PORTAINER.iO

E-Book

HIERARCHY OF NEEDS:

What Industrial Automation Engineers Need from a Container Platform

08.08.25

THE FIELD ENGINEER BECAME THE IT DEPARTMENT.

You won't find them in the data center. You won't find them in sprint planning.

And you probably won't see them pushing code to a GitHub repo. But you will find them on factory floors, in wind turbine substations, at wastewater treatment facilities, on railway switching stations, and deep inside production environments where latency, uptime, and operational resilience are non-negotiable. They are Industrial Automation Engineers. And in an increasingly digital world, they've quietly become one of the most complex, multidisciplinary roles in modern technology.

Historically, these engineers were responsible for programming PLCs, commissioning SCADA systems, wiring sensors, and ensuring that mechanical systems behaved as expected. But in the last decade, their world has changed dramatically. The rise of connected infrastructure, Industry 4.0 initiatives, and the growing demand for remote diagnostics, predictive analytics, and edge compute has pulled a new type of responsibility into their orbit: IT.

And not just traditional IT. We're talking about distributed systems, orchestration platforms, secure networking, containerized applications, image registries, GitOps pipelines, and zero-trust architectures. Things that were once the domain of enterprise IT departments are now appearing in the field; on rugged edge gateways, in industrial control cabinets, and across sprawling fleets of embedded compute units.

The OT engineer is no longer “just” an automation expert. They’re now expected to maintain Linux environments, understand TLS configurations, configure MQTT brokers, troubleshoot Kubernetes clusters, recover from container crashes, and enforce cyber security policies, often with no dedicated IT team nearby. In essence, they’ve become the IT department in a single person, only instead of operating in a climate-controlled data center, they’re doing it in places with dust, vibration, poor connectivity, and unforgiving uptime constraints.

This isn’t just scope creep.
It’s an identity shift.

And yet, the tools being pushed into their environments (containers, Kubernetes, GitOps platforms) have been designed primarily for IT specialists, not for engineers with an automation background. The ergonomics of Kubernetes, for instance, assume fluency with YAML, an understanding of abstractions like ingress controllers and service accounts, and a workflow driven by source control and pipelines. For someone steeped in fieldbus protocols, Modbus registers, and IEC 61131 ladder logic, this is not just unfamiliar, it’s alien.

So when the OT engineer is asked to “just run this container,” it’s not a matter of syntax. It’s a matter of perspective. The person being asked to take responsibility for deploying containerized workloads is often the same person who also manages firmware updates, electrical schematics, loop tuning, and SCADA alarms. They don’t have the luxury of specializing. They need tools that let them move fast, stay safe, and stay online, without needing to learn cloud-native development patterns from scratch.

This is where most platforms fail. They’re built for IT operators. For SREs. For DevOps engineers sitting in an office with stable internet. Not for the engineer in an oilfield or factory trying to deploy a system that has to stay up for the next decade with zero maintenance windows.

SO THE QUESTION IS SIMPLE:

DOES YOUR CONTAINER PLATFORM RESPECT THE REALITIES OF INDUSTRIAL WORK?

Because this persona (the Industrial Automation Engineer) isn't hypothetical. They're already out there, maintaining mission-critical systems in environments where rollback means rolling a truck, and where failure isn't a ticket, it's a production outage.

And they have very real needs.

