DAVi: A Slim, Secure and Scalable Framework for Developing Data Analytics and Visualization Platforms

Manish Agrawal*
ma331919@gmail.com
IIT Kanpur
Kanpur, Uttar Pradesh
India

Prashik Ganer*
prashik.ganer@gmail.com
IIT Kanpur
Kanpur, Uttar Pradesh
India

Amit Bhasita
amitnb24@iitk.ac.in
IIT Kanpur
Kanpur, Uttar Pradesh
India

Khushwant Kaswan
khushwantk24@iitk.ac.in
IIT Kanpur
Kanpur, Uttar Pradesh
India

Tippireddy Yashwanth
yashwantht24@iitk.ac.in
IIT Kanpur
Kanpur, Uttar Pradesh
India

Soumya Dutta
soumyad@cse.iitk.ac.in
IIT Kanpur
Kanpur, Uttar Pradesh
India

Purushottam Kar
purushot@cse.iitk.ac.in
IIT Kanpur
Kanpur, Uttar Pradesh
India

Abstract

Modern scientific, commercial and industrial systems continuously generate large volumes of data in the form of event logs, streaming telemetry, controller data, etc. The task of real-time analysis and visualization of this data is essential to gain actionable insights and troubleshoot any latent issues. However, existing solutions frequently present a choice. Robust business intelligence (BI) tools abound and offer excellent visualization capabilities but limited flexibility for data scientists to implement custom programmatic solutions. On the other hand, data science platforms offer sophisticated coding environments but lack advanced visualization tools. Existing solutions also struggle to offer fine-grained access control and do not allow automated alerts to be delivered to user subgroups should anomalies or failure states be detected in data. To bridge this feature gap, this paper introduces DAVi (Data Analytics and Visualization Interface), a secure, microservice-driven platform that delivers a unified user experience combining advanced visualization capabilities and fully fledged development workflows. A novel feature of DAVi is the in-built real-time notification system that allows data scientists to send automated notifications to an identified subset of users, for example, if an AI model detects impending system failure or deterioration in service quality. DAVi overcomes several technical challenges that may be of independent interest, such as the creation and integration of a honeypot-grade sandbox that must still be permitted access to restricted lines of communication. DAVi offers a single-tenant architecture at the organization level but allows the flexibility of per-user container isolation. Its core innovation lies in the headless integration of a powerful open-source BI engine, Apache Superset [5], with a React-based application. The architecture is sensitive to the demands of high-security environments, including centralized LDAP [27] authentication, fine-grained role-based access control (RBAC), isolated on-demand JupyterLab [19] sandboxes, session management, activity logging, automated snapshots with point-in-time data restoration, among others. DAVi offers a modular architecture with replaceable components and a secure, extensible, and comprehensive solution for modern data analytics.

CCS Concepts

• **Information systems** → **Computing platforms**.

Keywords

Data Analytics, Visualization, Microservice Architecture, Honeypot-grade Sandboxing, Headless System Integration, Fine-grained Access Control, Automated Real-time Notifications, Isolation

ACM Reference Format:
Manish Agrawal, Prashik Ganer, Amit Bhasita, Khushwant Kaswan, Tippireddy Yashwanth, Soumya Dutta, and Purushottam Kar. 2026. DAVi: A Slim, Secure and Scalable Framework for Developing Data Analytics and Visualization Platforms. In *19th Innovations in Software Engineering Conference (ISEC 2026), February 19–21, 2026, Jaipur, India.* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Effective data-driven strategies are essential to the smooth functioning of large, complex systems, especially those deployed at organizational levels. These systems produce high-velocity, continuous streams of data generated by event logs, telemetry channels, controller data, and others. The data streams may include both synchronous and asynchronous channels, may cover thousands of parameters, and may include data with diverse data types, arrival frequencies, and ranges. To ensure operational success, extend asset lifespan, and prevent catastrophic failures, it is essential to have the ability to monitor such information in real-time, detect subtle anomalies, and forecast future states. As manual analysis becomes quickly overwhelmed by both the amount and complexity of this data, there is an urgent need for a scalable, secure, and robust platform to deliver quick, actionable insights.

Design Considerations. This paper reports the design, development and implementation of the DAVi platform. DAVi was created to cater to the needs of an organization of national importance

*Work done while the author was a student at IIT Kanpur

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ISEC '26, Jaipur, India
© 2026 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN XXX-X-XXXX-XXXX-X/XXXX/XX
https://doi.org/XXXXXXX.XXXXXXX

dealing with sensitive information and was tasked with offering a feature-rich, web-based platform that could automate critical monitoring tasks by leveraging data generated by a large in-house data acquisition platform deployed within that organization. More than just a visualization dashboard, DAVi needed to be an integrated platform simultaneously offering the capabilities of data analytics, interactive visualization, and real-time monitoring, all while assuring fine-grained access control and the security of sensitive data. The platform was charged with the task of reducing the burden of manual oversight, allowing data scientists and engineers within the organization to focus on higher-order decision-making and analysis. Key asks were to ensure smooth ingestion of data from an existing infrastructure, be compatible with existing organizational workflows, and integrate automated alert mechanisms and an advanced analytical workflow.

DAVi 's design was underscored by the belief that an effective analytics platform for scientific and engineering domains must unify the traditionally separate paradigms of business intelligence (BI) and data science. Current tools force a trade-off: on one side are user-friendly BI tools such as Tableau and Power BI, which offer powerful visualization and self-service exploration but are fundamentally limited when custom modeling is required (say, writing custom anomaly detection models, data wrangling code, etc., that go beyond the capabilities of SQL queries or drag-and-drop visualizations). On the other side are data science environments, epitomized by notebooks, that provide a high level of flexibility but are often disconnected from production-grade visualization, governance, and access-control workflows.

DAVi's core innovation is to eliminate this dichotomy by integrating both capabilities into a single application and abstracting the underlying technical complexity from the end-user. DAVi allows scientists to move fluidly from viewing a high-level dashboard to launching an isolated, secure coding environment, all within the same interface and using identical data sources. DAVi's architectural design was heavily influenced by stringent client requirements, including absolute data sovereignty, high security, long-term maintainability, and the strategic avoidance of vendor lock-in. These considerations made commercial, cloud-based, off-the-shelf products unsuitable. The selection of an open-source core and the development of a bespoke, containerized microservices-based architecture thus became critical to ensuring full control over the platform's security, deployment environment, and future evolution.

