renewable energy tagging in CMDB and PDU telemetry
Energy intensity	Rack-to-room telemetry normalized by IT workload density
Physical climate risks	Overheat event detection, thermal capacity tracking
Material use and lifecycle	Telemetry-enriched health data for PDUs and sensors

Design Implication: Support ESRS-aligned exportable metrics (e.g., **kWh/rack/month**, **% renewable energy**, **ΔT over time**) and link observability dashboards to disclosure tools or corporate sustainability platforms.

EU Taxonomy Regulation: Substantial Contribution and DNSH

The **EU Taxonomy (Regulation (EU) 2020/852)** defines criteria for economic activities to be classified as "environmentally sustainable."

Data center activities must:

- Substantially contribute to at least one environmental goal (e.g., climate mitigation)
- Not significantly harming other (DNSH) (e.g., water ecosystems, circular economy)
- Respect minimum social and governance safeguards

Mapped DNSH Considerations:

DNSH Pillar	**Observability Design Response**
Climate mitigation	Real-time PUE tracking, renewable energy source tagging
Water protection	Integration with water flow meters (cooling towers, CRAC)
Pollution prevention	Waste heat BTU tracking, ΔT differentials across containment zones
Circular economy	Equipment age/lifecycle telemetry for predictive replacement
Resource use efficiency	Load balancing visualizations and underutilization reports

Design Implication: Introduce metadata tags for DNSH coverage at rack, room, and site levels. Support spatial overlays and metric summaries that reflect taxonomy criteria.

Energy Reuse and the Energy Reuse Factor (ERF)

Per Annex I of **Delegated Regulation (EU) 2024/1364**, ERF must be reported—the proportion of energy repurposed for heating or secondary uses. This requires in-rack temperature and airflow telemetry, waste heat measurement (via BTU estimation or Redfish API), and tracking reuse-eligible versus total energy consumed.

Design Implication: Extend telemetry to support ERF calculations and heat reuse mapping; include compliance indicators for areas with reclaimable heat versus passive losses.

Water Usage and Cooling Efficiency

Data centers using adiabatic or evaporative cooling must now report water usage in line with **EED Article 12 and 2024/1364**. Mapped Requirements are Flow sensor integration (via Modbus, BACnet, or REST). Normalization of usage per kWh delivered IT load and Geotagging for water-stressed regions and governance zones.

Design Implication: Ingest and store water metrics alongside energy, apply temporal alignment, and ensure audit traceability and add a telemetry schema extension to support WUE (Water Usage Effectiveness) export.

Climate Neutral Data Centre Pact (CNDCP) Self-Audit Metrics

- Operators participating in the CNDCP pledge to:
- Achieve PUE targets (≤1.3 for cooler zones),
- Reach 100% renewable energy by 2030,
- Implement heat reuse and water efficiency goals.

Design Implication: Introduce a Pact Compliance Dashboard template (see Appendix D), summarizing PUE trends and violations, Renewable sourcing status, Waste heat reclaims rate and zones of potential, Energy and water KPIs over time (rolling and seasonal).

The dashboard should support PDF/JSON/CSV exports, optionally validated against Pact milestones for automated self-audit.

Regulatory Design as System Constraint

Advanced regulatory frameworks no longer act solely as external requirements - they now serve as architectural constraints that shape metadata schemas, telemetry priorities, and reporting outputs. Designing for EU Taxonomy alignment or CSRD audit-readiness from the outset reduces long-term retrofit costs and positions observability platforms as not just technical, but strategic enablers of sustainability disclosure.

2 Understanding Datacenter Monitoring and Observability

2.4 Overview

Datacenter monitoring and observability are closely related; **however, observability** goes beyond traditional monitoring which focuses on understanding and analyzing the state of target systems based on their outputs.

Observability enables organizations to detect, diagnose and act towards timely and/or proactive resolution of issues through the added benefits of deeper insights in said systems.

In summary, observability is the practice of using telemetry data **– logs, metrics, traces, outputs** to understand the behavior of systems with the goal of enhanced management, enhanced stability, design, identification of bottlenecks, predicting failures and proactive actions.

Datacenter monitoring and observability are essential for modern infrastructure management. They go beyond merely tracking metrics to include comprehensive insights into system behavior and configuration states.

With the inclusion of CMDB and desired state reporting, we enhance visibility and control, ensuring systems remain aligned with operational goals and compliance requirements.

Monitoring focuses on "How it is going and what happened," while observability seeks to answer, "Is it acceptable and why it happened." With CMDB and desired state reporting, organizations can also address "how it aligns with expected behavior."

2.4.1 Hardware Layer

The Hardware Layer is the foundation for observability by providing telemetry data from specific datacenter components, such as:

- **Embedded Monitoring Units:** acts as a central point for collecting telemetry data from/about connected devices
- **Power Distribution Units (PDU's):** Provides detailed metrics on power usage, capacity, efficiency, fluctuations & etc.
- **Environmental sensors:** Capture information like temperature, humidity, for overall analysis of environmental parameters. Embedded monitoring
- **Switches and Breakers:** Offer operational metrics and fault detection capabilities.

2.4.2 Observability Software Layer

The Observability Software Layer collects, processes and organizes telemetry data from the hardware layer with features that include:

- **Telemetry Aggregation**: Collects logs, metrics, traces, outputs from targeted hardware sources.
- **Data Processing and Correlation**: Links data points to dependencies, issues and desired insight.
- **Timely Analysis**: pattern and anomalies identification upon occurrence
- **Configuration Management Integration**: Continuously synchronization with CMDB to reflect the current state.
- **Desired State Management:** Monitoring resources to ensure alignment with predefined configurations, alerting to deviations and config drift.
- **Self-Healing Capabilities**: Automatic remediate discrepancies between current and desired states when possible.

2.4.3 API and Telemetry Layer

This layer acts as a bridge between the hardware, monitoring and observability platform, external and third-party systems/tools. This layer includes:

- **Access to CMDB Data**: Enables real-time querying and updates to maintain synchronization between the physical and logical infrastructure.
- **Standardized Interfaces**: OpenTelemetry, SNMP, and REST APIs facilitate integration with observability and configuration.
- **Desired State Reporting**: APIs support state validation workflows, generating reports that highlight compliance or deviations.
- **Standardized Access:** Interfaces like OpenTelemetry, Redfish API, REST API and SNMP enable uniform data collection and interaction.
- **Integration**: Facilitates interoperability, management, orchestration, and analytics platforms.
- **Programmatic Control**: Allows automation, robotic workflows and customization of data pipelines.

2.4.4 Visualization and Insights Layer

This layer represents the actual point where stakeholders and interested parties (users) interact with monitoring and observability data. It includes:

- **Dashboards**: visual representation of key metrics and trends
- **Alerting mechanisms**: Notification for threshold breaches or anomalies
- **Advanced Analytics**: Historical and empirical analysis
- **CMDB Visualization:** integrated view of hardware and software configurations

- **Desired State Reports**: Highlights discrepancies between the current and desired states, including possible actionable insights into remediation.
- **Floorplan Visualization and Heatmapping:** Provides briefly graphical representation of data such as rack location, consumption, acceptable operating parameters, capacity etc.
- **Predictive Analytics and Alerts:** Based on telemetry data and CMDB information to predict states and/or issues.

2.5 Core Principles of Datacenter Monitoring and Observability

Principle Name	Description
Telemetry driven insights	Capturing and analysis of telemetry data such as logs, metrics, and traces and outputs to provide actionable insights into system behavior and performance.
Resource Visibility	Ensures all hardware and software resources, such as PDUs, Embedded Monitoring Units, and sensors, are visible and accounted for in monitoring systems.
Dynamic Resource Allocation	Allocating monitoring resources dynamically to adapt to changing workloads and environments against CMDB
Automation and Orchestration	Automates observability workflows and aligns resource allocation with real-world operational demands.
Interoperability	Ensures integration with diverse tools, platforms, and telemetry standards through APIs and open protocols.
Scalability and Elasticity	Enables observability solutions to scale as infrastructure grows, ensuring consistent performance across environments.
Proactive Maintenance	Predicts potential failures using advanced analytics.
Data Correlation and Context	Links telemetry data from multiple sources to provide context.
Flexibility and Adaptability	Adapts monitoring and observability processes to accommodate new technologies and evolving business requirements.
Software-Defined Control	Uses software to manage and configure observability processes, ensuring scalability and adaptability.
Resource Isolation	Ensure observability workloads do not interfere with operational systems, preserving performance and security.

Resource Efficiency	Optimizes the use of resources for telemetry collection and analysis, reducing overhead.

2.5.1 Benefits of Observability

Effective observability brings transformative benefits to datacenter management:

- **Deep System Understanding**: Observability provides detailed insights into the internal state of systems, revealing dependencies, operational state and potential risks.
- **Proactive Management**: Predictive analytics enables addressing issues before they impact operations.
- **Enhanced Automation**: Observability data powers automation workflows for incident response and resource optimization.
- **Integration**: Open standards and APIs facilitate interoperability with existing IT and operational platforms.

2.5.2 Observability vs. Monitoring

While monitoring tracks predefined metrics and alerts when anomalies occur, observability takes a broader approach:

- Monitoring answers the question, “What is wrong and how it is going?”
- Observability answers the question, “Is it normal and why is it wrong?”

Aspect	Monitoring	Observability
Scope	Tracks Predefined metrics and states	Analyzes telemetry to provide insights about system state and behavior.
Focus	What is wrong and how is it going?	Is it normal and why is it wrong?
Data Sources	Metrics from specific components like PDUs or sensors.	Combines metrics, logs, outputs and traces for a holistic view.
Use Cases	Threshold-based alerts and basic performance tracking.	Root cause analysis, anomaly detection, and predictive analytics.

2.5.3 Benefits of Integrating CMDB and Desired State Reporting

Integrating CMDB and desired state reporting into monitoring and observability practices provides significant advantages:

Configuration Compliance: Ensures that all resources remain aligned with predefined baselines.

Enhanced Reliability: Timely identification and remediation of discrepancies.

Centralized Management: Consolidates configuration and telemetry data for a holistic view of the datacenter or datacenter units.

Predictive Insights: Combines CMDB data with telemetry to anticipate configuration drift and its impact.

Improved Governance: Strengthens auditability and compliance with industry standards.

2.5.4 Monitoring, Observability and Desired State Reporting Example

Scenario: Configuration drift detected in a critical power distribution unit (PDU) with parameter threshold crossing in line with prior definition.

The process follows the pattern:

1. **Data Collection & Alerting**: The PDU telemetry indicates unexpected behavior.
2. **CMDB Validation:** Observability tools query the **CMDB** and detect that the current configuration does not match the desired state.
3. **Root Cause Analysis**: Logs and traces correlate the drift to a recent automated firmware update.
4. **Automated Remediation**: The system rolls back the firmware to the desired version and updates the CMDB.
5. **Reporting**: A desired state compliance report is generated, documenting the issue and resolution for auditing purposes.

To translate these principles into reality, we now provide a detailed implementation roadmap—outlining how to deploy and operate the observability platform across environments.

3 Stakeholders and Organizational Impact

A solution is only as valuable as the problems it solves for its users. This section maps key internal and external stakeholder groups across data center ecosystems, highlighting their priorities and challenges. By understanding business and operational needs, we ensure the observability platform is tailored for real-world impact. Each stakeholder group introduced here will have its requirements addressed in the subsequent architectural and implementation sections.

We hereby undertake an exemplary exploration of how observability practices affect specific stakeholder groups within the company. Understanding the who (i.e., key roles) and what (i.e., processes, workflows) is essential to articulating observability's strategic importance, especially for business professionals seeking buy-in from multiple parts of the organization.

3.1 Why Stakeholders Matter

The company operates an extensive global network of datacenters, co-locations and cloud environments, serving both external and internal clients, teams and interest groups. Therefore, it is standard practice that each facility must align to stringent requirements such as (but not limited to): **uptime, availability, stability, compliance and energy-efficiency.**

Complexity: Multiple hardware vendors, open-source tools, proprietary components, and partner integrations.

Accountability: Strict service-level agreements (SLAs) with enterprise customers require efficient incident management and rapid response times.

Sustainability: Sustainability metrics are critical across all data center locations.

Having these imperatives in mind, a wide range of stakeholders - beyond just data center operators - depend on reliable observability data. In this document we detail each group's concerns and how observability transforms their respective workflows.

3.2 Stakeholder Impact & Review Matrix

Data Center Observability is not a standalone technological function - **it is a business-critical enabler** across multiple stakeholder groups. In this section we attempt to provide a detailed mapping of each stakeholder group's priorities, the value derived from observability, and how the ApeiroRA Monitoring Blueprint addresses their specific needs.

Stakeholder Group	Primary Responsibilities
Data Center Operations & Engineering	Power/cooling oversight, infrastructure health, PDU/Embedded Monitoring Units maintenance
Infrastructure Architecture & Capacity Planning	Rack layout, scaling plans, energy design, power balance modeling
Sustainability & Energy Efficiency Teams	Energy footprint analysis, carbon reduction, reporting for GHG protocols
Security & Compliance	Audit readiness, fault logging, tamper alerts
Procurement & Asset Management	Inventory alignment, lifecycle planning, cost optimization
Incident & Operations Management (RunOps/NOC)	Incident triage, SLA compliance, root cause analysis

Cloud/Platform Engineering	Integration into cloud-native stacks, API access, GitOps
Management & Governance	Investment visibility, risk posture, policy enforcement

Table 1: Stakeholder Landscape and Responsibilities

Stakeholder	**How Observability Supports Their Role**	**Specific Feature Mapping**
DC Ops & Engineering	Dashboards, alerting, and root-cause analysis	SNMP telemetry from PDUs/Management points, alerts on 75%/95% thresholds
Infrastructure Architects	Predictive analytics, capacity utilization, load balancing	Consumption trends, rack/row/room granularity
Sustainability Teams	Visibility, carbon load tracking, anomaly detection	kWh tracking per rack/row/DC, load vs. estimated power vs. contract capacity
Security & Compliance	Fault/event correlation, tamper detection, audit logs	Logstash enrichment, Ticketing Platform for incident traceability, config drift validation
Incident Management (NOC/RunOps)	Single source of truth during outages	Ticketing + Data Lake + Alerting, PDU variance alerts
Cloud/Platform Teams	CI/CD pipeline compatibility, declarative infrastructure	GitHub Actions, Helm, Open Telemetry, Redfish API
Leadership & Governance	Evidence for strategic investments and compliance	KPI dashboards, audit compliance snapshots, capacity prediction charts

Table 2: Stakeholder Impact Summary

Stakeholder	**Sample KPI / Outcome**
DC Ops & Eng	50% faster root-cause resolution, 75% reduction in false alarms
Sustainability	Monthly GHG report automation, kWh per rack metric
Architects	Capacity forecast accuracy >90%
Security	Zero untracked config drifts in PDU Embedded Monitoring Units zones
Procurement	Asset utilization ratio >85%

Incident Teams	<15 min MTTR for power-related incidents
Cloud Engineers	Zero downtime during reconfigurations via GitOps
Governance	Blueprint adoption in 100% of new DC zones by 2025 Q3

Table 3: Stakeholder Outcomes & KPIs (example)

3.3 Primary Stakeholder Groups

3.6.1 Data Center Operations & Engineering

Responsibilities

- Oversee daily infrastructure tasks: **power distribution, cooling, rack management**.
- Maintain actual status for critical components like: **Embedded Monitoring Units, PDUs, Breakers, Loads and Environmental Sensors.**

Observability Benefits

- **Improved Incident Response:** Unified dashboards for SNMP/Redfish data expedite root-cause analysis (see Section 4.1.3). Operators can detect power overloads or thermal anomalies in seconds.
- **Compliance & Reporting:** Automated logs and continuous polling support streamlined audits (e.g., ISO 27001, EN 50600).
- **Scalability:** Leveraging open frameworks (Kubernetes, Open Compute Project) helps standardize new deployments without vendor-specific constraints.

