PLANTPULSE · ARCHITECTURE OVERVIEW

How the four products interlock
The complete architecture as one system.

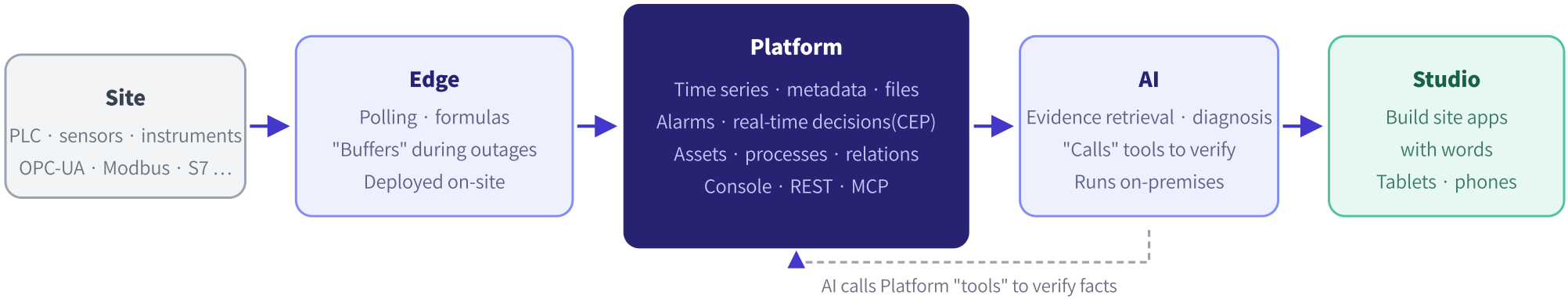
Where field signals pass before reaching the screen, what runs where when several components can perform the same task, and what remains after a disconnection—all summarized in **eight pages**.

4	1	0	8
products — edge · platform · AI · studio	data ledger — consolidated in one place	internet requirement — unchanged in closed networks	pages · 2026.09

01 · At a glance

From signal to screen — what the four products do

In one sentence — **Edge collects, Platform stores and evaluates, AI explains, and Studio turns it into screens.** Each can be installed separately, but **there is one data ledger**.



Installed separately, fail separately

The four are installed on separate machines. **If one stops, the others keep working** — Edge keeps collecting when Platform is unavailable, and Console still helps without AI.

Nothing leaves the premises

All four run **on customer equipment**. With no internet requirement, they operate **unchanged on isolated networks**, and both models and data remain on-premises.