Contributions. This paper presents the design, architecture, and implementation of the DAVi platform. The development of this framework required solving several technical challenges, such as the creation of secure, sandboxed compute environments with high-grade isolation such as those found in honeypots [12] but with secured ingress and egress routes. Other key contributions are:

- The design and implementation of a microservices-based, air-gapped, on-premise deployable analytics platform providing a unified experience for BI and data science applications.
- A demonstration of the "headless BI" pattern, successfully integrating a powerful BI engine (Apache Superset) providing seamless data ingestion from an existing infrastructure.
- The development of a robust, single-tenant, role-based access control governance model leveraging LDAP-based AuthN, JWT, and HttpOnly cookie-backed AuthZ, express-session & Redis-backed session management, Redmine [1] backed organizational issue tracking, JupyterLab backed coding environment, socket backed secure real-time user-user, admin-all and code-user notifications, and additional features such as user session management, backup, etc.
- A fully containerized deployment model, where all core components—including microservices, Superset, and Jupyter notebooks—are orchestrated using Docker [16], ensuring portability, scalability, and simplified maintenance.

Although isolated containers exist in previous works, DAVi offers novel master–worker orchestration, centralized identity, integrated notification and alerts pipeline, headless BI integration, authenticated compute workflows, session management, tamper-proof activity logging, issue ticketing and encrypted backup/recovery, all designed for air-gapped high-security environments.

2 Related Work

To contextualize DAVi's contributions, we survey the current landscape of data analytics and visualization platforms. Existing solutions can be broadly divided into two categories – BI engines/suites and integrated data science platforms.

Apache Superset. At the core of DAVi's BI capabilities is Apache Superset, a modern, open-source data exploration and visualization platform. The platform was created by Maxime Beauchemin and joined the Apache Incubator in 2017, graduating to a Top-Level Project in 2021. Superset is a web-based application offering a rich feature set, including a powerful visualization builder with over 40 chart types, a powerful SQL IDE known as "SQL Lab", RBAC (Role Based Access Control) and RLS (Row Level Security), and a plugin architecture enabling custom visualizations. It is designed to connect to a wide array of SQL-speaking data sources from traditional relational databases to modern data warehouses and real-time OLAP engines. However, Superset doesn't natively provide workflows for advanced session governance, push-based real-time notifications, or coding environments.

Industrial and Academic Adoption of Superset. Preset [6] is the commercial, fully managed, SaaS version of Superset, created by Superset's original developers. Superset has emerged as an enterprise-grade platform, being adopted by numerous technology companies. For example, Dropbox chose Superset to standardize its internal analytics, consolidating nearly ten different tools into one, citing its security, extensibility, and alignment with their existing Python/Flask stack [7]. The Nielsen Corporation also replaced their previous patchwork of paid solutions with a unified dashboarding solution using Superset and embedded it's dashboards into their already existing client-facing applications, leveraging Superset's plugin architecture to add new visualization patterns [20]. Other industrial adopters include Microsoft Bing, who utilized a customized version of Superset for their internal self-service analytics platform [21], TransIT by SolDevelo [24] offers transportation analytics solutions, Funda.nl [10], a real estate marketplace, selected Superset due to its "fully open-source, low barrier" requirements and utilized Superset's embedded dashboard for market insights, and technology companies [11, 25] such as Airbnb, American Express, Lyft, Netflix, and Twitter. Within academia, Superset is being

increasingly adopted as a dashboarding layer for research data systems by leveraging its ability to create lightweight, reproducible visualizations on top of diverse data [8, 9, 14, 15, 17, 18, 23]. Additionally, as shown in Table 1, Superset provides the widest set of features among other BI platforms, offering both cost-effectiveness and strong extensibility. Combined with widespread industrial-scale and academic adoption, this made Superset a natural choice for a BI core around which DAVi is built.

Pure-Play BI Tools. Commercial BI platforms such as Tableau and Power BI are market leaders in self-service analytics and corporate reporting but are often proprietary and expensive. Their strengths are polished user interfaces, a vast library of out-of-the-box visualizations, and strong integrations with enterprise software ecosystems. However, these platforms are primarily designed for data consumption and exploration through graphical interfaces. When complex programmatic workflows, custom statistical or AI/ML modeling are required, users typically export data and switch to a separate environment like Python or R.

Integrated Data Science Platforms. Commercial SaaS solutions like Dataiku [3] and Domo [2] resemble DAVi's goal of providing a unified environment, but at a cost. Dataiku supports the entire machine learning lifecycle (MLOps) with strong governance, collaboration features and visualizations. While its MLOps functionality is more extensive than DAVi 's current scope, DAVi 's integration of Superset provides a more mature and flexible visualization layer than Dataiku's native charting tools. Moreover, DAVi relies on fully open-source components and a fully containerized, microservices-based model. The resultant independence from commercial vendors addresses the concerns of high-security environments, where data sovereignty, maintainability, and freedom from vendor lock-in are non-negotiable. To our knowledge, no existing open-source system combines DAVi's capabilities into a unified platform tailored for secure, air-gapped deployments.

Supporting Technologies for DAVi. In addition to analytics platforms, several other systems are essential to the design of DAVi. Message brokers such as Apache Kafka [13] provide fault-tolerant pipelines for streaming and archived data capture. DAVi leverages Kafka in this "data stream capture" role, persisting data streams into relational stores. Kafka has become the de facto standard for real-time streaming pipelines, enabling high-throughput ingestion and distribution of streaming data. On the BI side, Apache Superset serves as the foundation for DAVi 's visualization layer, providing a flexible and extensible alternative to commercial BI platforms. For secure, directory-backed authentication, LDAP [27] continues to be a widely adopted standard, often deployed in combination with Redis for session management and performance optimization. To support interactive features, DAVi draws upon real-time communication frameworks such as Socket.IO [22], which enable live user-to-user and code-to-user notifications within the interface. The entire system is served via Nginx [26], a production-grade web server that provides scalability, load balancing, and security hardening. Collectively, these technologies form the backbone upon which DAVi is built.

3 Usage Scenarios and Design Considerations

DAVi's development was motivated by specific requirements that could not be adequately met by any single off-the-shelf commercial product or open-source project. DAVi was required to serve as the central hub for monitoring and development tasks for a large organization with an existing in-house data collection platform with the following functional requirements:

- Legacy Integration: ingest high-volume telemetry data from the organization's existing data acquisition platform.
- RBAC: provide fine-grained, role-based, access control. Permissions cover datasets, dashboards, compute resources etc. Users are permitted to assume multiple roles simultaneously.
- Comprehensive BI Dashboard: provide a rich, interactive BI dashboard for visualizing parameters, identifying trends, and monitoring system status.
- Sandboxed Environment: offer an isolated coding environment accessible only via DAVi interface allowing authenticated users to safely execute custom code. This environment should be sandboxed ensuring that experimental code executions cannot compromise security or system integrity.
- Real-Time Notification Mechanism: provide real-time push notification service across different contexts—for example, between users, between a user and the coding environment, or between monitoring AI agents and end-users.
- Security: LDAP backed, fine-grained authentication, including separation of admin and user privileges. All significant user actions—from logins to dashboard access and code executions—must be logged in a secure and audit-able manner.
- Recoverability: offer automated snapshots of user workspaces volumes at regular intervals, allowing user data to be recovered in the event of container failure, with a tiered retention policy and point-in-time restore support.