3.6.2 Capacity Planning & Infrastructure Architecture

Responsibilities

Strategically planned expansions, cluster configurations, and future resource allocations for cloud and on-premises offerings.

Ensure synergy between emerging technologies (e.g., hyperconverged systems, container orchestration) and existing data center assets.

Observability Benefits

Predictive Capacity Management: Historical usage trends, from rack-level power metrics to container CPU usage, feed into sophisticated modeling for future growth scenarios.

Reduced Overprovisioning: Visibility into real utilization helps right-size deployments - especially crucial as companies pivots toward “green data centers” where capacity must meet sustainability goals.

Rapid Innovation: Observability fosters faster trial cycles for new hardware or cloud services, as architectural impact is visible in near-real-time logs and telemetry.

3.6.3 Operations & Incident Management Teams

Responsibilities

- Manage cross-data-center events.
- Uphold global SLAs for enterprise software customers.

Observability Benefits

- **Single Source of Truth:** Combining central **data ingestion** with **Ticketing** integration enables incident teams to see correlated alerts across geographies.
- **Faster MTTR:** Granular data from **PDUs**, temperature sensors, and system logs reduce guesswork, zero in on anomalies that span infrastructure and application layers.
- **Proactive Alerts:** Telemetry-driven thresholds (e.g., socket-level consumption over 80%) to prevent cascading failures and **SLA breaches**.

3.6.4 Finance, Procurement & Vendor Management

Responsibilities

- Oversee costs for hardware procurement, power and cooling infrastructure, large-scale DC production environments.
- Negotiate vendor contracts to align with strategic and sustainability objectives.

Observability Benefits

- **Cost Visibility: PDU metrics** at the socket or row level highlight potential inefficiencies (e.g., heavily over-utilized PDUs in one region vs. underutilized capacity in another).
- **Data-Backed Negotiation:** Detailed usage data supports more effective vendor contract discussions—whether for electricity rates or specialized hardware
- **Budget Forecasting:** Historical consumption patterns are invaluable for accurate quarterly or annual budgeting, aligning with major product lines or planned capacity expansions.

3.6.5 Sustainability & Compliance Officers

Responsibilities

- Oversee environmental commitments, ensuring carbon footprint reduction and compliance with **EU data protection and energy directives**.
- Publish internal sustainability reports and facilitate external audits (e.g., for Data Center Efficiency classification).

Observability Benefits

- **Granular Energy Tracking:** Observability data—temperature, humidity, power usage—feeds into carbon footprint analyses and continuous improvement in **PUE** (Power Usage Effectiveness) and for meeting **ESG** (Environmental, Social, Governance) reporting criteria.
- **Regulatory Readiness:** Automated logging and centralized data archiving for simplified compliance with local regulations, such as **Germany's EnWG** for energy and water usage or the **EU's Code of Conduct for Data Centers**.
- **Transparency & Innovation:** Visibility into exact load distribution encourages pilot projects (e.g., reusing waste heat or advanced cooling solutions) to achieve **net-zero goals**.

3.6.6 Executive Leadership & Business Strategy

Responsibilities

- Assist in **strategic roadmap** on IT investments or expansions.
- Monitor risk exposure, brand reputation.

Observability Benefits

- **Holistic Risk Assessment:** Executive-friendly dashboards **highlight high-risk areas,** e.g., a cluster nearing capacity or repeated sensor alerts in a key data center.
- **Strategic ROI:** Data-driven evidence of reduced downtime, operational costs, and carbon footprint.
- **Competitive Differentiation:** By demonstrating robust observability across all data centers, companies can showcase resilience and sustainability as part of their unique value proposition.

3.6.7 Impact on Core Processes

Beyond targeting specific roles, Observability reshapes fundamental data center and cloud operations:

- **Incident Response Workflow**

Enhancement: correlation of events—power surge plus specific usage spike—enables immediate escalation to the right on-call teams.

Outcome: Fewer false alarms, shorter mean time to acknowledge (MTTA), and overall improved service availability.

- **Change & Release Management**

Enhancement: Observability data integrated into **GitHub Actions** (or other CI/CD pipelines) ensures new configurations are monitored from the first deployment.

Outcome: Rapid feedback on performance regressions or environmental anomalies, mitigating production-level disruptions.

- **Resource & Capacity Forecasting**

Enhancement: Historical usage patterns feed ML-driven forecasting to anticipate peak demands (e.g., during software version upgrades or seasonal cycles).

Outcome: Balanced allocations across global data center estate, minimizing both overprovisioning and sudden capacity crunches.

- **Budget & Cost Allocation**

Enhancement: Usage of metrics at the organizational unit level allow each department to see real costs tied to their workloads (e.g., a dev/test environment versus a productive system).

Outcome: Greater accountability and potential cost savings as departments make more informed scale decisions.

3.6.8 Linking Observability to Broader Goals

Observability is not simply a "tool upgrade," but rather a strategic capability:

- **Digital Transformation:** It underpins the shift to agile, software-defined data centers, ensuring that insights support continuous improvement and innovation.
- **Global Standardization:** Observability fosters consistent processes across numerous data centers, reinforcing best practices and shared standards.
- **Customer Confidence:** Transparent, well-documented data center performance helps assure **clients**, especially those in **regulated industries,** that their mission-critical solutions are supported by compliant, stable, efficient, and eco-friendly infrastructure.

3.6.9 Examples of use cases that can be achieved with the proposed solution:

Category	**Use Case**	**Description**	**Purpose**
Power Monitoring	Power Consumption Monitoring	Monitor Power Consumption Multiple Levels - Floor, Cage, Containment zone, Row, Rack, PDU, Socket kWh)	Ensure efficiency, Avoid overloads & disbalance, track trends, cross-reference, Observer Power Posture

Power Monitoring	Power Load Monitoring	Monitor Power load on Multiple Levels - Floor, Cage, Containment zone, Row, Rack, PDU, Socket (kW)	Ensure efficiency, Avoid overloads & imbalance, track trends, cross-reference, Observer Power Posture
Power Monitoring	Phase Balance Monitoring	Track phase loading and correlate with historical patterns, measure against desired state and maximum capacity	Prevent inefficiencies and ensure balanced power usage against desired state
Power Monitoring	Circuit breaker Monitoring	Monitor and alert on circuit breaker trips or nearing capacity limits	Enable preventative maintenance, avoid downtime to the extent possible
Power Monitoring	Socket-Level Load Analysis	Evaluate individual socket load patterns to detect underutilization or overdraw	Enable capacity planning, reduce energy waste, ensure proper balancing
Environmental Monitoring	Temperature Mapping	Display heatmaps and trends for rack and whitespace temperature	Avoid overheating and optimize cooling strategies
Alerting & incident Management	Threshold based alerts	Configure alerts for power events	Rapid Response for critical deviation or catastrophic failures
Alerting & incident Management	Threshold based alerts	Configure alerts for temperature events	Rapid Response for critical deviation or catastrophic failures
Alerting & incident Management	Critical Event detection	Detect and log SNMP traps for high-priority events (e.g., UPS, PDU failures) (Also possible thrum Redfish or REST API) - Power	Ensure Timely reaction in connection with operational disruption avoidance
Alerting & incident Management	Critical Event detection	Detect and log SNMP traps for high-priority events (e.g., UPS, PDU failures) (Also possible	Ensure Timely reaction in connection with operational disruption avoidance

		thrum Redfish or REST API) - Temperature	
Alerting & incident Management	Incident Correlation	Correlation Power & Environmental metrics to enable RCA	Enhance and Enable RCA + Resolution speed (TTF)
Alerting & incident Management	Rack Utilization and balancing	Visualize rack-level power and cooling usage for balance and optimization	Prevent overloading and improve resource allocation
Integration	Data Lake Integration	Send aggregated metrics into enterprise data lakes for initial analytics usage	Data Aggregation, Reporting, Enrichment & etc.
Security	Embedded Monitoring Units and Device Access Monitoring	Monitor access attempts and configuration changes on Embedded Monitoring Units	security and audit trails
Security	Firmware Compliance monitoring	Track Firmware version and PDU against compliant versions (automatic version alerting & enforcement)	Ensure security compliance, reduce vulnerability risks
Alerting & incident Management	Configuration Delta Detection	Detect deviations between live configurations and desired state stored in CMDB	consistency and compliance with system standards
Compliance Reporting	Energy Consumption Reporting	Create consumption reports for audits / compliance (carbon footprint reporting & etc. if needed)	Meet regulatory requirements and support sustainability
Compliance Reporting	Carbon Footprint reporting	Monitor power usage to calculate DC carbon footprint	Regulatory reporting needs specially in the EU

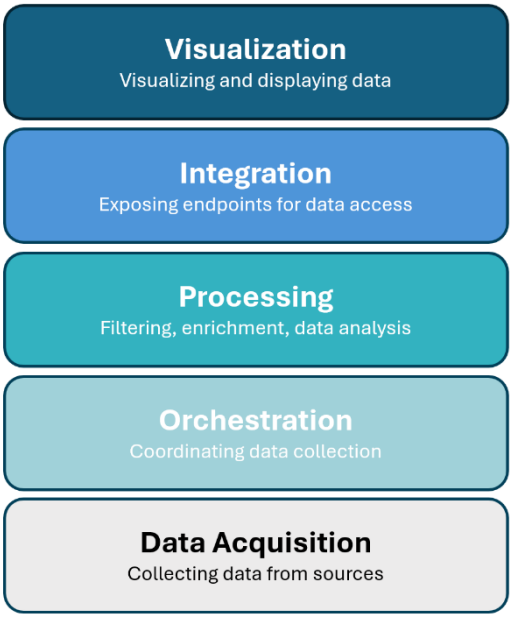
Integration	Automated Configuration Updates	Dynamically updated SNMP polling settings or thresholds based on system changes	up-to-date configuration and aligned with operations, maintenance & demand management
Environmental Monitoring	Humidity Mapping	Display heatmaps and trends for whitespace humidity	Avoid overheating and optimize cooling strategies
Alerting & incident Management	Threshold based alerts	Configure alerts for humidity events	Rapid Response for critical deviation or catastrophic failures
Alerting & incident Management	Critical Event detection	Detect and log SNMP traps for high-priority events (e.g., UPS, PDU failures) (Also possible thrum Redfish or REST API) - Humidity	Ensure Timely reaction in connection with operational disruption avoidance
Compliance Reporting	SLA & Uptime Monitoring	Monitor uptime metrics + Reporting against SLA/OLA	Compliance assurance, reliability

Having mapped the stakeholder landscape and strategic impact, we now explore the underlying technical framework that supports scalable, composable, and interoperable observability.

4 Framework Design

4.1 Logical Architecture

The observability platform is architected around modular, loosely coupled layers that separate concerns across data acquisition, processing, orchestration, integration, and visualization. It is built with open-source components and industry-standard protocols to ensure interoperability, scalability, and enterprise-readiness.



The architecture integrates critical tools and technologies into distinct layers.

A. **Data Source & Protocol Interfaces:**

Protocol / Interface	Purpose
SNMP (v2c / v3)	Polls metrics and receives traps from PDUs, EMUs, circuit breakers, and sensors
Redfish API (REST)	Collects structured telemetry from Redfish-compatible smart power/thermal devices
OpenTelemetry Protocol (OTLP)	Aggregates logs, traces, and metrics from distributed services and agents
REST APIs	Interfaces for CMDB or third-party data pull/push operations

These protocols enable secure, standardized communication with devices across heterogeneous environments.

Data Collection & Ingestion:

- **Telegraf:** Collects SNMP metrics and traps from Embedded Monitoring Units, PDUs, and environmental sensors.

- **Redfish Collector: (Python/REST-based or plugin-based):** Gathers metrics over HTTPS from Redfish-enabled devices.
- **OpenTelemetry Collector:** Ingests logs, traces, and metrics from agents deployed on infrastructure or services.

All collectors need to be configured with polling frequency, authentication, device mapping, retry logic & etc.

B. Orchestration and Deployment

- **Kubernetes (Gardener):** Orchestrates containerized components and ensures scalability including fault tolerance
- **Greenhouse:** Simplified deployment and management of applications including version-controlled deployment configuration among distributed clusters

C. Data Processing:

- **Collector-processors:** Processing and transformation of SNMP and telemetry data
- **Data lake:** Metrics/events storage, indexing, enrichment and querying.

D. Integration

- **CMDB:** Manages inventory, device metadata, desired and current state, providing visualization support.
- **Ticketing Platform:** handles incident response and workflow automation
- **Observability layer and correlation engine:** Receives enriched analytics for data lake and observability analytics.

E. Visualization:

- **CMDB:** Floorplan visualization, heatmap & overview
- **Perses:** Dashboards, data analysis

4.2. Metrics Monitored

Power Metrics:

Metric	Description
PDU Socket Level Consumption	Measures how much power is drawn from each outlet/socket on a PDU (in kWh). Helps track energy use per device.
Phase Balancing	Evaluates whether the electrical load is evenly distributed across all three power phases. Imbalance may reduce efficiency or cause faults.
Power Load Distribution	Tracks how power usage is distributed across racks, cages, and floors. Highlights hotspots or underutilized zones.
Consumption Fluctuations (Peaks)	Highlights sudden power spikes, which may indicate faults, high-load events, or cooling failure responses.

Power Stability	Monitors the consistency of power delivery over time (e.g., voltage/current variation). Instability may signal upstream electrical issues.
Power Capacity Consumption	Measures current consumption against rated capacity of PDUs or circuits. Helps avoid overloads and triggers capacity alerts.

Environmental Conditions:

Metric	Description
Temperature	Real-time and historical readings from rack, row, or room-level sensors. Helps manage thermal profiles and cooling efficiency.
Humidity	Tracks moisture levels within whitespace. Helps prevent equipment corrosion or electrostatic discharge.

Device Health:

Metric	Description
Uptime	Measures the operational availability of PDUs, EMUs, and other monitored devices.
Error Logging	Captures and classifies error events, warnings, or critical system logs from devices.
Device Response Metrics	Evaluates latency, success/failure of SNMP or API queries — signals device health and communication issues.
Fault Telemetry	Detects hardware-specific alerts like breaker trips, thermal failures, or tamper events.

To support such a platform, the infrastructure itself must be modular and dynamic. The next section introduces composable infrastructure and how it enables flexibility and control.

4.3 Data Quality & Normalization Principles

Any operational and fit for purpose observability platform relies on clean, consistent, and structured data to deliver meaningful insights, enable automation, and support downstream decision-making processes. In heterogeneous data center environments where equipment varies by vendor, model, firmware version, and communication protocol, ensuring data quality and semantic consistency becomes foundational to success.

This section outlines the principles and practices ensuring data ingested into the observability pipeline is normalized, consistent, and enterprise-usable across all modules.

4.3.1 Unified Data Model

All incoming telemetry — whether SNMP-based metrics, Redfish payloads, OpenTelemetry streams, or REST API results, is transformed into a unified internal data model (extending OpenTelemetry timeseries model and semantic conventions). This common format allows systems to interpret data uniformly across locations and devices.