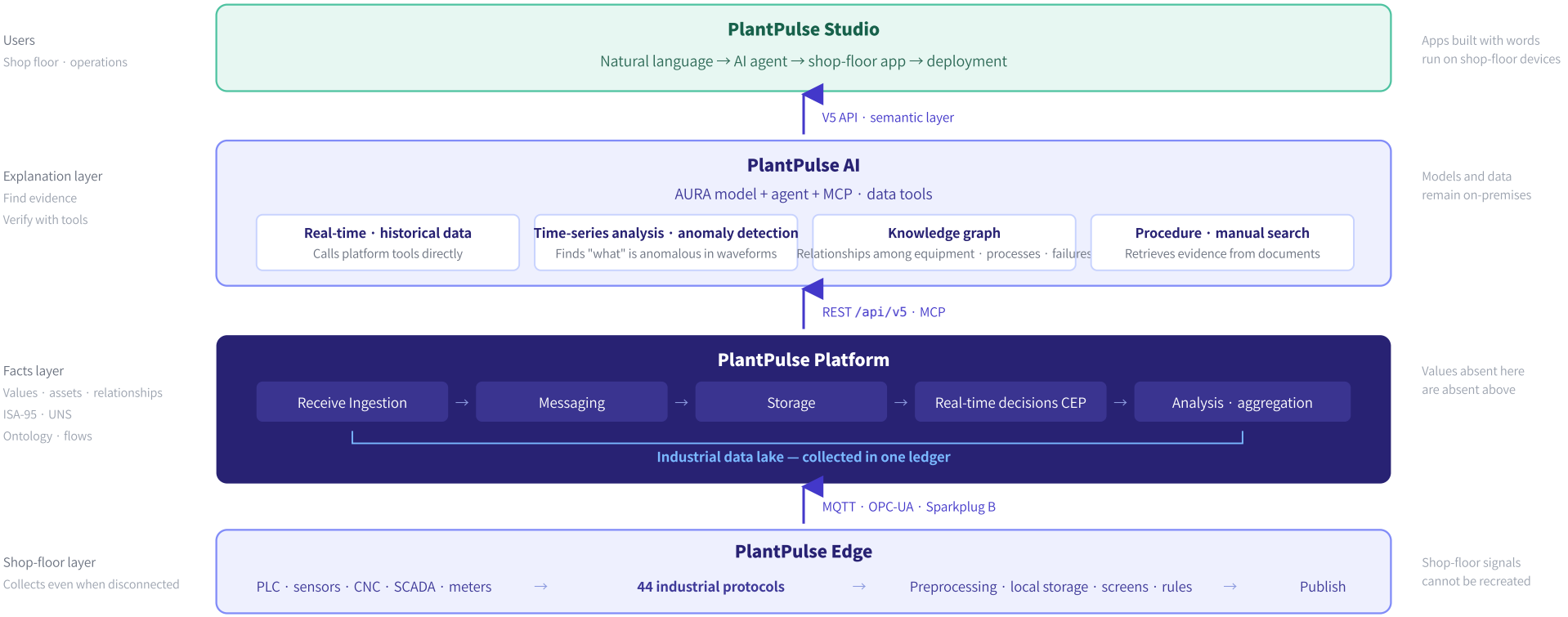
Contents

02 Stack — four layers, each supporting the one above 3
03 Overall architecture — components and paths between them 4
04 Technology stack — what it is built with 5
05 How data flows 6
06 Product boundaries — when several places can do the same job 7
07 What happens during outages 8
08 Networks · ports · firewalls 9
09 Security · points traversed by credentials 10
10 Deployment sequence · further reading 11

02 · Stack

Four Layers — Each Supports the One Above

It starts on the shop floor and ends on the screen. **Each layer uses only what the layer below provides**, and the layers are linked by **standard protocols and public APIs** — so replacing one layer leaves the others intact.