Example Usage Scenarios. Apart from its deployment at its primary client, DAVi's modular and flexible design choices allow it to be used in a variety of other scenarios e.g., augmenting SCADA platforms in industrial plants to offer insights on plant health and using AI-models to raise alerts about any (impending) outlier or off-nominal behavior, augmenting security operations centers (SOC) to allow AI-based alerts for incipient attacks or intrusion attempts, augmenting schedulers for data centers to offer AI-based insights into resource utilization and efficiency, creating monitoring and servicing platforms for sensor network sending primary data and secondary telemetry about device health, etc.

4 System Design and Implementation

DAVi integrates multiple open-source technologies into a cohesive, security-conscious architecture. This section details the design and implementation of the core modules of the platform.

4.1 Technical Challenges

Developing the DAVi framework required overcoming several technical challenges that may be of independent interest.

Honeypot-grade Sandboxing. The stringent security concerns of the client organization required the creation of sandboxes offering honeypot-grade isolation [12] typically reserved for malware

Table 1: Comparison of Popular BI Platforms

Feature / Function	Tableau	Power BI	Qlik Sense	Looker	Apache Superset
Cost (Open-source or Free)	✗	✗	✗	✗	✓
Customizable	✗	✗	✗	✓	✓
Wide Database Support	✓	✓	✓	✓	✓
Self-Service BI	✓	✓	✓	✓	✓
Advanced Visualizations	✓	✓	✗	✗	✓
Real-Time Data Processing	✓	✓	✓	✓	✓
Scalability for Large Datasets	✓	✗	✓	✓	✓
Interactive Dashboards	✓	✓	✓	✓	✓
Cloud-Native	✗	✓	✓	✓	✓
Custom SQL Queries	✗	✓	✓	✓	✓
Community and Open Development	✗	✗	✗	✗	✓
Data Governance	✓	✓	✓	✓	✓
AI/ML Integrations	✗	✓	✗	✓	✓
Offline Mode	✓	✗	✗	✗	✓

✓= Supported; ✗= Not Supported;

Table 2: Feature Comparison of DAVi

Feature	DAVi	Apache Superset *Open Source BI*	Tableau / Power BI *Proprietary BI Software*	Domo *AI & Data Platform*	Alteryx *Analytics Automation Platform*	Dataiku *Universal AI Platform*
Interactive Dashboards	✓	✓	✓	✓	△	✓
No-Code / Drag & Drop Chart Builder	✓	✓	✓	✓	△	✓
Advanced SQL IDE (SQL Lab)	✓	★	△	△	✓	✓
Wide Variety of Native Chart Types	✓	✓	✓	✓	△	✓
Ability to Embed Dashboards	★	✓	✓	✓	✗	✓
Integrated Jupyter/Python Environments	✓	✗	△	✓	△	★
Isolated, On-Demand Environments	★	✗	✗	△	✗	✓
Visual / Low-Code Workflow Builder	✗	✗	△	△	★	✓
Version Control	△	✗	△	✓	✓	✓
Open-Source Core Technology	✓	✓	✗	✗	✗	✗
API for Programmatic Control	✗	✓	✓	✓	✓	✓
Centralized LDAP Authentication	★	✓	✓	✓	✓	✓
User Onboarding Workflow	★	✗	✗	✗	✗	✗
Granular RBAC	★	✓	✓	✓	△	✓
Session Management	★	✗	✗	✗	✗	✗
Real-time User/Code-to-User Notifications	★	✗	✗	✓	✗	△
Issue Tracking	★	✗	✗	△	✗	✓
Resource Monitoring	★	✗	✗	✗	✗	△
Automated Container Backups/Snapshots	✓	✗	✗	✗	✗	✗

✓= Supported; ✗= Not Supported; ★= Distinctive Feature; △= Limited Support.

analysis settings. However, the sandboxes still need to ingest data and send out alert notifications resulting in the requirement of a single ingress point to accept data and a single egress point to send notifications outside, while maintaining high-grade isolation. Thus, the container is prohibited from accessing the internet or performing activities such as network scanning, DNS resolution, and accessing elevated privileges, and is only allowed to access the least capable vectors in the file system. This custom isolation setting was achieved at different levels.

(1) A custom bridge was created with specific settings per user to serve as a middleware between Jupyter compute and the worker Docker system.
(2) The Jupyter Docker was configured to be launched with specific settings (no port publishing, least privileges, non-sudo access, limited file system privileges, etc).
(3) Firewall rules were applied so that only the Nginx of the worker is allowed to hit the IP of each user's Jupyter containers, one ingress port and one egress port being whitelisted for that custom bridge subnet. This was done to enable DAVi notifications and data transfer to operate.
(4) Proper Nginx configurations such as proxy were enabled from master to worker and then worker to Jupyter IP.

Firewall Conflict. The stringent security requirements created a conflict between firewall rules (IP tables and docker chains) meant