Examples of Primary Fields:

- hw.type (e.g., “PDU”, “Embedded Monitoring Units”, “temperature_sensor”)
- hw.name (canonicalized identifier: DC-WDF-CAGE3-PDU-04)
- timestamp (UTC ISO 8601)
- location.path (hierarchical: DC > Cage > Rack > Device)
- hw.status (ok/degraded/failed)
- source_protocol (SNMP, Redfish, OTEL, etc.)

In this example the scheme is enforced in data collection pipelines, ensuring consistency regardless of source heterogeneity.

4.3.2 Naming & Tagging Conventions

Inconsistent naming is one of the primary causes of operational drift and observability blind spots. Therefore, a strict naming and tagging convention is used for all devices and metrics.

Device Names:

- Follows the pattern: Location Code-Room/CageCode-DeviceType-Sequence
- Example: DC-FRA-CG2-PDU-08

Rack Identifiers:

- Should match CMDB entries: RACK-WDF-01-15A

Metric Namespaces:

- Use dot notation with clear hierarchy: power. socket. load_kw, env. temp_c, breaker. status

Tagging:

- All data points include tags for site_id, rack_id, EMU_id, device_vendor, firmware_version, and region
- Tags are critical for filtering, alert scoping, and dashboard generation

Enforcement of naming policies is integrated into CMDB through validation rules and form constraints.

4.3.3 Timestamp Synchronization

- All devices must provide telemetry with timestamps either in UTC or local time zone with offset metadata.
- Embedded Monitoring Units without native timestamp support will have polling timestamps assigned by a collector, with a fixed polling interval.
- **NTP synchronization** is a hard requirement for all management devices, Embedded Monitoring Units, and hosts to prevent false-positive alerts caused by time drift.

4.3.4 Metric Normalization & Unit Handling

Vendors expose metrics in inconsistent units (e.g., watts vs. kilowatts, Fahrenheit vs. Celsius). This platform standardizes units for analytical consistency.

Standard Units:

- Power: kW
- Energy: kWh
- Temperature: °C
- Humidity: %
- Load/Capacity: %

All incoming metrics undergo unit normalization inside Logstash or Telegraf using transformation filters. Any metric that cannot be reliably converted is flagged and optionally dropped, ensuring **data cleanliness over data completeness.**

4.3.5 Data Integrity & Validation

To prevent corrupt or misleading data:

Zero and Null Handling:

- null, NaN, or zero values from known faulty sensors are discarded or flagged via status=unknown

Out-of-Range Detection:

- Thresholds are applied at the edge to drop or flag impossible values (e.g., temperature > 85°C, socket load > rated value)

Rate of Change Validation:

- Sudden changes are compared against prior samples (e.g., power load spike of 300% triggers a review)

These validations ensure that dashboards and automated alerts are not polluted by low-quality or erratic data.

4.3.6 Deviation Correction Model for Quantized Power Telemetry in PDUs

Some PDUs sometimes exhibit telemetry quantization effects, where reported power values are rounded to coarse steps - typically in 0.1 kW increments. This results in significant loss of resolution for low-power devices (<200 W), leading to underreported or zeroed-out power readings. This behavior introduces systematic deviation that hampers monitoring fidelity, capacity planning, and energy reporting accuracy.

4.3.6.1 Observed Behavior & Problem Statement

In example:

- P_{true} to be the actual power consumption at a given time (in Watts)
- $P_{reported}$ to be the power reported by the PDU

For Affected Devices:

$$P_{\{reported\}} = \left\lfloor \frac{P_{\{true\}}}{100} \right\rfloor \times 100$$

This causes:

- $P_{true} \in [0,99] \rightarrow P_{reported} = 0$

- $P_{true} \in [100,199] \rightarrow P_{reported} = 100$

- etc.

This quantization step of 100W (0.1kW) leads to:

- Up to 99W of underreporting per socket
- Aggregate underestimation of load
- Inaccurate power heatmaps and failure to trigger alerts

4.3.6.1 Proposed Correction Model

The correction model leverages more granular telemetry inputs voltage and current to compute apparent power, which avoids the quantization bias present in the reported real power.

4.3.6.1.1 Apparent Power Estimation

Given:

- I : RMS current reported by the PDU (in Amperes)
- V : RMS voltage reported by the PDU (in Volts)

Then the apparent power is $S = V \times I$ [in VA]

To convert to kilowatts: $P_{\text{estimated}} = \frac{S}{1000}$

If Power Factor (PF) is known or can be assumed: $P_{estimated} = \frac{V \times I \times PF}{1000}$

Where PF $\in [0.8, 1.0]$ depending on the device

4.3.6.1.1 Correction *Logic*

We define the corrected power reading:

$$P_{corrected} = \begin{cases} P_{\text{estimated}} \\ P_{reported} \end{cases}$$

If $P_{reported}$ is a multiple of 100W and deviation > $\in$

Where:

- $\in$ is a defined tolerance threshold (e.g., 20W)
- A deviation flag can be raised if $|P_{reported} - P_{estimated}| > \in$

4.3.6.2 Algorithmic Steps

1. Poll SNMP OIDs for:
   - V: Voltage per socket (or bank)
   - I: Current per socket (or bank)
   - $P_{reported}$: Real power per socket
2. Compute $P_{estimated} = V \times I/1000$
3. Compare $P_{estimated}$ and $P_{reported}$
4. If deviation > $\in$, substitute $P_{corrected} = P_{estimated}$
5. Flag deviations for visualization and alerting

Example:

Let:

- V=230 V
- I=0.43 A
- PF=0.95

Then: $P_{\text{estimated}} = \frac{230 \times 0.43 \times 0.95}{1000} \approx 93.9$ W

Assuming:

- $P_{reported} = 0$ W

- $\in = 20$W

Then: $\left|P_{estimated} - P_{reported}\right| = 93.9\ \text{W} > 20\ \text{W} \Rightarrow P_{corrected} = 93.9\ \text{W}$

4.3.6.3 Operational Deployment Notes

Implementing the deviation correction model in a real-world observability platform requires thoughtful integration with existing telemetry pipelines and system architecture. The following considerations guide the operationalization of the model:

Pipeline-Level Correction:

Deviation correction should be applied during the data ingestion phase, prior to storage or visualization. Tools such as Logstash (via Ruby filters), Telegraf (via Starlark or execd processor plugins), or Kafka Streams can be used to calculate estimated power values based on voltage and current telemetry, compare them against reported values, and override where deviations exceed defined thresholds.

Deviation Metadata Management:

Corrected readings must include associated metadata, such as:

- `correction_applied: true/false`
- `original_reported_value`
- `correction_method`
- `deviation_amount (W)`

This metadata ensures transparency, facilitates root cause analysis, and supports auditability, especially in regulatory or billing-sensitive environments.

Power Factor (PF) Handling:

When PF is not explicitly reported by the PDU, estimated real power should be derived using:

- Default values by device category (e.g., compute servers: 0.95, storage: 0.9)
- Operator-defined constants via configuration
- Adaptive heuristics, e.g., mapping PF to outlet type (C13/C19), or based on historical device signatures.

Accuracy Optimization for Critical Use Cases:

For environments requiring high accuracy, such as:

- Energy billing and chargeback models
- Regulatory reporting for CO_2 impact or PUE

The correction model should incorporate cross-referenced measurements from calibrated power meters or rack-level sensors, ensuring reconciliation of estimated

values with known ground truths. This may involve periodic calibration jobs or Bayesian smoothing of noisy SNMP signals.

Alert and Dashboard Integration:

Visualizations (e.g., Kibana) and alerts must reflect both corrected and reported values, especially where large deltas may indicate:

- Firmware bugs
- Device degradation
- Environmental influences (e.g., voltage sag)

This transparency supports operational decision-making and increases trust in observability outputs.

Extensibility and Modularity:

The correction logic should be encapsulated in modular enrichment components decoupled from ingestion logic, allowing independent updates, hot-patching, and vendor-specific adaptations.

4.3.6.4 Limitations and Considerations

While the deviation correction model significantly improves telemetry fidelity in some environments, it also comes with inherent limitations and practical boundaries.

Apparent vs. Real Power Estimation:

The model estimates real power from apparent power, relying on an assumed or configurable power factor (PF). Since PF varies with workload type, power supply efficiency, and transient conditions, the estimation remains an approximation. Without direct PF telemetry, precise restoration of real power is not possible.

Granularity Constraints in Legacy Devices:

Some PDUs lack per-socket or per-bank telemetry, exposing only total or aggregated readings.

In such cases:

- Socket-level deviation correction cannot be applied.
- Aggregated estimations may obscure localized power anomalies.

This limits the model's applicability to device-level or phase-level corrections only.

Firmware/Hardware Override Scenarios

If the vendor releases firmware updates that address the rounding issue or provide enhanced MIB access (e.g., unquantized values, PF telemetry), the correction model should be automatically disabled or bypassed to prevent double-modification and preserve original device fidelity.

Model Suitability Boundaries

The model is intended for operational visibility, anomaly detection, and predictive analytics. It is not certified for regulatory metering, legal energy billing, or forensic analysis where IEC-certified instrumentation is required.

Device-Specific Behavior

Correction logic must account for device-specific variations, such as:

- Different rounding schemes (e.g., ceiling vs. floor)
- Per-phase vs. per-outlet inconsistency
- Manufacturer-specific voltage scaling (e.g., 10x encoded)

Maintenance and Exception Handling

When deviation exceeds a predefined maximum threshold (e.g., 150 W), this may indicate a hardware fault, configuration drift, or telemetry mismatch. In such cases, operators should be alerted to investigate further rather than apply blind correction.

4.3.7 Device Identity Mapping & Source Trust

Each device is uniquely tracked across:

- **SNMP OIDs, MAC addresses**, and **CMDB UUIDs**
- Mappings are established and maintained in **a Device Identity Registry**, aligned with the CMDB (CMDB)

Data ingestion pipelines cross-validate telemetry against known inventory to prevent ingestion from:

- Rogue or Unregistered Devices
- Devices not yet approved in change control
- Devices with conflicting or spoofed identity

4.3.8 Data Enrichment Policies

After validation and normalization, data is **enriched with contextual metadata** pulled from CMDB:

- Rack location (aisle, row, quadrant)

- Device role (core PDU, edge Embedded Monitoring Units, spare)
- Assigned teams or departments
- Maintenance SLA and lifecycle stage (e.g., “End of Support”)

This enables context-aware alerting, SLA-aware incident routing, and cleaner dashboards filtered by business responsibility.

4.3.9 Monitoring Data Health

Finally, the observability stack itself is monitored for data quality indicators:

- **Missing data gaps** per device
- **Anomalous volume drops** (e.g., expected 10,000 samples/hour → currently 2,000)
- **Schema violations** (e.g., unexpected new field or missing tag)
- **Outdated CMDB mappings** triggering errors

Alerts are routed to the team responsible for observability platform health.

4.4 Security Considerations

A robust observability platform is only effective when built on a foundation of strong security. Data center telemetry involves sensitive infrastructure data, access credentials, and operational metadata all of which must be safeguarded to ensure operational continuity, regulatory compliance, and the protection of global data center landscape.

This section outlines the core security design principles, implementation strategies, and integration requirements for the observability platform.

4.4.1 Zero Trust Architecture

The platform adopts a Zero Trust model — no component is implicitly trusted, regardless of whether it resides inside or outside the network boundary. Every communication, user, or process must prove its identity and authorization at every interaction.

Key Aspects:

- **Micro segmentation** of telemetry collection, processing, and visualization services to reduce lateral movement risks.
- **Mutual TLS (mTLS)** for all internal services - external ingress points use TLS 1.2+ with strong cipher suites.
- **Service identity enforcement:** Every node, agent, and pipeline component are authenticated via service accounts or certificate authorities.
- **Policy enforcement points** placed at ingress (API gateways, reverse proxies) and internal traffic chokepoints.

4.4.2 Authentication & Authorization

Every interaction within the platform (e.g., device polling, data ingestion, dashboard access, config management) must undergo strict authentication and role-based access control (RBAC).

Components:

- **CMDB & data lake:** Integrated with corporate IAM (e.g., Identity Authentication Service) using SAML/OAuth2.
- **Data collectors:** Service accounts with least privilege for polling and transformation.
- **Kubernetes Cluster:** RBAC roles scoped to namespace and workload level secrets stored using sealed secrets or Vault.
- **Fine-grained RBAC policies defined per tool** (e.g., Kubernetes, CMDB, visualization dashboards).
- **Separate human access** (e.g., analyst dashboards) from machine access (e.g., polling agents, CI/CD systems).
- **Support token-based access** with automatic expiration for RESTful APIs or Redfish endpoints.

4.4.3 Secure Telemetry Ingestion

Given the reliance on SNMP and/or Redfish:

SNMP:

- Prefer SNMPv3 (authentication + encryption) over SNMPv2c wherever supported.
- Community strings must be rotated regularly and stored encrypted (e.g., Kubernetes secrets or Vault).
- SNMP traps are received on dedicated secure channels with strict firewall rules and traffic filtering.
- Use trap filtering and firewall whitelisting to restrict inbound telemetry.
- Rotate community strings or user credentials every 30–90 days.

Redfish API:

- Use HTTPS/TLS-only endpoints.
- Token-based authentication where possible, with short-lived access tokens.
- Device certificates must be validated against trusted internal certificate authorities.
- Validate API inputs against schemas to prevent injection attacks.

4.4.4 Data Integrity and Tamper Detection

Data ingested into ELK data lake and cross-referenced via CMDB is cryptographically hashed at rest to ensure integrity.

Techniques:

- Use Elasticsearch's built-in support for immutable indices with timestamped logs.
- Implement checksum validation during log ingestion for critical SNMP traps (e.g., PDU faults, breaker trips).
- Audit trails must be written to append-only storage for critical infrastructure components.

4.4.5 Role-Based Access Controls (RBAC) & Segregation of Duties

Each stakeholder group (e.g., DC Ops, RunOps, Finance, Security) is granted access only to the data and dashboards relevant to their responsibilities.

Governance:

- Read/write segregation (e.g. admins vs. viewers).
- Configuration drift alerts can only be acknowledged or overwritten by authorized infrastructure engineering roles.
- Visualization dashboards are scoped by department/team with granular index-level permissions.

4.4.6 Secrets & Credential Management

All secrets (SNMP credentials, Redfish API tokens, webhook keys, kubeconfigs, GitHub Actions tokens) must be centrally managed, encrypted, and rotated.

Recommended Tools:

- Vault or Kubernetes Secrets (with Sealed Secrets for GitOps).
- GitHub Actions: Avoid storing secrets in plaintext YAML - use GitHub Secrets and access them at runtime.

4.4.7 Secure CI/CD Pipelines

Automation workflows (e.g., GitHub Actions) that deploy Helm charts, update cluster configurations, or manage monitoring thresholds must be secured end-to-end.

Controls:

- Use branch protection rules to restrict changes to production pipelines.
- Require code reviews for CI/CD configurations.
- Monitor CI logs for secrets exposure and enforce secure linting policies.

4.4.8 Monitoring Platform Security

The observability platform must monitor itself to detect anomalies, intrusions, or unauthorized configuration changes.

Examples:

- SNMP traps for Embedded Monitoring Units access events pushed to data lake and analyzed with ML for unusual patterns.
- CMDB configuration deltas (desired vs. actual) logged and retained for post-incident forensics.
- Telemetry from Kubernetes control plane (e.g., API server logs, etcd access logs) analyzed via OpenTelemetry collectors.