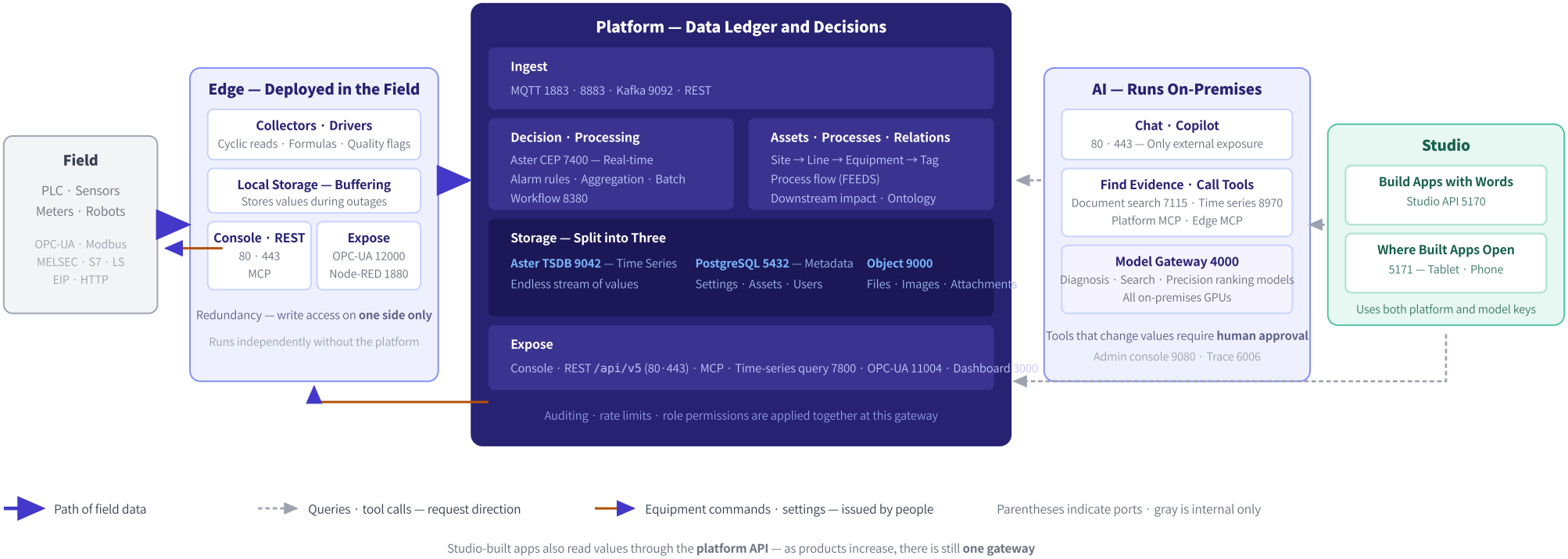


The value of layers is that they are "replaceable." You can deploy Edge first to collect shop-floor data and add the Platform later, or use only the Platform without AI and add it afterward. This is possible because **the layers are linked by standard protocols and public APIs** — if they were linked by paths that work only between one vendor's products, you could not choose that order.

03 · Overall Architecture

Components and the Paths Between Them

Thick arrows show the path of **field data**. Dashed lines show the **direction of requests** (queries · tool calls), and parentheses indicate ports. **Gray is used only within the internal network**.



Studio-built apps also read values through the **platform API** — as products increase, there is still **one gateway**

Why storage is split into three. Values arrive endlessly, settings rarely change, and files are simply large — **putting all three with different properties in one place** makes a choice for one harm the others. Instead, there is **one gateway**, so clients do not need to distinguish among the three.

04 · Technology Stack

What It Is Built With — and Why

We selected proven open source, **adapted it, and bundled it into one stack.** We built components in-house to meet **requirements unique to industrial sites**; elsewhere, we use industry standards as-is.

Edge — the layer that operates independently on site

Protocols	**44 industrial protocols** — OPC-UA · Modbus · MELSEC · S7 · LS · EIP · PROFINET · IEC 61850 · DNP3 · HART · FINS · BACnet · CANopen · M-Bus · SECS/GEM · FOCAS · MTConnect · ROS 2 …
Local storage	Buffers values **inside the box** while disconnected
Edge logic	Expressions · rules · **Node-RED** flows
Output	OPC-UA server · MQTT · **Sparkplug B** · REST · MCP

Platform — the data ledger

Time series	**Aster TSDB** — Derived from Apache Cassandra and **maintained by KOPENS**. Tuned to sustain unending writes
Metadata · configuration	**PostgreSQL** — For assets · users · configuration that must be correct **exactly once**
Files · images	**S3-compatible** object store
Messaging	**Aster MQTT** · **Apache Kafka** — The path from the field and the path within the platform
Real-time decisions	**Aster CEP** — Applies rules to streaming values
Analytics · batch	**Apache Spark** · **Kyuubi** · **Gravitino**(catalog) · **Kestra**(workflow) · **Temporal**
Cache · dashboard	**Valkey** · **Grafana**