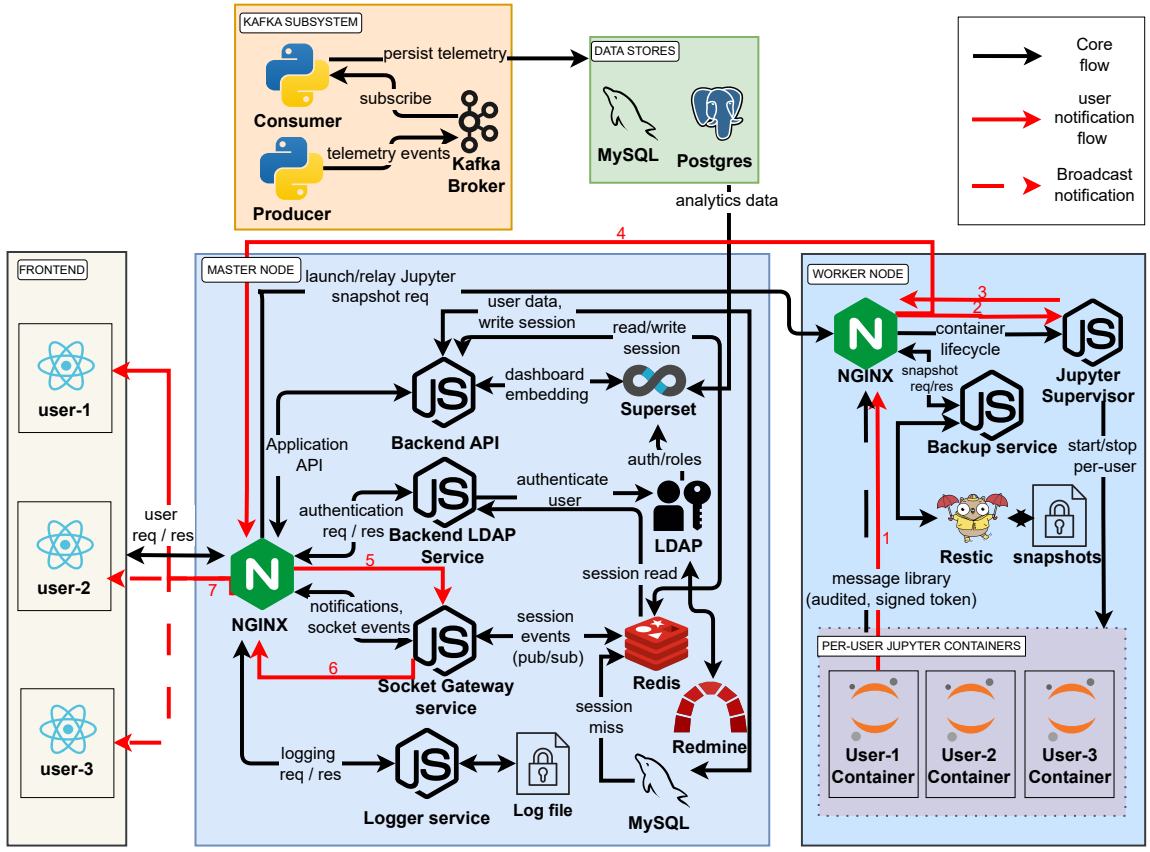


Figure 1: DAVi Architecture. A master-worker node separation enforces an additional barrier between sandboxed containers and the core utilities of the platform. The Superset/Kafka subsystem is integrated in a modular fashion. Notifications may originate from admin action or (more commonly) from user code running in a sandboxed container. The numbered arrows in the diagram depict the flow of notifications that can be sent to a specific user or be broadcasted to a specific subset of users.

for containers hosting the sandboxes (that required stricter rules) and containers hosting other critical system components such as Superset, Redis, etc (that demanded relaxed rules). To resolve this conflict, a two-plane master-worker configuration was adapted. This choice offered the added advantage of further isolating compute nodes from the rest of the system and improving contagion confinement.

Secure Backups at Scale. The straightforward solution of archiving and de-archiving the entire per-user volume is simple but makes every snapshot and restore proportional to the total volume size, regardless of the diff size. This may saturate the host I/O due to minute-level scheduling and large user assets. This was addressed by replacing tar-based backups with Restic [4] that offers content-defined chunking and de-duplication to reduce snapshot costs to roughly the size of the diff, enabling fast backups even for large volumes. As a result, snapshot and restore latency decreased substantially and repository growth aligned with true data churn. In addition, all snapshots are encrypted, and policy-based retention simplifies pruning and lowers operational overhead.

Compute IDE.. A design choice had to be made when crafting the compute environment. Various options were available such as the easy-to-integrate web-based Monaco editor with a Python LSP server for auto-completion, and the feature-rich, but harder-to-integrate, JupyterLab dev environment. DAVi opted for the latter and resolved integration issues via dockerization.

Issue Resolution. A choice was presented between two popular solutions: Trac and Redmine. Trac is simpler to use but has no default support for LDAP and no official Docker image while Redmine has a steeper learning curve. DAVi settled for the latter given the emphasis on security and long-term maintenance.

Tamper-proof Logs. The security considerations of the client demanded tamper-proof access and click logs. Various solutions including blockchain and simple encryption were considered. However, since logs are rarely accessed, there was little need for the heavy machinery of blockchain that would have introduced queuing and consensus delays, risking log modification before commit.

Instead, DAVi opted for encrypted, append-only logs that offer near-instant commits, lower latency, lower overheads and a similar level of security if implemented using strong PKI protocols.

Lack of Documentation. A significant hurdle faced during development was the lack of proper documentation for LDAP integration with Superset and Trac, and Superset by itself more generally. A separate project is being initiated to create helpful user and developer manuals for these services individually and their integration.

4.2 Overall Architecture

All external traffic enters through an Nginx reverse proxy. Requests are forwarded to the backend API (for REST/HTTP calls) or the socket gateway (for real-time communication). Authentication is LDAP-backed, with Redis and MySQL maintaining session state and actively logging all user activity. Apache Superset provides governed dashboards, Redmine provides Ticket management and Jupyter containers enable interactive compute. Kafka serves as a proof-of-concept streaming bus, bridging external telemetry into SQL databases. Figure 1 summarizes the overall system.

4.3 Master–Worker Compute Plane

DAVi employs as a two-plane system with distinct responsibilities.

Master node (control & services). This node hosts user-facing and control-plane services such as Nginx (public ingress), Backend API, Backend LDAP, Socket gateway, Logger, Redis, MySQL and Superset/Postgres. The master Nginx is the only public-facing entry point and enforces TLS and rate-limiting policies while proxying to the backend and socket services.

Worker node (secure compute). This node is dedicated to launching and supervising user-scoped Jupyter containers. A separate worker-side Nginx fronts only minimal endpoints needed for container lifecycle and controlled messaging. This step further bolsters sandbox security. Compute requests are validated on the master and relayed to the worker, which then launches an isolated container bound to the requesting user. The container may keep executing its code (and also keep generating notifications to be broadcasted) even if the associated user logs off. However, the system ensures that containers are never orphaned. Upon logging back in again, the user is immediately directed to their associated container to continue their code development and system monitoring work.

Master–worker integration. The planes communicate through authenticated internal APIs. The master issues short-lived signed tokens that the worker verifies before honoring lifecycle or message calls, thus confining blast radius: even if a container faces runaway code or even if it is compromised by malware or other contagion, impact is limited to that container on the worker plane alone. An admin may then terminate that container. The backup service (discussed later) can be used to restore user workspace data to a timestamp deemed to precede the contagion or bug.

4.4 Frontend: React Interface

The frontend is built with Vite (React), providing multiple components for user interaction:

Login & Navigation: Authenticates users against the LDAP service, payloading JWT (stored in HttpOnly, SameSite cookie) for subsequent requests.

CSV Uploader: A dedicated component allowing bulk manual onboarding of new users. Uploaded CSV files contain `id`, `name`, `email`, `password`, and `role`, which are transmitted to the backend for LDAP integration.

Dashboard : Superset dashboards are embedded within the DAVi UI, authorized via Superset guest tokens.

Admin Assistant: A rudimentary recommendation agent suggests role names during policy creation, easing administrative overhead.

Session Manager. : Users can review all their active sessions across different devices and browsers and have the ability to remotely revoke any session they do not recognize.

Admin-specific Components. : Admin has privileges to add users, create new roles and assign them to existing users, access the control panel, view click and access logs, fetch new onboarding requests and messaging components.