4.4.9 Compliance & Audit Readiness

All telemetry and configuration data must support internal and external audits, particularly for frameworks such as:

ISO/IEC 27001, NIS2 (EU), EN 50600 / EN 50701, German EnWG (Energy Industry Act)

Controls:

- Immutable logs with retention policies (e.g., 1 year for critical systems).
- Automated monthly audit reports show observability platform changes, alerts, escalations.
- Integration with Ticketing Platform GRC for ticket traceability and workflow documentation.

4.4.10 Network and API Security

- Firewalls and segmentation restrict telemetry flows to only approved source/destination pairs.
- Rate limiting and DoS protection applied on API endpoints and SNMP trap receivers.
- Input validation and schema enforcement at ingestion points to prevent malformed or malicious payloads.

Optional Enhancements (Pluggable per Maturity Level)

- **SIEM integration:** Feed telemetry alerts into enterprise SIEMs (e.g. Enterprise Threat Detection).
- **Multi-Factor Authentication** (MFA)
- **Behavioral analytics:** Apply anomaly detection to telemetry patterns for early breach detection.

4.5 Data Lifecycle Management and Compliance Considerations

As telemetry data volumes grow across the global data center landscape, managing the lifecycle of observability data is critical. Effective lifecycle management ensures compliance with regulatory standards, optimizes storage costs, and supports high-performance analytics.

This platform enforces structured lifecycle practices for telemetry data, aligned with enterprise policies and European Union regulations such as GDPR, EnWG, and the Ecodesign Directive & etc.

4.5.1 Retention and Archiving Strategy

Telemetry data is categorized based on type, criticality, and operational usage. The following table outlines **standard** retention policies and archiving behavior:

Data Type	Retention Period	Archival / Rollup Behavior
Power metrics (e.g., kW, kWh per socket/PDU)	12 months (standard)	Daily aggregates after 30 days
Environmental metrics (temperature, humidity)	6–12 months	Monthly aggregation and compression
Alerts, Events, SNMP Traps	3–6 months	Indexed for correlation - optionally archived
Configuration Snapshots (CMDB)	12–24 months	Immutable snapshots retained for audit compliance
User & dashboard config logs	3 months	Anonymized, non-critical, deleted after expiration

Lifecycle tags are applied at ingestion time to support automated transitions in Data Lake (ELK Stack).

4.5.2 Data Classification and Tagging

All telemetry data is tagged with lifecycle metadata for automated policy enforcement:

- **Data type:** (e.g., power, temperature, water, CO_2)
- **Sustainability relevance:** (e.g., CSRD, ERF, Scope 2)
- **Jurisdictional tagging:** (e.g., DE, FR, EU-wide, non-EU)
- **Confidentiality level:** (e.g., internal, public-facing, audit-only),
- **Lifecycle status:** (e.g., real-time, archived, export-ready).
- **Origin location:** (e.g., DC-WDF-FL1-RACK12)

- **Business retention class:** (e.g., short-term, audit, long-term)

Integrate tagging directly at the point of ingestion and propagate through data lake or CMDB overlays for full pipeline visibility.

Tags need to support filtered queries, dashboard scoping, and compliance-aligned rollup or removal workflows.

4.5.3 GDPR and Regulatory Compliance Considerations in respect to data

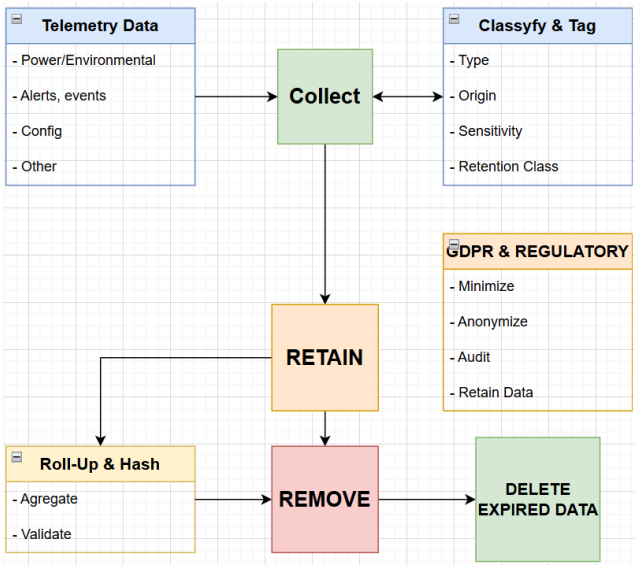
The platform should adhere to data handling best practices in accordance with:

- GDPR (EU 2016/679)
- German EnWG (Energy Industry Act)
- EU Ecodesign Directive
- EU Code of Conduct on Data Centre Energy Efficiency

Key compliance measures:

- No personal data (PII) is collected or stored included such that may identify a person from naming of accounts, digital identities & etc.
- Anonymization and data minimization principles are applied.
- Logs and telemetry for regulated energy and carbon reporting are retained by applicable guidelines.
- Immutable logs support audit trails and forensic requirements.

Example Diagram:



4.5.4 Data Integrity During Retention

To maintain the legal admissibility and operational utility of telemetry over time, integrity must be enforced through:

- **Hash chaining or checksum validation** at the archival layer (e.g., for immutable snapshots),
- **Clock synchronization** across telemetry-producing agents (NTP/NTS enforcement),
- **Tamper-evident metadata**, such as automated origin tagging and time-sequencing.

This is particularly relevant for CSRD or ISO 50001-aligned energy records, where historical traceability may be audited several years post-ingestion.

Compliance-Centric Design Goals

Compliance Requirement	Lifecycle Design Feature
CSRD/ESRS monthly disclosures	Monthly rollup snapshots of PUE, ERF, Scope 2 telemetry, with CSV export capability
Article 12 EED (annual uploads to EU database)	Year-based data extraction schemas from the data lake (e.g., Kibana Saved Queries or API-bound Dashboards)
Climate Neutral Pact KPIs	Rolling year-on-year dashboards with built-in deltas and threshold alarms for self-audit use
EU Taxonomy DNSH proof	Tag historical telemetry by environmental objective and extract as evidence for financing documentation

4.6 Enterprise Integration Expectations

The observability platform is designed and intended to integrate with broader enterprise ecosystems, ensuring data consistency, insights, and alignment with operational workflows across internal tools and external service platforms.

This section outlines key integration points, governance responsibilities, and expected data flows across systems such as CMDB, Ticketing, observability layer + correlation engine, and optional enterprise observability platforms.

4.6.1 Integration Overview

The platform both produces and consumes telemetry, CMDB, and alert data, aiding timely decisions and long-term strategic analysis.

System	Purpose	Integration Method
CMDB	Asset registry, location tagging, config state, visualization of reports	Bi-directional sync (via API / webhooks)

Ticketing Platform	Incident management, alert correlation, audit	REST API integration Data Lake
Data Lake	Data lake for enrichment, analytics, history	Central ingestion e.g. OpenTelemetry receiver or dedicated API
Observability layer and correlation engine	Company-wide observability analytics backbone	API-based forwarding from Data Lake
Alerting Framework	Rule-based alert correlation & suppression	Connected through OpenTelemetry + metadata tagging
Optional SIEMs	Security analytics and event management	Log forwarding with tag filters

4.6.2 CMDB - Source of Truth Alignment

CMDB should provide:

- **Inventory Management:** Data of racks, PDUs, Embedded Monitoring Units, breakers, outlets, sensors, and logical grouping.
- **Configuration Metadata:** Tracks firmware, desired state, vendor mappings, and lifecycle data.
- **Desired State Management:** tags and metadata are used for additional definition of alert thresholds, polling configurations, and compliance rules.

Visualization Plugin Integration:

- Supports floorplan visualization, heatmaps, and topology mapping via integrated plugins.
- Enables operations teams to graphically navigate the data center environment, quickly identify issues (e.g., overheating rack, power imbalance), and correlate telemetry visually.
- Dashboards can reflect telemetry overlaid on physical layouts or rack elevations.

All telemetry collected is cross-referenced against CMDB:

- Device ID and location → used to anchor telemetry context
- CMDB-to-live mapping → enables drift detection
- Tagging and metadata → supports enriched alerting and ticket context
- By enriching telemetry with CMDB metadata and exposing spatial context, the platform supports real-time, intuitive troubleshooting and planning workflows.

4.7 Governance and Change Control (non-technical)

An observability platform is only as effective as the operational policies and ownership structures supporting it. To ensure maintainability, consistent data quality, and business alignment, this section defines non-technical governance responsibilities across stakeholders.

This includes control of thresholds, alert logic, versioning, and data consumption standards.

4.7.1 Governance Scope

Governance covers three key domains:

- **Configuration Control:** Who defines thresholds, tags, alert logic, polling intervals, etc.
- **Platform Evolution:** Who owns the lifecycle of dashboard updates, CMDB schema extensions, and CI/CD changes.
- **Operational Ownership:** Who reacts to alerts, investigates data anomalies, and tunes observability workflows.

4.7.2 RACI Matrix: Platform Ownership (example)

Activity / Function	RunOps / NOC	Infra Architecture	Platform Team	Security	Infra Engineering (Cloud/DC)
Define alert thresholds (power, temp, humidity)	R	A	C	C	I
Manage SNMP/API polling & intervals	C	R	A	I	C
CMDB schema extensions	I	A	R	C	C
Dashboard versioning	C	R	A	I	C
Ticketing Platform integration tuning	A	I	R	I	I
Credential rotation (SNMPv3, API tokens)	I	I	C	A	R
Compliance reporting data definitions	C	C	I	A	I

Table 4: Legend: R = Responsible / A = Accountable / C = Consulted / I = Informed

4.7.3 Example Change Management Policy

Changes to observability configurations (e.g., thresholds, polling, data pipelines) must follow documented change control procedures:

- **Proposal Phase:** Changes proposed via GitHub Pull Request or internal ticketing.
- **Review Phase:** At least two stakeholder teams must approve (e.g., Platform + RunOps).
- **Staging Validation:** New configs tested in non-production data lake and CMDB environments.
- **Production Release:** Via Helm or GitHub Actions, with change log updated.
- **Rollback:** Each change includes rollback procedure and validation window.

4.7.4 Operational Ownership by Platform Zone

Platform Zone	Owning Team	Change Cadence	Example Artifacts
Data collectors	Platform Engineering	Weekly	SNMP OID changes, new polling targets
Data lake	Observability Platform Team	Bi-weekly	New field mappings, tag parsing rules
CMDB	Infrastructure Architects	Monthly	New device types, rack location updates
Alerting	Platform Team + Ops	On demand	Alert tuning, uptime views, KPI visualizations
Ticketing Workflows	RunOps	Quarterly	Alert suppression, escalation chains

4.7.5 Configuration Drift & Change Automatic Detection Example

To detect unauthorized or accidental configuration changes:

- Asset Information and CMDB desired state is compared nightly to current telemetry.
- Drift reports are auto generated and routed to **owners**.

Examples:

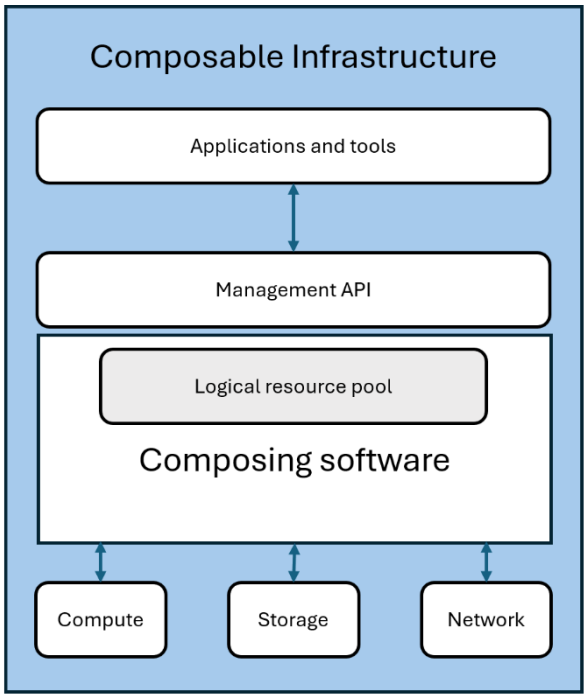
- Rack moved without CMDB update
- Device firmware mismatch
- Unexpected polling interval change
- Alerts are logged and tracked as potential compliance issues.

5 Understanding Composable Infrastructure

5.4 Overview

"Composable" represents a service-centric model where a wide range of resources are rapidly reassigned to accommodate service components. It integrates software-defined capabilities into hardware elements to streamline and automate administrative tasks involved in deploying and managing disassembled infrastructure.

In brief, “Composable infrastructure is an information technology framework where the physical resources are treated as services.”



5.4.1 Hardware Layer

The Hardware layer is the foundational components targeted components in this case **Embedded Monitoring Units, PDU’s, Outlets, Brakers, Phases, Temperature** and **humidity sensors** that form the basis of the infrastructure.

5.4.2 Composable Software Layer

The Composable software layer assists as an abstraction layer for the physical components, arranging them into logical resource pools that can be accessed through the **API, SNMP** or other data mechanisms. This software is equipped with programmable, configurable, and self-correcting functionalities. It can autonomously orchestrate the essential logical resources to meet specific requirements. It has the capability to utilize templates that offer preconfigured setups tailored for specific use cases. This layer relies heavily on software-defined control.

5.4.3 API, SNMP Layer

The API and SNMP play a crucial role by enabling access to the hardware resources within the infrastructure. It acts as a consolidated interface for executing a wide range of operations, encompassing tasks like reporting, alerting, searching, managing inventory, provisioning, conducting updates, and performing diagnostics.

5.5 Application or End-User Usage Layer

The application layer refers to the topmost layer in the architecture, where end-user applications or services interact with and utilize the underlying infrastructure and resources.

5.6 Core Principles

The platform is built on several core principles that define its approach to data center monitoring, observability, architecture and resource management. These principles help to understand and implement the platform effectively.

Principle Name	Description
Resource Pools	Infrastructure abstracts physical hardware components, such as Embedded Monitoring Units, PDU's, Sockets, Sensors, breakers & etc., into resource pools. Resource pools are established by aggregating these hardware resources, thereby enabling their availability.
Software-Defined Control	Infrastructure and platform are built upon software-defined technologies for its control and management. The management and availability of resources controlled by software rather than being tightly bound to specific hardware configurations.
Dynamic Allocation	Resources can be allocated dynamically to different target objects or devices as needed, and these allocations can be adjusted in real-time.
API-Driven Management	This solution aims to provide APIs (Application Programming Interfaces) that allow administrators to programmatically manage target devices, addressing and data resources. API-driven management enables automation, orchestration, and seamless integration with other IT management systems.

Resource Isolation	Composable infrastructure enables resource isolation, ensuring that each workload or application has dedicated resources for performance, security, and compliance purposes. Resource isolation is achieved through software-defined resource allocation.
Elasticity and Scalability	Composable infrastructure is designed to be elastic and scalable, allowing organizations to easily scale up or down in response to changing demands. This scalability is achieved by adding or removing resources from the available pools.
Automation and Orchestration	Automation is a central principle of composable infrastructure, enabling the execution of predefined tasks and workflows without manual intervention. Orchestration coordinates the allocation and configuration of resources to meet specific requirements.
Resource Efficiency	Aims to optimize resource utilization, minimizing waste and underutilization. By efficiently allocating resources so the organizations can reduce both capital and operational costs.
Flexibility and Adaptability	Flexibility and adaptability are key principles, ensuring that the infrastructure can respond to changing business conditions and technology requirements.
Interoperability	Ensures interoperability, enabling different systems, devices, and applications to work together seamlessly. This principle supports the integration of various technologies and platforms, allowing for cohesive operation and communication across diverse environments. This is achieved using open standards, protocols, and APIs, enabling integration across heterogeneous environments and fostering a cohesive ecosystem.