AI — the layer running on premises

Model	**AURA** — Diagnosis · search · precise alignment. **KOPENS handles everything from data design through training · evaluation · quantization** in-house
Serving	Inference engine on in-house GPUs + **OpenAI-compatible gateway** — Uses existing clients as-is
Evidence retrieval	Vector search + reranking. Covers procedures · standards · failure histories
Knowledge graph	Graphs relationships among equipment · processes · failures
Tool integration	**MCP** — Invokes platform and edge tools directly
Observability	Tool execution ledger · traces · circuit breakers

Studio — the layer built with words

App creation	Natural language → agent → **field app**. Created apps open directly on tablets and phones
Storage	**PostgreSQL** — Included with the stack
Data	Reads through the platform **V5 API** — No separate connection is created

Standards followed

Model · structure	**ISA-95** hierarchy · **UNS**(Unified Namespace) · ontology relationships
Communication	**OPC-UA** · **MQTT** · **Sparkplug B** · Modbus and 41 others
Integration	**OpenAPI** · **MCP** · OpenAI-compatible endpoints

Why we built some components in-house. Time series · real-time decisions · messaging **have field requirements that differ from general-purpose products** — data arrives endlessly by the second, surges after disconnections, and must not be reordered. We therefore **adopt and maintain open source ourselves** for those three. For the rest, we use **industry-proven products as-is** — changing what does not need fixing **only increases the maintenance burden.**

05 · Path of Flow

From a Sensor Point to the Screen

The path of a single value is listed in order. **Where latency occurs and where data accumulates** are all covered in this table.

Step	Location	What happens
① **Read**	Site → Edge	PLCs and instruments are read at the configured **interval**. Protocols include OPC-UA · Modbus · MELSEC · S7 · LS · EIP · HTTP
② **Refine**	Edge	Formulas create **derived values** and mark quality — **a value of 0 and a failed read** are distinguished from this point
③ **Send**	Edge → Platform	MQTT(1883 · TLS 8883) or platform REST. SparkplugB is also used
④ **Receive**	Platform	Data is received through messaging and processed **in order for each tag**
⑤ **Store**	Platform storage	**Time series** in Aster TSDB(9042) · **metadata and configuration** in PostgreSQL(5432) · **files and images** in object storage(9000)
⑥ **Evaluate**	Platform	Alarm rules and **real-time event processing(CEP, 7400)**. Aggregation and batch processing run separately
⑦ **Retrieve**	Platform	Time-series query engine(7800) · REST(`/api/v5`) · console screen
⑧ **Explain**	AI	It finds evidence in documents and history, **"calls" platform tools** to verify facts, then responds
⑨ **Build**	Studio	Builds screens for use at the site — accessible on tablets and phones

Reverse direction — what people "send down"

Equipment commands	Commands travel from the console or API through the edge to the PLC. Because this is **the reverse of reading**, controls are applied in reverse as well
AI writes	When AI calls a tool that changes a value, it requires **human approval**. **Write tools are not exposed** on paths without approval
Deploy configuration	Tag and connection definitions are managed on the platform and retrieved by the edge

> **"0" and "failed to read" are distinguished from the start.** If a collection failure is recorded as a value of 0, **equipment that no one has inspected is reported as "normal"**. Quality is marked at the edge, and storage, screens, and AI **carry that mark through to the end.**

06 · Product Boundaries

When Multiple Places Can Do the Same Work

The most common question during adoption reviews is **"Where is that done?"**. We selected areas where the four appear to overlap and described **the criteria for choosing**.