4.5 Backend API

The backend is implemented in Node.js/Express, providing endpoints for authorization, user management, data requests, and notification delivery.
Session Management: Redis serves as the store of ground truth for active sessions. Keyspace notifications publish expiration events, which propagate to the socket gateway for forced disconnection.
Superset Integration: The backend provides routes for access token retrieval, refresh tokens, and guest token issuance, ensuring seamless embedding of dashboards and charts.
CSV Upload: Uploaded CSV files are parsed, LDIF entries generated and applied to the LDAP directory in the containerized `openldap`.
Notification APIs: Secure endpoints allow trusted backends to inject messages into the socket service. Messages persist in MySQL for offline delivery to users who are currently not logged in.
Jupyter Control: The backend validates JWTs and sessions before launching Jupyter containers on worker nodes, enforcing strict user-container binding.
Jupyter lifecycle and routing: The backend exposes a `/api/launch-jupyter` control path that validates the user's web JWT and active session, then relays an authenticated request to the worker to start the user's container. The public route `/jupyter/{username}/...` is terminated at the master Nginx, which calls `/auth/validate-token` then proxies to the worker Nginx for that user's container.

4.6 LDAP Service

The LDAP subsystem governs identity and role assignments. Users are imported via CSV, with LDIF scripts mapping them into `superset_user` or `superset_admin` groups (extendable to multiple groups). The LDAP authentication endpoint verifies user credentials for DAVi, issues JWTs signed on user-name and user-roles. Sessions are mirrored in MySQL for auditability, with Redis acting

as the live store for expiration and invalidation. LDAP based authentication is done for general access to DAVi, but repeated when accessing the Superset and Redmine services.

4.7 Socket Gateway

Real-time interactions are managed by a Socket.IO service:

Authentication: Each socket connection is validated against JWTs or backend-issued tokens. Session IDs are mapped to socket IDs.

Notifications: Admin users may broadcast system-wide messages, target individuals, or message groups. Offline delivery is guaranteed by saving messages to MySQL.

Session Deactivation: Users may deactivate remote sessions; expired sessions trigger forced logout events through Redis pub/sub.

Integration: A Python WebSocket server can send JSON messages into the gateway, demonstrating extensibility to external services.

4.8 Superset

Dockerised Superset with modified configuration as per our requirements has been used.

LDAP-Integrated Authentication: User credentials are validated directly against the enterprise LDAP directory replacing the default Superset authentication mechanism ensuring a centralized and secure identity management.

Headless Integration: Due to Superset's authentication workflow, users may occasionally be required to re-authenticate themselves if they want to create a new dashboard directly using the DAVi interface. This happens only if Superset's own session cookie has either not been created yet (e.g., first-time login) or else has been cleared for some other reason. Other actions such as listing and viewing of existing dashboards, and even admin actions, are seamless and do not require re-authentication.

4.9 User Activity Logging Service

For compliance and auditing, this service listens for events from across the front-end and records them in an encrypted log store.

Client-side: only the action code and minimal relevant details are transmitted.

Server-side: the username is extracted by decoding the JWT, and the timestamp is generated, preventing client-side tampering. Each action code undergoes a server-side sanity check before being committed to the log, minimizing the risk that forged or invalid actions are recorded.

Encrypted Storage: Logs are stored in encrypted form using RSA public–private key encryption, ensuring that only authorized parties with the private key can decrypt and review activity records.

4.10 Backup Service

To protect user workspaces and enable point-in-time recovery, DAVi includes a dedicated Backup service that provides automated, per-user, volume-level snapshots with secure, space-efficient retention. The service runs on the worker plane alongside the Jupyter Supervisor (Figure 1), ensuring close integration with user containers while keeping repository access off the public ingress.

Scheduling & retention. A cron task performs automated backups every minute for any running container volume. The backups are pruned using a retention policy that keeps one snapshot every minute within the last hour, one per hour for 24 hours, one per day for 7 days, one per week for 4 weeks, and one per month for 12 months. This setup achieves the "Automated Container Backups/Snapshots" capability as outlined in Table 2.

Security & efficiency. Restic [4] provides encryption at rest for all snapshots and content-defined chunking/de-duplication, which reduces storage for repeated notebook assets and improves the speed of incremental backups. To further strengthen the security of workspace snapshots, the Backup Service is executed under a dedicated non-root Linux user, ensuring that Restic repositories and credentials remain inaccessible to Jupyter containers even in the event of sandbox compromise. Moreover, restore operations require the full 64-character snapshot hash and are additionally constrained by user-specific tags, limiting both accidental and malicious cross-user recovery attempts. Together, these measures provide strong practical isolation by combining strict privilege separation, protected repository ownership, and rigorous snapshot validation, thereby significantly reducing the risk of unauthorized access to backup data.

4.11 Kafka Streaming Subsystem

To demonstrate data stream ingestion, DAVi includes a Kafka-based proof of concept:

Producer: A Python producer emits telemetry events (like `device_id`, `timestamp`, `cpu`, `memory`, `temperature`) to `telemetry_topic`.

Consumer: A Python consumer subscribes to the topic, persisting records into a MySQL `timeseries` schema.

Visualization: Persisted database can be queried through Superset dashboards with RBAC, and authorized users may export results to Jupyter environment for analysis.

4.12 Operations and Administration

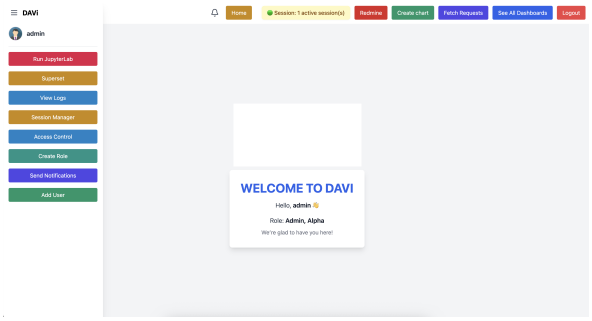
Three platform features support resilience and governance:

Snapshot-based backups. Regular snapshots capture MySQL/Redis state, LDAP directory data, Superset configuration and per-user Jupyter volumes via a Restic-based Backup service, enabling rapid restoration after failures and safe environment promotion.

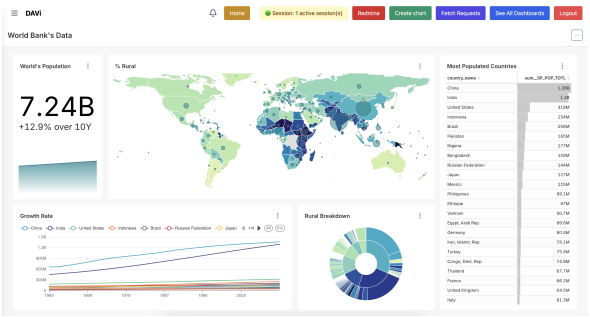
Security & efficiency. Restic provides encryption at rest for all snapshots and content-defined chunking/de-duplication reducing storage for repeated notebook assets and accelerates backups.