6. Implementation Strategy & Practical Considerations for Real-World Adoption

The successful implementation of a data center observability platform requires more than selecting technical components or deploying agents. It is a cross-disciplinary endeavor involving infrastructure engineering, operations management, regulatory compliance, platform governance, and strategic business alignment. Organizations differ in maturity, architecture, and constraints - hence, a single prescriptive approach is neither feasible nor desirable.

This section provides a practical, adaptable framework that enables organizations, whether operating a hyperscale data center or a hybrid co-location site, to craft their own observability roadmap. The guidance herein is not tool-specific, but concept-driven, combining technical realism, organizational foresight, and regulatory awareness.

Rather than asking "**What tool should I deploy first?",** we encourage stakeholders to ask:

- *What decisions must be observable to improve outcomes?*
- *Which teams rely on telemetry for critical processes?*
- *How to connect data collection with governance, compliance, and strategic impact?*

We address a broad spectrum of roles:

- **Operators** will need guidance on integrating telemetry with timely workflows.
- **Architects** will need information on how to design resilient, layered observability stacks.
- **Compliance teams** will need aid on how metrics map to regulatory obligations.
- **Executives** will need insight into how observability supports business continuity and ESG targets.

The subsections below follow a maturity-aware progression, designed to help any organization assess its context, map capabilities, structure architecture, and ensure long-term alignment between systems, users, and regulatory demands.

6.1 Organizational Readiness – Contextual definition

Before deploying any observability platform, organizations must prepare beyond just infrastructure and software readiness. The success of observability depends on whether the organization is structurally and strategically aligned to use it effectively. This means clarifying who owns what, what compliance obligations exist, and how observability insights will influence real decisions.

This section introduces a foundational lens to assess readiness. Rather than rushing into tool selection or protocol integration, stakeholders should examine internal alignment across technical, organizational, and compliance domains.

Design Insight: An observability platform implemented without organizational readiness risks becoming an underutilized data silo. Without clarity on thresholds, roles, and escalation paths, alerts may be ignored, and telemetry may never reach the right decision-makers.

Framing Questions by Dimension

The table below offers a guided peek to facilitate cross-functional discussions. Each dimension uncovers crucial insights that determine whether observability will be useful or merely visual.

Dimension	Guiding Questions	Why It Matters
Organizational	Who owns the racks, PDUs, and EMUs? Who approves observability changes?	Clarifies accountability and change control boundaries.
Technical	What telemetry exists today? Which protocols (e.g., SNMP, Redfish) are supported?	Informs tool compatibility and defines the potential data pipeline.
Process	Are there defined alert thresholds? Is there a CMDB integration or an incident response process?	Ensures observability aligns with operational workflows.
Compliance	What regulations apply (e.g., EED, CSRD)? What metrics must be tracked or reported?	Enables early planning for regulatory alignment and audit readiness.
Data Maturity	Are there consistent naming conventions? Is the asset metadata clean, current, and structured?	Supports accurate alerting, filtering, and long-term automation.
End Users	What do different user groups (e.g., operations, planning, compliance, executives) expect to see or act on?	Ensure dashboards and alerts are purpose-built, not generic.

Example: Multi-Stakeholder Kickoff in a Co-Located Environment.

In a large-scale co-located facility shared by multiple clients, observability readiness begins with determining who owns what equipment and telemetry rights. If the hosting provider controls the infrastructure but tenants demand energy visibility, the observability platform must be positioned as a shared service with clear data contracts.

Organizational Need: Align co-location operator and tenant expectations.

Compliance Need: EnWG in Germany mandates transparency for facilities ≥ 500 kW.

Process Alignment: Establish shared incident flows and visibility scopes for tenant dashboards.

Maturity Reflection

Even sophisticated organizations may have gaps.

For example:

- Is telemetry polled but unused due to lack of thresholds?
- Do security teams have visibility into telemetry access control?
- Does procurement know which assets can produce observability data?

Operational Implication: Readiness assessments often uncover unowned telemetry devices that produce useful metrics but are not mapped to any team's responsibility. Making these relationships explicit is a critical precondition for platform sustainability.

6.2 Implementation Planning Framework

Data center observability cannot be treated as a plug-and-play deployment. It is a **strategic capability** that matures over time through structured planning, iterative deployment, and continuous refinement.

This section introduces a four-phase implementation framework designed to help stakeholders translate strategy into actionable delivery while accommodating site-specific realities, tooling constraints, and evolving regulatory demands.

The framework is flexible enough to suit diverse organizational profiles—from a single-edge site to multi-region enterprise networks—yet anchored in universal implementation disciplines such as governance, telemetry alignment, and value mapping.

We propose organizing the implementation into four tightly interlinked phases. These are not simply sequential steps, but **interdependent cycles** that may iterate as business needs and technical capabilities evolve.

Phase	Objective
1. Contextual Definition	Clarify the intent, operational boundaries, and business drivers of the observability initiative.
2. Capability Assessment	Map the current-state telemetry environment, available protocols, and data quality baselines.

3. Platform Architecture	Define the logical structure, governance model, data flows, and integration touchpoints.
4. Operational Integration	Embed observability into daily operations, compliance frameworks, and long-term planning cycles.

Each phase is elaborated in more detail below with corresponding examples and success criteria.

6.2.1 Phase 1 Contextual Definition

Goal: Define the **"Why," "What," and "Who"**

The first phase addresses **intentionality**: What problems are we solving? Who benefits? What outcomes define success?

Key Activities:

- **Use Case Mapping:** Examples include power monitoring at the socket level, ambient temperature alerts, or firmware drift detection.
- **Stakeholder Alignment:** Identify data consumers and sponsors—data center ops, cloud platform teams, RunOps/NOC, sustainability, finance.
- **Driver Analysis:** Define the legal, operational, or strategic reasons behind the platform. These could include EU regulatory compliance, internal audit mandates, or uptime SLAs.

Practical Tip: Use a stakeholder canvas to record expectations, risks, and dependencies for each stakeholder group. This prevents technical solutions from outpacing organizational needs.

Example: In a newly built data center intended to serve regulated industries, the primary business driver may be auditability of temperature and energy metrics. Thus, observability success is measured by compliance report generation, not only uptime visualization.

6.2.2 Phase 2: Capability Assessment

Goal: Understand Your Starting Point

Before architecture decisions, understand what telemetry **already exists**, its reliability, and how to integrate it.

This phase evaluates the technical landscape to determine telemetry potential and integration feasibility.

It includes:

- **Asset Taxonomy:** Classification of PDUs, EMUs, sensors, devices and racks based on telemetry exposure (SNMP v3, Redfish, etc.).
- **Protocol Compatibility Mapping:** Identification of supported communication protocols and data export mechanisms.
- **Data Availability and Quality:** Review of existing naming conventions, time-stamp policies, and data completeness.

Example: In a multi-vendor environment with legacy PDUs supporting only SNMP v2c, a transformation plan may involve standardizing telemetry through intermediate collectors with normalization logic.

6.2.3 Phase 3: Platform Architecture

Goal: Define the Engine Behind Observability

This phase lays the foundation of how telemetry is ingested, transformed, stored, enriched, visualized, and acted upon.

The observability stack is logically structured to enable modularity, scalability, and traceability.

Key considerations include:

- **Data Ingestion Design:** Use of collectors (e.g., Telegraf, OpenTelemetry) to receive telemetry, normalized via schema enforcement.
- **Contextual Enrichment:** Tagging of data with asset metadata (location, vendor, lifecycle stage) from an authoritative CMDB.
- **Data Tiering:** Categorizing telemetry into operational, compliance-critical, and archival layers.
- **Governance Principles:** Ownership of thresholds, alert rules, and lifecycle policies across domains.

Best Practice: The architecture should separate telemetry tiers (e.g., operational alerts vs. sustainability reports) and use metadata to drive filtering and alert scoping.

Example: For operations, architectural uniformity is maintained via GitOps-based deployment, with localization handled through metadata-driven dashboards and region & system-specific alert profiles.

6.2.4 Phase 4: Operational Integration

Goal: Make Observability Actionable and Sustainable

No observability platform is complete until it is used by actual stakeholders and supports real-world decisions.

The final phase ensures the observability platform is embedded in workflows and contributes to business continuity, strategic planning, and regulatory readiness.

Key Practices include:

- **Alert Management:** Integration with incident management systems (e.g., Ticketing), including role-based escalation paths.
- **Drift Detection:** Periodic comparison of real-time telemetry with desired state declarations in the CMDB.
- **KPI Reporting:** Automated export of PUE, renewable share, energy footprint per zone, mapped to EU regulatory requirements.
- **Audit Support:** Immutable logs, version-controlled configurations, and change-tracking systems enable compliance with ISO 27001, NIS2, and CSRD.

The implementation framework is not about technology first - it's about context, alignment, and structured deployment. Without a shared purpose and maturity-aware approach, observability risks becoming fragmented or underused.

6.3 Implementation Scenarios

Not all organizations begin their observability journey from the same starting point. Infrastructure age, protocol diversity, organizational structure, regulatory demands, and internal capabilities all shape what is possible, and what is pragmatic.

This section presents two perspectives on implementation scenarios:

Maturity Progression ("Crawl–Walk–Run–Fly"): How organizations typically evolve over time in observability sophistication.

Situational Profiles: Representative deployment conditions (e.g., legacy, greenfield, sustainability-driven) and how observability must adapt accordingly.

By structuring implementation scenarios in this dual view, we help organizations benchmark where they are and plan realistic next steps.

6.3.1 Maturity Scenarios – How to Grow Over Time

Stage	Telemetry Scope	Process Maturity	Business Impact
Crawl	PDU and temperature data from a single site, basic SNMP polling	Manual threshold checks, no CMDB linkage	Partial visibility, limited response automation, frequent alert fatigue
Walk	Socket-level and EMU telemetry across multiple sites	Initial CMDB integration, static alert rules, basic tiering	Improved regional oversight, SLA monitoring, reduced noise
Run	End-to-end telemetry: PDUs, racks, sensors,	Role-based access, CI/CD pipelines, structured escalation	Predictive maintenance, SLA & ESG conformance, continuous improvement

	firmware, configuration drift detection		
Fly	Cross-site analytics with ML, timely signal enrichment, full compliance instrumentation	GitOps observability, closed-loop automation, regulation-aware dashboards	Enterprise-wide energy optimization, adaptive compliance, business-aligned observability

NOTE: *Each stage introduces new system dependencies (e.g., data enrichment at "Walk" requires CMDB reliability), and new stakeholders (e.g., ESG officers become relevant at "Run"). The maturity journey is as much about governance and culture as it is about data pipelines.*

6.3.2 Implementation Conditions – Real-World Scenarios

These scenario profiles reflect real-world deployment challenges and constraints, helping clients frame observability in a practical, context-sensitive way.

Scenario	Characteristics	Observability Focus
Legacy Co-Location Environment	Older hardware, limited SNMPv2 support, manually maintained asset records	Establish a minimum viable telemetry, normalize inputs, enable socket-level alerting with collector throttling
Greenfield Deployment (Edge or Cloud)	Brand-new build, Redfish and OpenTelemetry native, KPI-driven design	Implement full observability stack from day one, enforce structured tagging, track renewable energy input
Multi-Tenant DC with SLA Commitments	Shared infrastructure, customer-specific SLAs, carbon tracking required	Tenant-tagged telemetry streams, SLA-aware alerts, customer-facing dashboards, traceable KPI histories
Sustainability Compliance Site	Subject to EU taxonomy, water, cooling, and renewable targets	Integrate facility data, enforce ESG metric capture, link drift detection with sustainability controls

Clarification:

- **Legacy scenarios** often require architectural compromises (e.g., less frequent polling, read-only integrations).
- **Greenfield builds** offer ideal conditions for reference architecture deployment but also carry a blank-sheet burden - no legacy constraints, but also no institutional experience or historical data.
- **Multi-tenant environments** introduce the challenge of data segmentation, ensuring observability is actionable and compliant for each tenant without risking leakage or misattribution.
- **Sustainability sites** push observability beyond IT into facility, energy, and ESG domains, demanding interoperability with non-standard systems (e.g., chillers, water meters, or power purchase agreements).

Best Practice: Organizations operating under multiple conditions (e.g., a mix of legacy and edge sites) should define site-level observability tiers. Each site is then monitored according to its own capabilities and strategic value, avoiding one-size-fits-all deployments.

6.4 Observability Success Drivers

An observability platform's value is not defined merely by its tooling or data ingestion rates. True success is measured by its ability to align with organizational goals, regulatory demands, and **operational realities**. This section highlights the underlying drivers that determine whether observability efforts scale from pilot projects to institutionalized platforms.

These drivers are not "features", they are architectural and governance enablers that define the sustainability, relevance, and impact of observability across the enterprise.

The effectiveness of implementation is driven by several organizational and structural success drivers:

Driver	Why It Matters	How to Implement
Telemetry–CMDB Convergence	Context-aware insights, root cause analysis	Bi-directional sync with CMDB, enriched alerts
Granularity by Purpose	Cost-efficient and actionable monitoring	Align polling intervals and retention per use case
Lifecycle Governance	Compliance, auditability, clarity	Define policies for data tiers, access roles, and version control
Stakeholder Visibility	Adoption and operational value	Role-specific dashboards and reporting outputs

Data Health Monitoring	Integrity, trust, quality assurance	Automated anomaly detection on telemetry pipelines

6.4.1 Telemetry-CMDB Convergence

Aligning live data with asset metadata ensures consistency, drift detection, and accurate alerting. Observability becomes fragile when telemetry exists without metadata context.

Example: Socket-level power readings are useful, but without knowing which rack, which team, or which environment the data belongs to (via the CMDB), the insight is incomplete.

Best Practice:

- Ensure every telemetry stream is mapped to a canonical asset in the CMDB
- Enable bi-directional sync so that configuration changes (e.g., PDU swap) update telemetry logic automatically.
- Use CMDB tags (e.g., lifecycle stage, business unit, vendor) to enrich alerts and support role-specific dashboards.

6.4.2 Granularity by Purpose

Tailoring data collection depth to the use case (e.g., rack-level for capacity vs. zone-level for compliance). Not all metrics require the same level of precision or frequency.

Example: *A rack-level temperature reading might suffice for compliance, but socket-level load data is necessary for incident prevention***.**

Consideration:

- Granular telemetry (e.g., socket load every 5 seconds) can be valuable, but expensive to store and analyze.

Align granularity and retention policies with the purpose:

- Regulatory → summarized and archived
- Operational → real-time and short retention
- Predictive → enriched with tags, used for training models

6.4.3 Lifecycle Governance

Retention, archival, and access policies ensure observability aligns with regulatory and operational norms. A successful observability platform has clear policies for:

- **Retention:** How long is the telemetry kept?
- **Archiving:** Which data is rolled up? Which data is deleted?

- **Access:** Who can view or modify dashboards? Who owns the configuration?
- **Versioning:** Are dashboard and schema changes tracked?