Task	Where It Can Be Done	Selection Criteria
Value processing · derivation	**Edge expressions** / platform processing	**Use the edge if needed immediately on site** (it works even when disconnected). Use the platform for cross-equipment views or retention
Anomaly detection	**Platform alarms · CEP** / AI	Use **alarms and CEP** when rules are clear. Use AI when "why" must be explained in sentences
Screen creation	**Platform console** / studio	Use the console for standard operator screens. Use the studio for **dedicated apps used by site personnel**
Search documents · history	**AI search**	Use when the targets are procedures · specifications · failure history. The platform is faster for tag value queries
External system integration	Platform REST · MCP / edge REST · MCP	**Use the platform if enterprise-wide data is needed**, **and the edge if only that box is needed**. The edge responds even when disconnected from the platform
Adding an AI assistant	Platform MCP / AI / edge MCP	Use the platform to access assets · alarms · ontology (22 tools). Use the edge for only one site box (6 read tools)

Three Boundary Criteria

① **Latency** — Must it respond immediately on site, or can it wait a few seconds? ② **Retention** — Is it disposable after passing, or must it be revisited over several years? ③ **Permissions** — Must it be able to "write" values there?

Do Not Put Something That Works in "Both" in Both

Putting the same calculation on both the edge and platform causes **the two values to diverge.** Once they diverge, no one knows which is correct — **choose one place and have the others read from it.**

What Each Product Can Do "Alone"

Edge standalone	Collection · calculation · local screens · output as an OPC-UA server · REST/MCP responses. **It operates without the platform**
Platform standalone	Storage · alarms · console · analytics · API. **External systems can also input data directly** without the edge
AI	The platform is required to verify "facts" — because it reads values through tools
Studio	It accesses **both** the platform API key and AI gateway (installation checks those two first)

07 · When Disconnected

What Remains When the Link Breaks

Factory networks disconnect. Therefore, we designed for **what continues while disconnected and what catches up after reconnection**.

Disconnected Segment	During Disconnection	After Reconnection
Edge ↔ Platform	**Collection continues.** The edge stores data locally	**It sends the stored data** — the gap is filled
Edge ↔ PLC	Only that connection is lost. Other connections continue operating	It reconnects automatically. The gap **remains marked as "unreadable"** — it is not filled with 0
AI ↔ Platform	AI **cannot verify facts.** It therefore does not fabricate and **says so**	It recovers immediately — AI holds no state
Studio ↔ Platform	Existing apps open, but **field values do not arrive**	It recovers immediately

Edge "buffering" is central to this architecture

Field signals **cannot be recreated.** Once that moment is missed, it is gone — therefore, storage is placed **where data is collected**, not where it is sent. Even if the platform is unavailable for several hours, **that period does not remain as a data gap.**

Redundancy — write access belongs to "one side only"

When deploying two edge nodes, **only one has permission to write to the PLC**. If both have permission, two nodes write to the same equipment. Therefore, access is transferred in the order **disable → verify → enable**.

How a blockage appears — where operators should look first

Edge Status	The edge console's connection list and **health status**(UP/DOWN by component)
Platform Ingestion	Console **monitoring** — view per-second ingestion · queues · storage status on one screen
Transfer Status	Edge transfer status — the **backlog volume** is shown

"No impact" and "unknown" are different. Without mapping the flows between equipment, the question "How far does the stoppage propagate if this equipment stops?" receives **an empty answer**. This does not mean there is no impact; it means **nothing has been mapped** — therefore, the screen and API distinguish and report the two states.

08 · Networks and Ports

What Connects Where — at a Glance

This lists only what is needed to design the firewall. **What must be externally accessible** and **what is used only internally** are separated.

User Access Points

Product	Port	Purpose
Platform	**80 · 443**	Console and REST API. **The web UI and API use the same address**
Edge	**80 · 443**	Edge console and `/api/v1`. External calls **require HTTPS**
AI	**80 · 443**	Chat UI — **the only external exposure**
Studio	**5171**	Where created apps open. **Blocking it prevents previews and deployed apps from opening**

Product-to-Product Paths

Edge → Platform	**1883**(MQTT) or **8883**(TLS), plus Platform **443**
AI → Platform	Platform **443** — REST and MCP use the same entry point
Studio → Platform	Platform **443**(API key)
Studio·AI → Model	Internal model gateway(**4000**). **No traffic leaves the internal network**
External SCADA → Edge	**12000 · 12443** — only when Edge serves as an OPC-UA server

🔴 **Immediately after installation, ports are "broadly open" — narrowing access is the operator's responsibility.** The installer registers the ports required for operation with the firewall. Of these, **database · messaging · internal listener** ports are generally needed **only within the internal network**. Review them during deployment and retain **only those that need external access** — each product's documentation states **for every port, "Is there a reason to leave it externally accessible?"**

Air-Gapped Network

None of the four **requires internet access.** Installation uses an offline bundle, and the models also run internally. The absence of outbound traffic is **the first condition verified in network design**, so it is stated here.