Issue tracking with Redmine. Administrative request workflows are integrated with Redmine for ticketing, release notes, and audit trails. The user console links directly to Redmine, aligning operational actions with recorded change history.

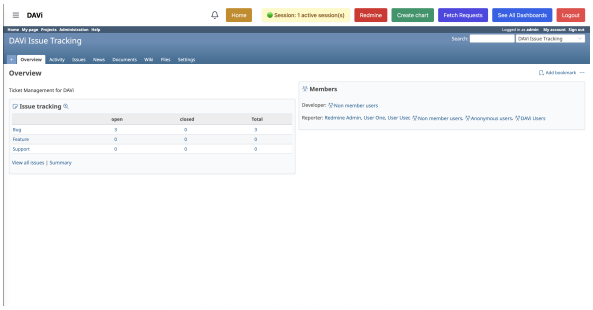
Encrypted Activity Logging. : Significant user actions are captured by the Activity Logger service. The logs are encrypted using an RSA



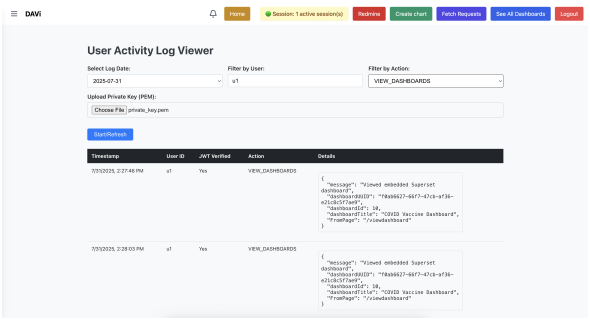
(a) Homepage – portions of the image redacted to preserve anonymity



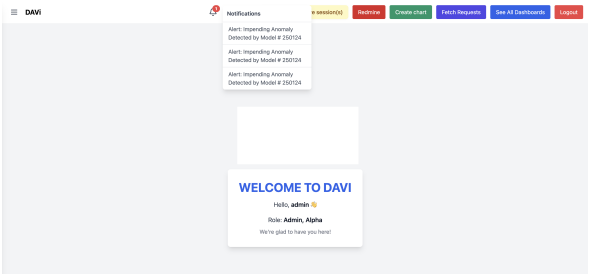
(b) View Dashboard



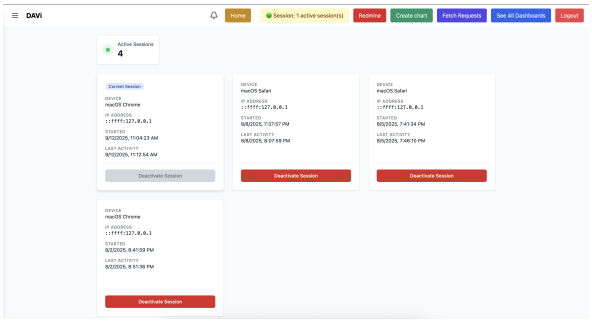
(c) Redmine Issue Tracker – allows users to raise issues to the admin for remedial action



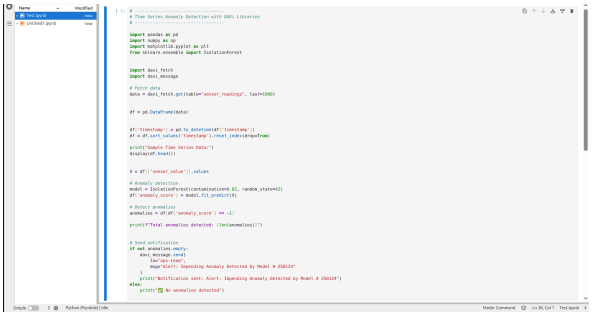
(d) Activity Logger – allows admins to securely inspect activity and click logs to monitor user activity



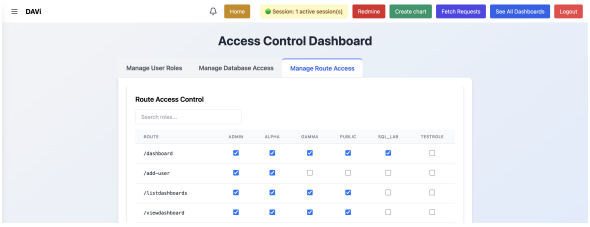
(e) A sample Notification generated by user code presented in Fig 2(g) – portions of the image redacted to preserve anonymity



(f) Session Manager – allows users to inspect sessions under their login and terminate inactive, suspicious sessions



(g) JupyterLab Environment – allows users to write custom code inside a sandboxed environment that can generate notifications and broadcast them to multiple users



(h) Access Control Panel – allows admins to add new users and assign one or more roles to them

Figure 2: Screenshots of the DAVi platform

key pair before being stored in an append-only file store, ensuring that their contents are protected at rest and can only be accessed by authorized personnel holding the corresponding private key.

4.13 Security Model

DAVi enforces a multi-layered security model:

- All ingress is routed via Nginx with TLS and rate limiting.
- Session expiration is enforced consistently across backend, socket, and frontend.
- APIs requiring cross-service integration (e.g., notifications) mandate a backend-shared secret.
- Fine-grained route permissions are defined and filtered per-role by the backend before rendering routes to clients.
- Database RBAC and RLS are provided by Superset.

Password and Token Handling Security. OpenLDAP ensures secure credential handling. As all communication will occur over HTTPS, the current design provides strong protection against token leakage, passwords are stored in LDAP only as salted hashes, and cleartext passwords exist only in LDAP's memory during verification. For Superset integration, refresh tokens are never stored, only short-lived access tokens are stored as `HttpOnly`, `SameSite=Strict` cookies. A controller-proxy microservice for managing Superset tokens was considered but rejected, as it would add complexity and latency without much meaningful security benefit.

4.14 Secure Compute and Container Isolation

Given the stringent security requirements, all user code being run inside worker containers is treated as high risk and confined to the worker plane:

Per-user custom bridge: Each container is bound to a user-specific route (`/jupyter/{username}/...`) and is addressed only through that bridge.

Ingress mediation via Nginx: Only the master Nginx is allowed to communicate with the container's direct IP. It intercepts the request, validates the token, and forwards it to the worker Nginx.

Firewall rules: Direct container IP traffic is blocked by default; only whitelisted ports are open to the worker Nginx. Master services are unreachable from within containers.

Single egress path (message library): Containers cannot contact internal services directly; the only sanctioned outward path is a message library endpoint that accepts a signed, short-lived token. Messages are validated, rate-limited, and audited before fan-out.