NOTE: *This is especially critical in regulated environments, where reporting periods and audit trails are mandatory. Many organizations fail audits not because they lack the data, but because they cannot prove its completeness or origin.*

6.4.4 Ease of Use

Role-specific dashboards and controls support both technical users and executive sponsors. A technical sound platform is irrelevant if its output is unreadable to key stakeholders.

Success Factor:

- Tailor dashboards, alerts, and reports to audience needs:
- DC Ops → root cause analysis
- Procurement → utilization reports
- Sustainability → CO_2 per rack
- Executives → compliance and ROI summaries

Good observability is not about showing everything, it is about showing the right thing to the right person in the right format.

EXAMPLE: A capacity planner might want **phase balance** per row over time, while a sustainability officer needs **to aggregate kWh** per building per month with renewable split.

6.4.5 Trust and Data Health Monitoring

Even the best observability platform must observe itself.

Critical Practices:

- Monitor data gaps, unexpected value patterns, and telemetry volume anomalies.
- Alert on schema violations (e.g., missing timestamps, invalid tags).
- Track pipeline drift, where enrichment logic may silently fail due to upstream changes.

This is the foundation of trustable observability - systems where telemetry is not just ingested, but audited and verifiable.

6.5 Interoperability and Compatibility Considerations

Interoperability is not just a technical challenge - it is a business enabler. As data centers evolve into federated, compliance-driven, and customer-facing ecosystems, ensuring compatibility across heterogeneous systems becomes foundational to platform sustainability and auditability.

Modern data center environments are characterized by their diversity: multiple vendors, evolving firmware baselines, protocol mismatches, and coexistence of legacy and next-generation equipment. An observability platform must therefore not only operate across this heterogeneity, but it must also bridge it intelligently.

In large-scale, multi-vendor, or hybrid data center environments, interoperability and compatibility are not nice-to-haves they are essential for platform viability.

Observability needs to:

- Diverse protocols and data formats
- Varying firmware maturity and telemetry granularity
- Mixed ownership models (e.g., co-location, cloud edge, internal IT)

This section outlines how to ensure technical interoperability while minimizing fragmentation, avoiding data silos, and achieving coherent visibility across heterogeneous infrastructure.

6.5.1 Protocol Interoperability

Data centers contain devices with vastly different capabilities from 10-year-old SNMPv2c PDUs to next-gen servers supporting Redfish and OpenTelemetry.

While modern architectures may favor Redfish or OpenTelemetry, SNMP remains dominant in legacy and co-location environments. This duality requires observability platforms to be “bilingual” - capable of ingesting structured, modern APIs while gracefully handling older, less secure formats.

Telemetry data is typically sourced from devices supporting a mix of:

SNMP (v2c, v3): Ubiquitous but varies in implementation and security (v3 preferred for encryption/authentication).

Redfish API (REST-based): Common in newer hardware for power/thermal telemetry.

OpenTelemetry: Typically used for application-level metrics but increasingly relevant for infrastructure observability.

Custom REST APIs: Proprietary interfaces exposed by vendors or co-location providers.

Syslog & SNMP Traps: Event-driven telemetry requiring near-real-time ingestion logic.

Best Practice: Implement an abstraction layer in the ingestion pipeline that standardizes input (e.g., Telegraf → Logstash → Unified Schema) to decouple downstream systems from protocol-specific behavior.

6.5.2 Semantic Compatibility & Data Normalization

Protocol support alone does not guarantee interoperability. Semantic mismatches, where identical metrics are labeled, formatted, or timestamped differently, are a leading

cause of faulty alerts, data duplication, and operator confusion. This subsection illustrates how to bring coherence across a fragmented telemetry landscape.

Mismatch Type	Example	Mitigation Strategy
Unit inconsistency	Power reported in W vs. kW	Normalize during ingestion using a predefined transformation dictionary
Inconsistent naming	pdu_socket_01_kwh vs Outlet03_Consumption	Enforce naming schema
Timestamp granularity	Millisecond precision in one device, 30s polling in another	Align to a reference clock. assign collection timestamp if native timestamp is missing
Firmware metric drift	New firmware exposes different OID trees or changes field semantics	Maintain a compatibility registry by device model + firmware in CMDB

6.5.3 CMDB Integration for Source Consistency

A mature CMDB is not just a passive inventory - it becomes an active compatibility broker.

CMDB Responsibilities:

- Store metadata such as:
  - Supported telemetry protocols
  - Device model and firmware version
  - Ownership, SLA level, and business-criticality
- Track configuration drift: actual vs. expected telemetry behaviors
- Define which devices can and should be polled for specific metrics

Example: Before enabling phase load monitoring, the platform checks in CMDB whether the device:

- Supports SNMP v3
- Has phase metrics enabled in the current firmware
- Belongs to a site with advanced analytics enabled

This ensures intentional monitoring, not blind data scraping.

Real-World example: In one software-operated site, CMDB was extended to include firmware compatibility metadata. This enabled ingestion pipelines to auto-select polling

templates based on device model, reducing false alerts by 38% and speeding up onboarding of new devices by 62%.

6.5.4 Interoperability with Enterprise Systems

Observability platforms are not islands. They must interoperate with enterprise systems for incident management, compliance, and analytics.

Example Integration points:

System	Function	Integration Method
Alerting	Incident & change management	REST API, CMDB sync, ticketing
GRC Platforms	Governance, risk, compliance	Reporting pipeline or export
Observability layer and correlation engine	Data lakes, trend analysis	Logstash → Elasticsearch forwarding
CI/CD	Observability-as-code	GitHub Actions, Helm, Terraform
SIEM	Security incident detection	Syslog forwarding, log tagging
Data Visualization (e.g., Kibana)	Dashboards for stakeholders	Elasticsearch sync, metadata tagging

Regulatory Relevance: EU reporting regulations (e.g., EED, CSRD) increasingly mandate **interoperability** - e.g., the ability to export metrics in standardized formats (JSON, CSV, XML) to government databases or auditors.

6.5.5 Legacy and Vendor-Locked Devices

In real-world environments, total compatibility is rare. A platform must offer graceful degradation:

- **For non-instrumented racks:** Use metadata tagging only
- **For legacy SNMPv2c:** Poll via gateway collectors with throttled frequency
- **For proprietary systems:** Request vendor integration plugins, or treat as passive data sources

Strategic Guidance: Maintain a "**telemetry tier**" classification:

Tier 1: Fully instrumented with real-time polling and event support

Tier 2: Partially instrumented or legacy-polled with relaxed alerting

Tier 3: Inventory-only, requiring manual data entry or offline updates

Summary of Compatibility Challenges & Mitigations

Challenge	Mitigation
Protocol mismatch	Collector abstraction layer, protocol tagging
Semantic drift	Normalization schema, CMDB enforcement
Firmware inconsistency	Compatibility matrix, ingestion validation
Legacy devices	Tiered telemetry model, passive tagging
Regulatory exports	Format standardization, audit logging

As industry moves toward unified observability fabric standards (e.g., OpenConfig, DMTF Redfish extensions), platform architecture should anticipate convergence. Investing in modular, schema-aware ingestion and CMDB-driven normalization will not only ease today's interoperability issues but also position the platform for adaptability as standards evolve.

6.6 Practical Adoption Tips by Stakeholder Type

The success of an observability platform is not solely contingent upon its technical sophistication. Equally critical is the diverse roles that engage with, govern, and depend on the platform outputs. From data center operators to executive leadership, stakeholders have distinct concerns, expectations, and success criteria.

This section provides a structured overview of how different stakeholder groups can interpret, adopt, and benefit from observability - highlighting practical considerations, decision factors, and real-world enablers. Rather than presenting generic best practices, we translate stakeholder needs into implementation touchpoints that enhance relevance, foster alignment, and ensure long-term adoption.

6.6.1 Data Center Operations Teams

For facility engineers and on-site operations staff, observability must move beyond passive monitoring and become a tool for proactive management.

Their primary interests often include:

- Fault detection and resolution (e.g., thermal hotspots, circuit imbalances)
- Capacity and load forecasting

- Compliance with operational SLAs (e.g., temperature, uptime, power safety thresholds)

Operational Consideration: Operators require telemetry that is both granular and actionable. Socket-level monitoring is valuable only when paired with intuitive dashboards, actionable alert thresholds, and clear escalation paths.

Adoption Enabler: Co-locate observability dashboards, integrate alert flows into NOC/RunOps workflows, and avoid data overload by filtering to role-specific signals (e.g., only alerts triggered in the operator site zone).

Example: In one regional hub, timely energy telemetry helped an operator detect that redundant power strips were being underutilized. Through minor rack layout changes, they achieved 8% better load balancing, reducing thermal hotspots and risk of breaker trips.

6.6.2 Platform and Infrastructure Architects: Designing for Modularity and Scalability

Infrastructure architects must ensure that observability stacks are modular, maintainable, and adaptable across heterogeneous environments.

Their focus is strategic and structural:

- Integration patterns (e.g., GitOps, CI/CD observability-as-code)
- Modular component selection (e.g., choosing Telegraf over custom scripts)
- System-level consistency across sites and platforms

Architectural Concern: Observability solutions that are hardcoded to local configurations or lack clear modularity quickly become technical debt. Architects must define common schemas, telemetry pipelines, and governance layers to avoid fragmentation.

Adoption Enabler: Provide reference implementations, validate platform components via architecture review, and ensure observability aligns with existing DevOps and platform operations frameworks.

Example: In a distributed setup spanning co-location and edge data centers, the architect enforced a shared metadata taxonomy (via CMDB) to ensure that all telemetry adhered to a consistent format. This unlocked cross-site analytics and reduced integration issues across observability layers.

6.6.3 Compliance and Sustainability Officers: Verifiability, Traceability, and Audit-Readiness

For compliance stakeholders, observability is not just a technical capability - it is a regulatory instrument. Their primary interest is proving that operations adhere to legal

and environmental standards, such as the European Energy Efficiency Directive (EED), Corporate Sustainability Reporting Directive (CSRD), or local energy codes.

Compliance Challenge: Raw telemetry alone is insufficient for audits. Data must be time-stamped, normalized, and traceable back to physical infrastructure components and operating conditions.

Adoption Enabler: Integrate observability pipelines with GRC systems and data export workflows. Use version-controlled dashboards for regulatory KPIs (e.g., PUE, renewable energy share). Establish clear data retention policies and immutable logs aligned with audit timelines.

Example: At a sustainability-driven site in the EU, automated telemetry from PDUs and EMUs was combined with facility sensor data to generate monthly Scope 2 emissions reports, directly fulfilling CSRD requirements for energy disclosure.

6.6.4 Procurement and Asset Management: Lifecycle Intelligence and Capacity Planning

Procurement teams benefit from observability through better lifecycle planning and cost efficiency. Knowing which racks are overprovisioned, underutilized, or aging helps inform purchasing decisions and contract renegotiations with co-location providers.

Procurement Insight: Telemetry can be used to correlate usage patterns with asset lifespans, identify redundant equipment, and optimize refresh cycles.

Adoption Enabler: Provide procurement with tailored dashboards (e.g., average power draw vs. rated capacity per vendor) and link telemetry data with asset inventory systems to enable financial forecasting and SLA optimization.

Example: By analyzing load distribution across PDUs, one enterprise discovered consistent underutilization of a high-capacity rack zone. This insight enabled them to downscale their lease.

6.6.5 Executive Leadership: Strategic Alignment and ESG Impact

Executives are not typically interested in raw telemetry - but they are deeply invested in the business impact of observability.

This includes:

- Risk mitigation through early failure detection
- ESG reporting credibility
- Strategic facility expansion decisions

Leadership Focus: Observability must communicate high-level narratives: “Are we compliant?”, “Are we efficient?”, “Are we at risk?”

Adoption Enabler: Develop executive-level dashboards with quarterly trends, predictive forecasts (e.g., PUE projections), and regulatory alignment indicators. Link observability metrics to ROI (e.g., savings from deferred expansion, energy efficiency gains).

Example: In a quarterly board meeting, telemetry insights supported a decision to defer data center expansion by showcasing how improved rack consolidation extended capacity greatly.

6.7 Lifecycle Sustainability and Long-Term Maintainability

Establishing an observability platform is not merely a technical milestone - it is the initiation of a governance lifecycle. Once deployed, the platform must remain adaptable, interpretable, and actionable in the face of organizational change, infrastructure evolution, regulatory updates, and shifting stakeholder priorities.

This section outlines the major categories of sustainability risks and provides a forward-looking framework to mitigate long-term degradation in observability platform relevance, performance, and trustworthiness.

6.7.1 Understanding Sustainability in Observability Context

Sustainability in observability refers not only to environmental considerations, such as minimizing storage and computing waste, but also to the organizational, procedural, and epistemic longevity of the platform.

Key questions arise:

- Will the system still make sense when key personnel depart?
- Will telemetry continue to align with business needs as infrastructure evolves?
- Will dashboards and alerting policies remain valid under new regulatory regimes?

These questions frame observability as a dynamic system of knowledge, not just a static monitoring toolkit. Like any socio-technical system, observability requires active stewardship to remain resilient, interpretable, and valuable.

6.7.2 Key Categories of Sustainability Risk

Sustainability risks fall into six major domains, each with cascading operational implications:

Risk Domain	Description	Typical Manifestation
Telemetry Drift	Device updates, firmware changes, or topology changes silently alter telemetry	Values shift or disappear, dashboards become outdated

Semantic Decay	Misalignment between data meaning and stakeholder interpretation	Dashboards are misread; alerts are misunderstood
Governance Gaps	No clear ownership of thresholds, policies, or platform evolution	Alert noise increases, platform becomes siloed
Tooling Fragmentation	Parallel tools emerge with uncoordinated scopes or duplication	Shadow monitoring solutions develop, wasted resources
Regulatory Misalignment	Legal frameworks evolve (e.g., CSRD, EED) but telemetry stays static	Non-compliance, reporting failures
User Disengagement	Stakeholders stop trusting or using observability outputs	Underused systems, reduced ROI, false sense of security

6.7.3 A Sustainability Framework for Observability

To proactively manage these risks, we propose a Sustainability Framework that integrates platform engineering with organizational learning and regulatory foresight.

It consists of five interlinked principles:

1. **Temporal Validity of Telemetry**

Challenge: Device output change over time due to firmware upgrades or hardware replacements.

Mitigation: Implement a telemetry validation pipeline that cross-checks incoming data against CMDB-logged expectations (e.g., OIDs, unit formats, update frequency).

2. **Intent Preservation via Metadata**

Challenge: Dashboards and alerts lose context if creators leave, or documentation is lost.

Mitigation: Mandate metadata tagging for all observability assets (dashboards, alerts, transforms), capturing "why" alongside "what." For example, a tag like {"compliance-metric": "EED-AnnexVII-1"} ensures intent persists.

3. **Embedded Governance Anchors**

Challenge: Without ownership models, observability becomes fragmented.

Mitigation: Assign platform Product Owner (PO) role and domain-specific telemetry stewards. Require quarterly reviews of alert thresholds, metric utility, and system integration fidelity.

4. **Regulatory Horizon Scanning**

Challenge: Regulations change faster than platform updates.

Mitigation: Link GRC (Governance, Risk, Compliance) workflows with observability design. Ensure dashboards are annotated with regulation-specific references (e.g., "CSRD-Scope3-CO2") to ease auditability.

5. **Participatory Review Mechanisms**

Challenge: Platforms drift if users are passive.

Mitigation: Introduce structured feedback loops (e.g., observability retrospectives, design review boards). Create usage analytics to detect declining interaction and launch targeted re-engagement.