09 · Security

Where Accounts and Keys Pass

This summarizes where and how authentication occurs, and **what must be changed during deployment**.

Path	Method	What to know
User → Platform	Console login · role	Write access varies by role
Application → Platform	Exchange issued token **for a key**	**HTTP 200 does not indicate success** — check both the response status and key. Also, **503 does not mean "wrong key"**
Application → Edge	`X-API-Key` or Bearer	**HTTPS is enforced** by default, so plaintext requests are rejected even with the correct key
AI → Platform	Platform API key	Tools are **mainly read-only**. Tools that change values require **human approval**
→ Model Gateway	Gateway key	During **installation**, Studio verifies this key and model name by making an actual call

What must be changed during deployment

Each product has an **initial password**, stated directly in the documentation. **It is not hidden** because it is included in the installation anyway, and documenting it makes it possible to **instruct users to change it** — each product has an "Change Initial Password" document as a required commissioning procedure.

Rejection is better "before installation"

Before **installing Docker**, Studio installation makes an actual call using the supplied credentials. If the key is wrong, **it stops immediately** — preventing "installation completed, but everything fails on first use." However, **it does not stop if the server cannot respond (5xx)**. This avoids treating a server under maintenance as a "wrong key."

Audit and Logging

Platform	API calls and changes are recorded in the audit log. A reason is also recorded when **AI acknowledges an alarm**
Edge	REST calls are recorded in the access log, and **write-access changes are logged separately**
AI	A separate **tool execution ledger** records which tool was called with which arguments

10 · Deployment Order

What to Install First and Where to Read More

There is no need to install all four at once. Because **each preceding step is a prerequisite for the next**, the order reflects the dependencies.

Order	Component	Completion criterion
1	**Platform**	You can log in to the console and **have changed the initial password**. All stores are up
2	**Edge**	It is registered with the platform, and **one connection is sending values**. Disconnect and reconnect it to verify that **buffered data catches up**
3	Assets · Tags · Flows	The equipment hierarchy is established, and **the process flow has been mapped** — a prerequisite for downstream impact queries
4	Alarms · Screens	The **screen operators view every day** has been selected
5	**AI**	The model gateway is running and **can call platform tools**
6	**Studio**	One field app **opens on a tablet**

> **Be sure to test "disconnecting and reconnecting" in step 2.** This function is not apparent during normal operation and **becomes apparent when it is actually needed**. Verifying it once during the deployment review prevents later questions about "why the data was missing then."

Product Documentation

Platform	docs.plantpulse.io /ko/platform	**Data Lake Technical White Paper**	/ko/platform/whitepaper
Edge	docs.plantpulse.io /ko/edge	**AURA Model Technical White Paper**	/ko/ai/overview/whitepaper
AI	docs.plantpulse.io /ko/ai	**Process Diagram**	/ko/platform/user-guide/process
Studio	docs.plantpulse.io /ko/studio	**MCP Server**	/ko/platform/api/mcp · /ko/edge/api/mcp

Contact

Phone · Email	1588-5268 · webmaster@kopens.com
Inquiry · Demo	kopens.com/ko/contact

㈜한국오픈솔루션 · KOPENS, Inc. © 2026 KOPENS. 플랜트펄스® is a registered trademark of ㈜한국오픈솔루션. This document Rev. 2026.09.