Blast-radius control: In the worst case, say malicious code being executed within a container, the sandboxing ensures that the contagion does not spread and can at worst crash the container itself, leaving other services (auth, sessions, dashboards, messaging) unaffected by design.

Load Balancing: A container that consumes more than the allotted resources (upto a grace period) gets terminated automatically. The owner must make a fresh compute request to spin up a new container. A user has at most a single container associated with them at any point of time.

5 User Interface and User Experience

DAVi emphasizes clarity, progressive disclosure of capabilities, and security-by-default. This section describes the main interaction flows, the rationale for key UX decisions, and how the frontend collaborates with the backend to provide responsive feedback under strict governance. An overview of the UI is shown in Fig. 2.

5.1 Login and Session Posture

Upon visiting the DAVi portal, the user is greeted with a minimal login screen. Credentials are validated against the LDAP directory and on success, a short-lived JWT token and a server session are established. The session posture is shown on the user interface (e.g., "Last access", "Devices", "IP") to make account activity visible, and the user can terminate other active sessions from within the account menu. If a session expires or is deactivated elsewhere, the UI receives a real-time logout signal and returns the user to the login screen with a concise explanation.

5.2 Role-Aware Navigation

After login, navigation and feature visibility are dynamically tailored to the user's role(s). Administrators see onboarding, routing-policy, and audit views; standard users see dashboards, notifications, and compute requests. The left-hand navigation lists only those routes authorized by policy; unauthorized routes are omitted rather than disabled to reduce confusion and minimize information disclosure. Any kind of unauthorized route access attempt is logged.

5.3 Dashboards and Embedded Analytics

Dashboards are embedded directly within the DAVi platform. Users can switch between curated dashboards (e.g., operational telemetry, request status, and resource usage). The embedding experience avoids re-login prompts and provides a consistent session lifetime with the web application. However, if a user wants to create a new dashboard, he will be prompted to Login to Superset inside the embedded iframe as the dashboard creation function of Superset does not rely on guest-token based mechanism as the case with just viewing the dashboards.

5.4 Bulk User Onboarding

DAVi supports two onboarding modes. Administrators may perform a **bulk upload** by submitting a CSV file with header `id,name,email,password,role`. The UI previews the parsed table for validation before processing, finally displaying outcomes of LDAP updates. Alternatively, a **manual entry** form allows adding a single user; the system internally converts the entry into a one-line CSV and routes it through the same onboarding pipeline.

5.5 Notifications and Real-Time Feedback

The shell maintains a real-time connection for system notifications:

- **Admin broadcast**: Administrators can send a message to all users (e.g., maintenance windows, policy changes). Online users receive a live toast; offline users see the message upon next login.
- **Targeted messaging**: Authorized operators can message a specific user or a named group (e.g., a distribution list). Delivery status and any denials are shown inline.
- **Compute to Frontend**: Push notifications can be unicasted or broadcasted from compute environments upon some

trigger to the DAVi frontend and can be further extended to Push notification service.

A badge counter surfaces unread notifications. Users can mark all as read; the count and list synchronize with the server.

5.6 Real-Time Session Governance

If a user deactivates one of their other active sessions, the UI for that device receives a real-time "force sign-out" event. The experience is designed to be explicit but unobtrusive: an interstitial explains the reason for deactivation, provides a link back to the login screen, and offers an "Appeal/Report" affordance when enabled by policy.

5.7 From Dashboards to Compute

Some workflows require data extraction into a notebook environment for exploratory analysis. The interface provides an easily identifiable button titled "Run JupyterLab" that can be used to request compute. The request is authorized with the same user identity and roles. On approval, the DAVi UI surfaces connection details and a short-lived message token for the user's container. System status and basic resource metrics (e.g., CPU, memory) are periodically polled and displayed; if the container is idle or exceeds limits (upto a grace period), the UI communicates impending shutdown windows and preserves session state for resumption.

5.8 Error Handling and Resilience Cues

Across flows, the UI emphasizes clear, actionable errors:

- **Authentication errors**: concise cause (invalid credentials, expired session) with guidance to retry.
- **Policy denials**: specific reason (insufficient role, missing approval) and a one-click access request where applicable.
- **Upload failures**: validation feedback for bulk onboarding.
- **Redmine**: Other requests that require admin intervention can be raised as a ticket on Redmine.

Transient network issues trigger non-blocking banners and automatic retry when safe. Superset, Ngnix and DAVi's logging mechanism records all such events.

5.9 Performance metrics

As DAVi is developed for sensitive applications, the system is evaluated through qualitative, architecture-driven, non-sensitive metrics derived from controlled synthetic tests providing a structured and defensible assessment of DAVi 's operational profile in Table 3.

5.10 Administrator Experience

Administrators have a unified console for:

- reviewing new onboarding requests for approval,
- managing access control like new role creation, user role change, role-to-route mapping, manage database access etc.
- auditing notifications, session and logging activity
- assessing container health signals relevant to end users.

Design intentionally separates *policy authoring* from *operational messaging* to reduce accidental misconfiguration and support clearer accountability.

Table 3: Qualitative Performance Metrics

Metric	Description
Sandbox Startup Latency	Time from Launch Request to Jupyter Readiness; indicates isolation overhead and responsiveness.
Backup Creation Latency	Restic snapshot time and delta size; shows storage efficiency and snapshot scalability.
Backup Restore Latency	Time to Restore a User Volume Snapshot; Confirms Recovery Reliability.
Notification Delivery Latency	Time from Container Message to Frontend Receipt; Verifies Real-Time Alerting.
Dashboard Embedding Overhead	Superset dashboard load time; measures UI embedding and Superset–NGINX–token integration efficiency.
Access Authorization Time	Backend + Superset RBAC evaluation time; reflects governance accuracy and authorization overhead.
Session Governance Latency	Propagation Time for Session Expiration/Forced Logout; Checks Revocation Behavior.
Compute Resource Stability	CPU-heavy stress tests; evaluates sandbox robustness and auto-kill triggers.
Kafka-to-MySQL Throughput	Synthetic Events/sec Persisted; Validates Streaming Ingestion Performance.
API Response Latency	Control-plane API Turnaround Time: Shows backend responsiveness.

6 Conclusion and Future Directions

DAVi is a secure compute–visualization platform addressing client requirements such as stringent security, scalability and legacy integration, allowing users to move fluidly between high-level monitoring on an interactive dashboard to programmatic anomaly investigation within a secure, unified web application. DAVi provides a modular framework to build other integrated data visualization and analytics platforms, by providing a blueprint for similar systems in other data-intensive scientific and industrial domains. The platform effectively empowers its users with a powerful, secure, and unified environment, enabling them to transform vast streams of complex operational data into the actionable intelligence necessary for operational success.