6.7.4 Example: Sustainability Breakdown in a Multi-Site Data Center

In one enterprise deployment, a leading European data center operator experienced a gradual erosion of observability value over 18 months. Initially laid for its socket-level energy insights, the platform suffered from the following:

- Firmware upgrades invalidated 20% of SNMP-based metrics.
- No one noticed missing data for 7 weeks due to absent meta-alerting.
- New compliance rules (EU 2023/1791) required reporting formats unsupported by the system.
- Stakeholders began exporting raw data to Excel for custom reporting, leading to shadow systems.

A sustainability audit revealed that while telemetry ingestion was technically functional, governance, documentation, and intent alignment had eroded. A recovery plan was introduced involving metadata backfilling, CMDB validation automation, quarterly governance councils, and integration of legal compliance roles into the observability team.

6.7.5 Strategic Recommendations

To institutionalize sustainability, organizations should treat observability not as a finished product but as a living capability.

Recommended practices include:

Adopt Lifecycle Audits: Every 6-12 months, assess telemetry integrity, alert fidelity, stakeholder engagement, and regulatory alignment.

Define an Observability Constitution: A lightweight policy artifact describing principles, thresholds for action, naming conventions, and governance roles.

Bake Sustainability into Procurement: Require vendors to document telemetry support per firmware, including change policies and backward compatibility.

Formalize Change Notifications: Connect firmware updates or site topology changes to alert platform stewards via CMDB triggers or ITSM workflows.

Long-term observability success depends less on initial tooling selection and more on how well an organization governs, interprets, and adapts its telemetry ecosystem over time. Without proactive lifecycle management, even the most advanced observability stack will drift into obscurity, becoming a shelfware platform of latent potential. By embedding governance, metadata clarity, and regulatory responsiveness into the platform's DNA, organizations ensure that observability becomes not just sustainable, but indispensable.

6.8 Common Pitfalls and How to Avoid Them

While the implementation of a data center observability platform promises enhanced transparency, resilience, and regulatory alignment, real-world deployments are frequently derailed by predictable, yet preventable failures. These missteps span across technical, organizational, procedural, and cultural domains. Understanding these pitfalls not only strengthens project outcomes but also builds institutional resilience against future disruptions.

This section categorizes some of the most common implementation errors, analyzes their systemic roots, and provides structured mitigation approaches. It serves both as a diagnostic tool for ongoing projects and as a proactive planning guide for new observability initiatives.

6.8.1 Pitfall №1: Technology-First Thinking

Symptom: Stakeholders rush to deploy tools or protocols (e.g., SNMP polling or Redfish APIs) without clearly understanding business drivers, end-user needs, or compliance obligations.

Underlying Cause: A bias toward solutionism, if technology alone will deliver value, irrespective of context.

Impact: Misaligned data streams, unused dashboards, alert fatigue, and poor stakeholder adoption.

Mitigation Strategy:

- Conduct structured **Contextual Definition** workshops before choosing tools.

- Translate platform goals into observable business decisions (e.g., “Are we exceeding PUE targets?”), not just technical metrics.
- Involve compliance and operational stakeholders during the design phase, not just after deployment.

Academic Insight: Technology without governance becomes entropy. In complex systems theory, emergent value arises not from individual nodes (tools), but from their structured interrelation, an idea as per the ideas of sociotechnical systems engineering.

6.8.2 Pitfall №2: Over-Engineering in Immature Environments

Symptom: Teams attempt to deploy fully integrated, real-time observability stacks in legacy or telemetry-poor environments.

Underlying Cause: Aspirational design that ignores current-state limitations - e.g., SNMPv2-only devices, no CMDB integration, fragmented ownership.

Impact: Technical debt accrues rapidly. Stakeholders lose trust as platform outputs do not reflect operational realities.

Mitigation Strategy:

- Use a maturity-based rollout model (see §6.3.1 Crawl–Walk–Run–Fly).
- Build from minimum viable telemetry (MVT) upward: start with basic PDU polling and alert normalization.
- Formalize a site capability map to match ambitions with reality.

Academic Insight: System maturity must be scaffolded. Drawing from capability maturity models (e.g., CMMI), premature optimization is not only wasteful, but also structurally unsustainable.

6.8.3 Pitfall №3: Ignoring Semantic Normalization

Symptom: Metrics are collected in incompatible formats, units, or naming conventions across vendors and sites.

Underlying Cause: Absence of data modeling, lack of shared vocabulary, and no enforced transformation logic.

Impact: Visualization errors, alerting mismatches, broken compliance reports.

Mitigation Strategy:

- Establish a **semantic schema** for telemetry fields (e.g., voltage_phase_avg, power_draw_kw).
- Normalize during ingestion (e.g., Telegraf → Logstash) using a transformation dictionary.
- Leverage CMDB as a schema validation engine.

Real-World Example: One organization failed to detect a power imbalance due to inconsistent OID mappings across firmware versions. Once normalization logic was introduced, the alert logic stabilized, and false positives dropped by 85%.

6.8.4 Pitfall №4: Siloed Ownership and Governance Drift

Symptom: No one "owns" the observability platform for post-deployment. Thresholds are outdated, dashboards unmaintained, and alerts misrouted.

Underlying Cause: Observability is viewed as a project, not a living system requiring ongoing governance.

Impact: Degradation of value, alert noise, compliance gaps, executive disengagement.

Mitigation Strategy:

- Establish long-term governance anchors (see §6.7.3), such as:
- A Platform Product Owner (PPO)
- Domain-specific telemetry stewards
- Quarterly governance review cycles
- Use a version-controlled observability repository (e.g., via Git) to track schema, alert, and dashboard changes.

Academic Insight: From a systems governance perspective, unmaintained observability is analogous to unpatched infrastructure: vulnerable to drift, decay, and organizational amnesia.

6.8.5 Pitfall №5: Treating Observability as a Technical Island

Symptom: Observability is implemented in isolation, unintegrated with CMDB, ITSM platforms, GRC systems, or ESG reporting workflows.

Underlying Cause: Lack of stakeholder alignment and architecture foresight.

Impact: Duplicated effort, fragmented insights, inability to prove compliance or trigger operational actions.

Mitigation Strategy:

- Design for interoperability from the outset (see §6.5.4):
- Link to ticketing platform for incident workflows
- Forward metrics to ELK data Lake for compliance visualization
- Integrate with GRC tools for audit readiness

Document use-case-based system interdependencies (e.g., "power drift → alert → SNOW ticket → mitigation → CMDB update").

Academic Insight: The value of observability is not in seeing but in acting. From a cybernetic perspective, observability without actuation is only half a system.

6.8.6 Pitfall №6: Underestimating Change Management and Training

Symptom: Platform is technically functional but remains underutilized. Stakeholders bypass it using spreadsheets or custom scripts.

Underlying Cause: Poor onboarding, lack of clarity, or fear of change.

Impact: Shadow systems, fragmented data, platform abandonment.

Mitigation Strategy:

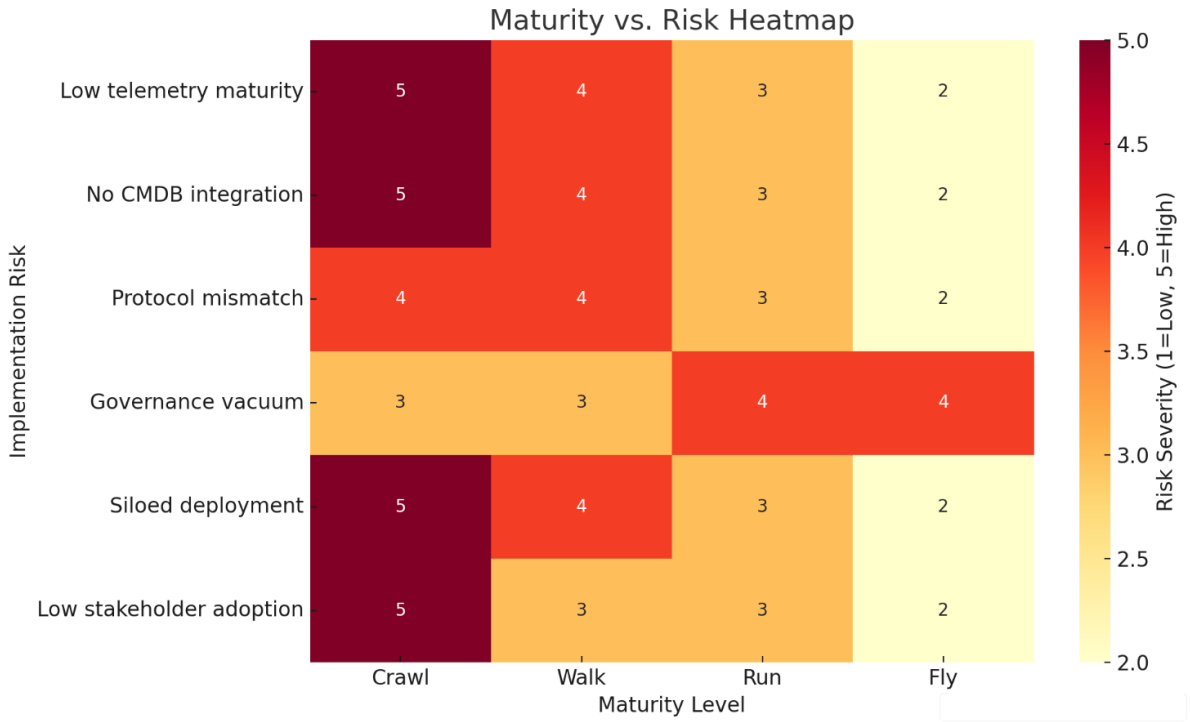
- **Provide persona-based onboarding:** different roles receive dashboards, alerts, and training tailored to their decisions and processes.
- **Launch continuous engagement programs:** Office hours, internal champions, quarterly design jams.

Real-World Observation: In a hybrid cloud operator, dashboard usage increased 4x after introducing role-specific landing pages and targeted walkthroughs using internal micro-learnings.

6.8.7 Pitfall Awareness as a Strategic Advantage

Avoiding failure is not simply about foresight - it is about institutional learning. The observability platform must evolve from being seen as a static deployment toward a resilient, governable, and socially adopted capability. The pitfalls outlined here are not isolated mistakes, they are systemic patterns. By recognizing them early and embedding countermeasures into design, implementation, and operational practice, organizations enhance not only their observability readiness but their overall digital resilience.

Figure: Maturity vs Risk Heatmap



6.9 Long-Term Sustainability & Continuous Improvement

Deploying a data center observability platform is the foundation of a sustainable capability that must evolve alongside the organization's infrastructure, regulatory obligations, and strategic goals. This section explores how to design for long-term relevance, avoid obsolescence, and institutionalize observability as a dynamic, value-generating asset rather than a static technical deployment.

6.9.1 The Lifecycle of Observability Platforms

Sustainability in observability refers to the platform's capacity to remain accurate, relevant, and actionable over time. Like infrastructure itself, observability systems undergo phases:

- **Initialization:** Configuration of the minimal viable telemetry and data ingestion infrastructure.
- **Operationalization:** Integration with workflows, dashboards, alerting, and compliance reporting.
- **Optimization:** Refinement of telemetry granularity, data tagging, and automation.
- **Evolution:** Adaptation to new protocols, new equipment, and evolving stakeholder needs.

To sustain these phases, organizations must shift from "deployment mindset" to observability governance.

This includes:

- Policy-based configuration management (e.g., via GitOps),
- Governed change control over thresholds, alerting logic, and naming conventions.
- Continuous validation of telemetry quality, coverage, and latency.

Example: A telemetry field used in annual ESG reports should be version-controlled, auditable, and tested for semantic stability across firmware updates. Failure to do so risks non-compliance.

6.9.2 Governance and Stewardship Models

Sustainability is not possible without clear organizational ownership and distributed stewardship. Observability systems that lack governance often suffer from drift, duplication, or decay. Successful models embed stewardship at three levels:

Governance Level	Responsibility	Example Practice
Strategic	Sponsor budget and business value alignment	ESG board defines reporting KPIs
Tactical	Enforce architectural standards and evolution	Architecture Review Board approves ingestion schemas
Operational	Maintain dashboards, alerts, CMDB mappings	NOC engineers refine alert thresholds quarterly

A common pitfall is the “abandoned platform” syndrome - where telemetry is collected but never acted upon because ownership of alerts or metrics has not been assigned.

Best Practice: Implement a Telemetry Ownership Matrix, mapping each signal or KPI to a business function and escalation path.

6.9.3 Feedback Loops and Telemetry Refinement

A sustainable platform incorporates mechanisms for continuous improvement through structured feedback. Observability is not a “set-and-forget” tool it must learn from experience.

Effective feedback loops include:

Post-incident analysis: Were alerts timely and actionable?

Data quality audits: Are there silent failures in collection pipelines?

Stakeholder reviews: Are dashboards still aligned with stakeholder needs?

Capacity planning exercises: Is telemetry guiding infrastructure growth or underutilization?

Practical Tip: Use quarterly "observability reviews" as part of operational governance boards, like budget or performance reviews to recalibrate thresholds, dashboards, or coverage areas.

6.9.4 Obsolescence and Futureproofing

Long-term sustainability requires systems to be designed with graceful evolution in mind.

Key strategies include:

Abstracted ingestion layers: Decouple data collection from analysis and storage to support protocol evolution (e.g., shift from SNMP to Redfish).

Modular architecture: Ensure new telemetry sources or visualization layers can be added without refactoring core logic.

Semantic versioning: Apply API-like discipline to metric formats, naming conventions, and dashboard schema.

6.9.5 Metrics for Platform Sustainability

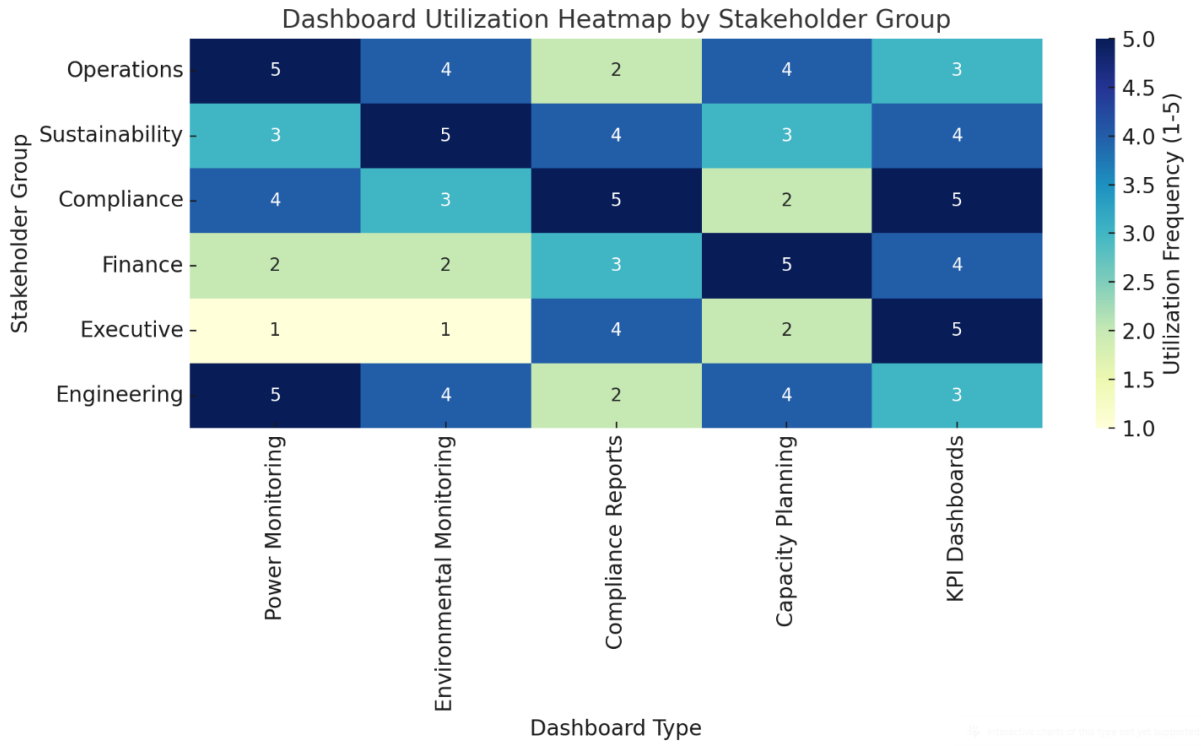
Just as observability enables organizations to monitor their environments, the observability platform itself should be monitored and evaluated.

Suggested sustainability KPIs include:

Metric	Description
Telemetry Coverage Ratio	Percentage of racks, PDUs, and sensors actively reporting
Alert Fatigue Index	Ratio of alerts resolved vs. ignored or auto closed
Compliance Coverage Score	Degree of alignment with required regulatory metrics (EED, CSRD, etc.)
Data Pipeline Health	Average ingestion latency, packet drop rate, schema adherence
Dashboard Utilization	Frequency of access and updates by defined stakeholder groups

These metrics should themselves be visualized as part of a "platform health dashboard" that is reviewed quarterly.

I.e. – Fig :



This heatmap is visualizing the utilization frequency of various dashboard types by different stakeholder groups. The values (from 1 to 5) indicate how often each group engages with specific observability dashboards:

- Operations and Engineering show the highest usage of power monitoring tools.
- Sustainability and Compliance teams rely heavily on environmental monitoring and compliance reports.
- Finance and Executives favor KPI dashboards and strategic insights.

Appendix A Use Cases

This appendix consolidates use cases and technical implementation requirements for key areas of the observability infrastructure.

A.1 Configuration management for monitored devices

General guidelines:

- Ensure appropriate version & change control processes are implemented (as outlined in Section 4.7)
- Store configuration in common CMDB system
    - Keep CMDB entries up to date, ensuring it reflects appropriate physical placement and connections
    - Keep track of device level monitoring readiness e.g. in case of component replacement or introducing new hardware to the system report if usual monitoring configuration could not be applied (e.g. missing component / device model mismatch)
- Use standardized configuration templates for all managed device types
- Ensure configuration drift control measures are in place

Configuration key items:

Device identification information

- System name (e.g. network hostname) - well known device identifiers used in CMDB system
- Additional identification properties e.g. physical location information, ownership, device type
    - Applies also to the device modules / attached accessories e.g. environmental sensors
- Network connection settings including supported IP protocols / Addressing / DNS server usage
- Time & date configuration e.g. Network Time Protocol configuration
- Appropriate operating system patches or firmware version control

Security

- Allowed access protocols (for both configuration and monitoring), depending on selected methods, e.g.
    - Minimum security specification for HTTPS access methods for REST APIs / Redfish endpoints
    - Appropriate SNMP protocol settings like versions, communities, encryption methods
- Authentication
    - Local user accounts configuration / integration with centralized systems

    - MFA configuration
- Authorization
    - Role based access control configuration – separating “administrator” and “monitoring” role
- Auditing
    - Appling minimum security specification e.g. integration with SIEM tools whenever possible
- Additional security hardening best practices
    - Ensuring not used services/features/access methods are disabled e.g. TFTP or Telnet servers

Monitoring

- Configure appropriate access methods to monitoring/telemetry data
- Configure desired monitoring features are enabled in device settings e.g. environmental sensors data collection or outlet level data collection is turned on
- Configure appropriate alerting thresholds
- Configure alerting severity
- Ensure monitoring agent/collector settings are in sync with monitored device side configuration e.g. appropriate addresses of SNMP Trap receivers / Redfish event subscription addresses are used in device-level configuration

A.2 Power & environmental metrics visualization

General guidelines:

- Whenever possible use version-controlled visualization engine and follow standard change control procedures
- When creating breakdowns or filters use appropriate set of tags to ensure right metric representation, keeping in mind relationships between objects (e.g., breaker metrics for single PDU in given rack would be identified at least by tags: host, hw.parent and hw.name/hw.id)
    - This applies also for visualization of event data like error logs or SNMP trap data
- Apply appropriate color-coding to indicate primary / secondary (backup) power source (as indicated by appropriate resource tagging and/or taken from device naming convention)
- When visualizing multiple metrics on single dashboard ensure color coding is consistent for common set of tags
- Indicated threshold/classes/boundaries on graphs should use well-known values, desirably taken from monitoring (e.g., device power limits) or CMDB system (location specific limits, forecasted values)

Observability infrastructure dashboard

Provides an overview of monitoring infrastructure, providing information with agent/collectors statuses, number of collected metrics, and error rates.

- Monitoring infrastructure errors view e.g., containing sync issues with CMDB system, unhealthy instances
- Metric collection errors time chart, grouped by monitoring agent/collector instance
- Collection error rate per monitored device
- If applicable – metrics analysis/persistence layer errors per instance of agent/collector

Rack dashboard

Provides detailed view of a rack, providing information useful for day-to-day operations, including the most detailed data from PDU units including phase power, outlets, and breakers.

- Current device state, “at-glance” overview for active alerts, power devices & their component statuses
- PDU (true) power (kilowatts) & apparent power in time chart view if possible visual comparison with device rating limits and forecasted values.
- PDU hourly energy consumption in time chart view if possible visual comparisons with forecasted values.
- Power Phase power (kilowatts) & apparent power (kilovolt-ampere) in time chart view
  PDU Breakers data breakdown (table) - side-by-side comparison with primary & secondary (backup) feed including peak current, current rating
- PDU Outlets data breakdown (table) - side-by-side comparison with primary & secondary (backup) feed including: braker relationship, peak true & apparent power, peak current, outlet current rating
- Environmental sensor data:
  Temperature & humidity time charts, breakdown over sensor location, with ASHRAE recommendations indication (e.g. appropriate area coloring).

Datacenter room dashboard

Provide detailed view for datacenter rooms, to provide summarized PDU & rack data for given physical location (usually sharing cooling and power sources).

- Power data summary for PDUs as time chart – visual comparison with forecasted values
- Peak environmental sensor values, breakdown over rack & sensor location – with room averages (“at-glance” overview for peak values for selected period)

Datacenter room heatmaps

Provide environmental sensor data visualization with related infrastructure (e.g., racks, cages. hot/cold containment zones) including spatial information in 2D/3D view (depending on availability of the data and integration with CMDB systems).
Gradient colors should reflect local SLA classes, e.g., using ASHRAE data center thermal guidelines, starting with yellow when exceeding “recommended thresholds, turning red when approaching “allowable” boundaries.

- Temperature gradient breakdowns depending on sensor location (front/rear and/or bottom/middle/top)
- Humidity gradient breakdowns depending on sensor location (front/rear and/or bottom/middle/top)

Appendix B: Software Bill of Materials (SBOM)

This appendix provides a comprehensive Software Bill of Materials (SBOM) for the open-source components utilized within the Data Center Observability Platform (including Aperio components), as defined by the blueprint. It ensures transparency, traceability, and compliance with the sustainability and interoperability objectives outlined in this blueprint.

Only open-source or open-standard components have been included to support principles of modularity, openness, and composability, while minimizing vendor lock-in.

B.1 Methodology

- Implementing Apeiro Reference Architecture
- Utilizing components provided by Aperio management plane (Greenhouse)
- Identification of all foundational and integration platform elements.
- Validation of open-source licensing for each component.
- Mapping functional roles to logical architectural layers.
- Alignment with regulatory, operational, and sustainability drivers.

B.2 Component Overview

Component	Function within Architecture	Open-Source License	Notes
Kubernetes - Gardener	Orchestration of containerized observability services	Apache License 2.0	Gardener extends to native Kubernetes. Apeiro cloud-edge layer component
Greenhouse	Orchestration of observability components in distributed environment	Apache License 2.0	Orchestration of distributed Kubernetes infrastructure. Apeiro management plane component
OpenTelemetry Collector	Aggregation and export of telemetry data (metrics, logs, traces)	Apache License 2.0	Vendor-agnostic telemetry instrumentation. Main Apeiro Observability component (managed by Greenhouse)
Telegraf	SNMP polling agent for environmental	MIT License	Supports direct ingestion from PDUs and sensors.

	and power telemetry		
NetBox	CMDB and device metadata management	Apache License 2.0	Key for asset-to-telemetry mapping.
Perses	Visualization / dashboarding	Apache License 2.0	Metrics visualization platform (managed by Greenhouse)
Prometheus & Thanos	Metrics collection, storage and alerting	Apache License 2.0	Thanos extending Prometheus functionality with long term metrics storage (managed by Greenhouse)
OpenSearch	Storage, search, analysis and visualization	Apache License 2.0	Event storage, data visualization and analysis (managed by Greenhouse)
SNMP Libraries (e.g., PySNMP, GoSNMP)	Communication with PDUs, EMUs, sensors	BSD/MIT Licenses	Enables polling and trap handling for device telemetry.
Redfish API Clients (optional integration)	API access for environmental and power telemetry from hardware	DMTF Open Specification	Optional based on hardware support.

B.3 License Summary

License Type	**Components Governed**	**Compliance Considerations**
Apache License 2.0	Gardener, Greenhouse, Kubernetes, OpenTelemetry, Netbox, Perses, Prometheus, OpenSearch, Thanos	Permissive; modification, distribution allowed with attribution.
MIT License	Telegraf, SNMP libraries	Very permissive; minimal obligations.

Elastic License (OSS Version)	Elasticsearch OSS, Kibana OSS	OSS versions must be used to avoid commercial licensing restrictions.
AGPLv3	Grafana	Strong copyleft license; acceptable under open-source goals.
OpenAPI / REST Standard	ServiceNow connectors (custom)	Standard-based; open implementations available.
DMTF Standard	Redfish API	No proprietary restrictions.

B.4 Architectural Mapping

Logical Architecture Layer	**Associated Components**
Orchestration and Deployment	Gardener, Kubernetes, Greenhouse
Telemetry and Ingestion	Telegraf, OpenTelemetry Collector, SNMP Libraries
Storage	Prometheus, Thanos, OpenSearch
Visualization	Perses, OpenSearch
Asset Management and CMDB	NetBox
Alerting and Incident Detection	Prometheus - AlertManager
Ticketing Integration	REST API (open-source connectors)
Optional Hardware API Layer	Redfish API Clients

B.5 Sustainability and Interoperability Alignment

Consistent with sustainability objectives (Chapter 6.7):

- Open governance and extensibility across all components.
- Avoidance of vendor-locked ecosystems.
- Lightweight, scalable telemetry pipelines reduce operational overhead.
- Compatibility with energy efficiency and regulatory reporting frameworks.

B.6 Versioning and Change Management

- Platform components are maintained via Greenhouse plugin structure
- Upgrades must preserve open-source compliance and be tested against regression suites.
- Change requests related to core components must follow governance processes (outlined in Section 4.7)

Appendix C - References and Source Materials

This appendix enumerates the principal references, standards, regulatory frameworks, and project documentation sources that informed the extension of Aperio Reference Architecture in the form of Data Center Observability Platform Reference Architecture. It serves to acknowledge foundational materials and provide transparency regarding the basis for design decisions, regulatory mappings, and operational models discussed throughout the blueprint.

C.1 Regulatory and Policy References

Title	Source / Publisher	Relevance
Directive (EU) 2023/1791 (Energy Efficiency Directive)	European Union	Legal driver for mandatory energy and efficiency reporting by data centers exceeding 500kW installed IT power.
Delegated Regulation (EU) 2024/1364 (Annex I KPIs)	European Union	Specifies detailed reporting requirements for sustainability KPIs in data center operations.
Regulation (EU) 2019/424 (Ecodesign Regulation for servers and storage)	European Union	Establishes minimum efficiency standards for data center equipment procurement and operation.
Climate Neutral Data Centre Pact	Industry Self-Regulation	Voluntary commitments towards achieving carbon neutrality and sustainability targets by 2030.
EU Code of Conduct for Data Centre Energy Efficiency	European Commission	Best practices framework for energy-efficient data center design and operation.

C.2 Open Standards and Technology Specifications

Title	Source / Publisher	Relevance
Apeiro Reference Architecture	SAP SE / NeoNephos Foundation	Blueprint for Cloud-Edge Continuum
Gardener Project Documentation	NeoNephos Foundation	Management and orchestration of Kubernetes clusters at scale.

Greenhouse Project Documentation	NeoNephos Foundation	Cloud operation platform for distributed Kubernetes infrastructure.
Kubernetes Documentation	Cloud Native Computing Foundation (CNCF)	Foundation for container orchestration within the observability platform.
OpenTelemetry Specification	OpenTelemetry Project (CNCF)	Telemetry collection framework for metrics, logs, and traces.
NetBox Documentation	NetBox Community	CMDB and infrastructure metadata management.
Prometheus documentation	Cloud Native Computing Foundation (CNCF)	Metrics collection and storage, alerting
OpenSearch documentation	Linux foundation	Data search, analysis and visualization
Thanos documentation	Cloud Native Computing Foundation (CNCF)	Prometheus long term metrics storage
Perses documentation	Cloud Native Computing Foundation (CNCF)	Visualization dashboards
Telegraf Documentation	InfluxData	SNMP telemetry ingestion agent for environmental and power monitoring.
SNMP Protocol Specifications (RFC 1157, RFC 3411)	IETF (Internet Engineering Task Force)	Protocols enabling telemetry polling and traps from PDUs and environmental sensors.
Redfish API Standard	DMTF (Distributed Management Task Force)	Open standard for secure and scalable hardware telemetry acquisition.
Open Compute Project (OCP) Documentation	Open Compute Project Foundation	Sustainability, modularity, and transparency principles for hardware and infrastructure design.

C.3 Methodological References and Community Practices

Title	Source / Publisher	Relevance

Cloud reference architecture	Neo Nephos Foundation	General guidelines for building cloud infrastructure
Best Practices for Observability Architectures	CNCF Observability Working Group	Conceptual alignment for defining observability layers, telemetry processing, and stakeholder needs.
Infrastructure as Code and CI/CD Practices	GitHub Actions Documentation	Automation framework for infrastructure deployment, onboarding, and configuration drift monitoring.
OpenTelemetry API Reference	OpenTelemetry Project	Provides detailed API documentation for implementing observability in applications.
Gardener Architecture Documentation	Gardener Project	Offers insight into the architecture and operation of Gardener for Kubernetes cluster management.
NetBox Labs Documentation	NetBox Labs	Detailed documentation on NetBox features, deployment, and integration capabilities.
Telegraf Configuration Guide	InfluxData	Instructions for configuring Telegraf for various data collection scenarios.
Redfish Specification DSP0266	DMTF	Defines the Redfish standard for hardware management using RESTful APIs.
Open Compute Project Specifications	Open Compute Project Foundation	Provides hardware specifications and guidelines promoting open and efficient data center designs.
Data Center Standards and Guides	ASHRAE	Set of guidelines related to thermal conditions withing data centers