Future directions of work include integrating user-friendly tools such as allowing natural language queries via LLM integration, comprehensive failure-mode and reliability testing and creation of a *marketplace* of code snippets and workflows allowing scientists and analysts within the same org to share code and data flows.

The platform is fully integrated and currently deployed as a pilot within a controlled, high-security environment. Although core features are fully functional and the platform enforces strong isolation and governance, residual risks inherent to containerized compute such as potential replay attacks, impersonation risks in service communication, and container-escape vectors, remain subject to ongoing mitigation. Several enhancements, such as mTLS-based inter-service authentication, SBOM generation, stricter container profiles, and formal penetration testing are part of the platform's planned hardening phase before production roll-out.

Acknowledgments

This work was supported by a grant from the Indian Space Research Organization via the IIT Kanpur Space Technology Cell project number STC/CS/2023664Q. The authors thank the Center for Developing Intelligent Systems, especially Mr. Harikrishna P, and Mr. Manish Kumar for helpful consultations. PK thanks Microsoft Research and Tower Research for research grants.

References

[1] Redmine 2006. *Redmine: A Project Management Web Application.* Redmine. https://www.redmine.org

[2] Domo, Inc. 2011. *Domo: The Modern AI and Data Products Platform.* Domo, Inc. https://www.domo.com/

[3] Dataiku 2014. *Dataiku: The Universal AI Platform.* Dataiku. https://www.dataiku.com/

[4] restic 2014. *restic: Backups done right!* restic. https://restic.net

[5] The Apache Software Foundation 2021. *Apache Superset.* The Apache Software Foundation. https://superset.apache.org/

[6] Preset, Inc. 2022. *Preset: Modern BI Powered by Open Source Apache Superset.* Preset, Inc. https://preset.io/

[7] Dropbox Tech Blog. 2020. Why we chose Apache Superset as our data exploration platform. https://dropbox.tech/application/why-we-chose-apache-superset-as-our-data-exploration-platform.

[8] R. Bovkir and A. C. Aydinoglu. 2021. Big Urban Data Visualization Approaches Within The Smart City: Gis-based Open-source Dashboard Example. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* Xlvi-4/w5-2021 (2021), 125–130. doi:10.5194/isprs-archives-XLVI-4-W5-2021-125-2021

[9] Open edX. 2023. Open edX Aspects: Superset Decision Record. https://docs.openedx.org/projects/openedx-aspects/en/open-release-palm.master/decisions/0003_superset.html.

[10] Funda.nl Engineering. 2020. How We Decided on Using Apache Superset for Embedded Analytics. https://blog.funda.nl/how-we-decided-on-using-apache-superset-for-embedded-analytics/.

[11] Apache Software Foundation. 2021. The Apache Software Foundation Announces Apache Superset as a Top-Level Project. https://news.apache.org/foundation/entry/the-apache-software-foundation-announces70.

[12] Javier Franco, Ahmet Aris, Berk Canberk, and A. Selcuk Uluagac. 2021. A Survey of Honeypots and Honeynets for Internet of Things, Industrial Internet of Things, and Cyber-Physical Systems. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2351–2383. doi:10.1109/comst.2021.3106669

[13] Jay Kreps, Neha Narkhede, and Jun Rao. 2011. Kafka: a Distributed Messaging System for Log Processing. *Proceedings of the NetDB Conference* (2011).

[14] Jinqing Lian, Xinyi Liu, Yingxia Shao, Yang Dong, Ming Wang, Zhang Wei, Tianqi Wan, Ming Dong, and Hailin Yan. 2024. ChatBI: Towards Natural Language to Complex Business Intelligence SQL. arXiv:2405.00527 [cs.DB] https://arxiv.org/abs/2405.00527

[15] David McVicar, Brian Avant, Adrian Gould, Diego Torrejon, Charles Della Porta, and Ryan Mukherjee. 2023. Smartflow: Enabling Scalable Spatiotemporal Geospatial Research. In *IGARSS 2023 - 2023 IEEE International Geoscience and Remote Sensing Symposium*. Ieee, 1193–1196. doi:10.1109/igarss52108.2023.10283095

[16] Dirk Merkel. 2014. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux Journal* (May 2014). https://www.linuxjournal.com/content/docker-lightweight-linux-containers-consistent-development-and-deployment

[17] Petito Michele, Francesca Fallucchi, and Ernesto William De Luca. 2019. Create Dashboards and Data Story with the Data & Analytics Frameworks. In *Metadata and Semantic Research*, Emmanouel Garoufallou, Francesca Fallucchi, and Ernesto William De Luca (Eds.). Springer International Publishing, Cham, 272–283. https://link.springer.com/chapter/10.1007/978-3-030-36599-8%5F24

[18] Ameir El Ouadi, William Knowlton, Adrian Pimentel, and David Beskow. 2025. Gaining the Edge: Visualizing Information Advantage through Machine Learning-Driven Dashboards. In *Proceedings of the Annual General Donald R. Keith Memorial ConferenceWest Point, A Regional Conference of the Society for Industrial and Systems Engineering, New York, USA April 24, 2025.* https://www.ieworldconference.org/content/WP2025/Papers/GDRKMCC25%5F11.pdf

[19] Fernando Pérez and Brian Granger. 2014. IPython: A System for Interactive Scientific Computing. In *Proceedings of the 13th Python in Science Conference.*

[20] Preset.io. 2020. Nielsen uses Apache Superset for scalable analytics. https://preset.io/blog/2020-08-11-nielsen-superset/.

[21] Preset.io. 2021. How the Bing Team Heavily Customized Superset for their Internal Data Platform. https://preset.io/events/how-the-bing-team-heavily-customized-superset-for-their-internal-data/.

[22] Guillermo Rauch. 2012. Socket.IO: Real-time Bidirectional Event-based Communication. *Open Source Project* (2012). https://socket.io/

[23] Guilherme H. Soares and Miguel A. Brito. 2023. Business Intelligence Over and Above Apache Superset. In *2023 18th Iberian Conference on Information Systems and Technologies (CISTI).* doi:10.23919/cisti58278.2023.10211907

[24] SolDevelo. 2021. How to Set Up Superset: Case Study Based on the TransIT Project. https://soldevelo.com/blog/how-to-set-up-superset-case-study-based-on-the-transit-project/.

[25] Apache Superset. 2025. In the Wild: Organizations Using Superset. https://github.com/apache/superset/blob/master/RESOURCES/INTHEWILD.md.

[26] Igor Sysoev. 2004. NGINX: High Performance Load Balancer, Web Server, and Reverse Proxy. *Open Source Project* (2004). https://nginx.org/

[27] M. Wahl, T. Howes, and S. Kille. 1995. Lightweight Directory Access Protocol (LDAP). In *Rfc 1777.* https://www.rfc-editor.org/rfc/rfc1777