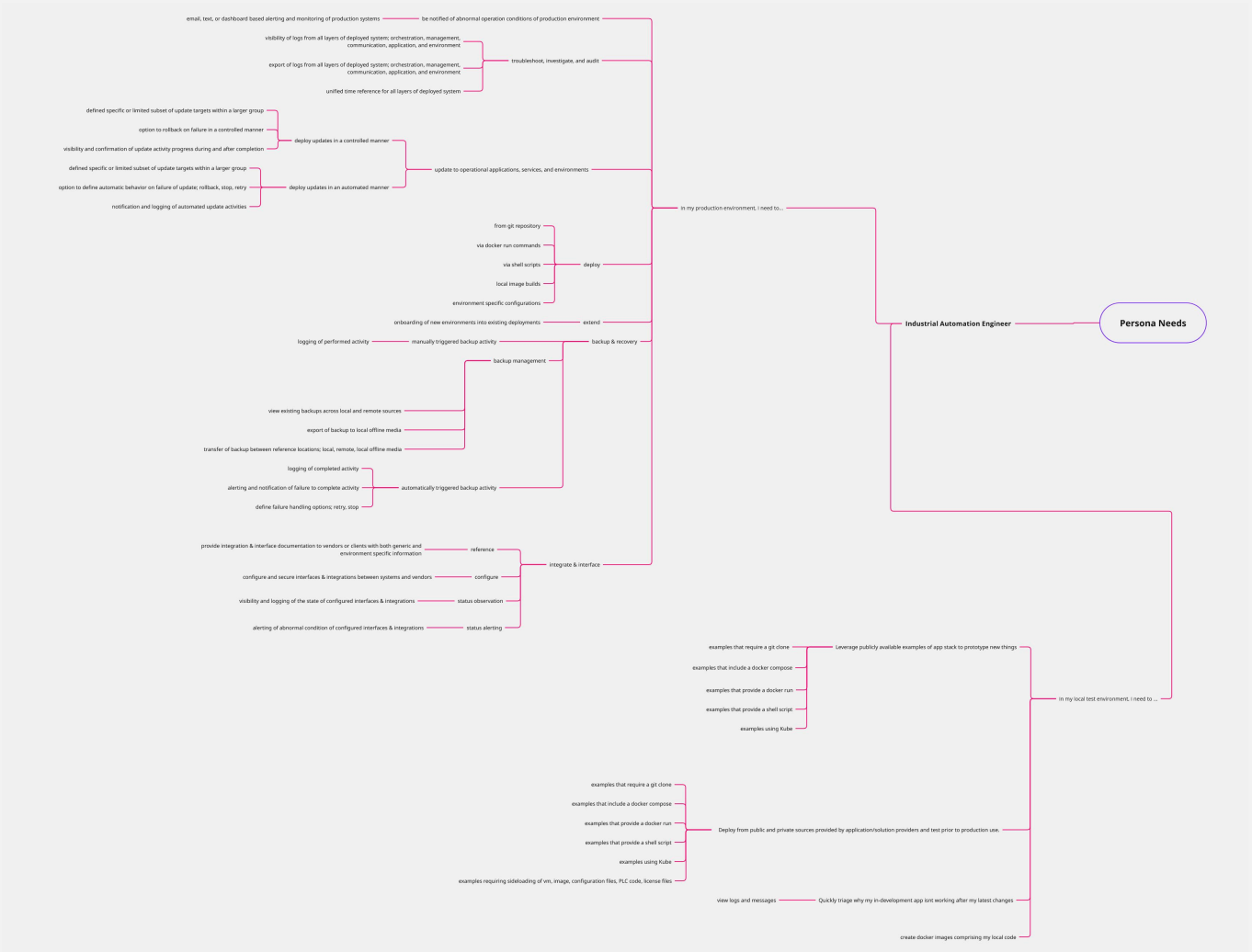
Some are about deploying workloads without internet access. Others are about knowing what's running, when it changed, and whether it's working. Some are about backup and recovery in field-isolated environments. Others are about configuring integrations with SCADA, OPC-UA servers, or message brokers like MQTT.

They need visibility, traceability, and control. But above all, they need tools that work the way they do, not tools that assume they're part of an enterprise DevOps team with access to a fleet of specialists.

This document lays out those needs in the form of narrative user stories, grounded in the real-world tasks OT engineers must perform daily. It's not aspirational. It's operational.

Because if we want to bring the benefits of containerization to industrial environments, we must build platforms that speak the language of those already running the show.

HIERARCHY OF NEEDS: INDUSTRIAL AUTOMATION ENGINEER MIND MAP.



Scan the QR code to download the full res mind map.

USER STORIES: INDUSTRIAL AUTOMATION ENGINEER

Local Testing & Image Creation

As an Industrial Automation Engineer, I want:

- to quickly spin up my in-development applications locally after making changes, so that I can validate that they behave as expected before deploying to a remote production site.
- to create a container image from local code, so that I can test and package application updates in an air-gapped or semi-connected environment.
- to view logs and messages from my application, so that I can troubleshoot early-stage issues before deployment.
- to use example stacks (e.g., Compose files, Dockerfiles, Helm charts) from public sources to prototype solutions, so that I can reduce development effort and adapt existing patterns to OT constraints.

Production Deployment

As an Industrial Automation Engineer, I want:

- to deploy images and workloads provided by my application or solution provider, so that I can run vendor-supported software on my edge devices and gateways.
- to deploy from .tar files, air-gapped image bundles, or raw OCI-compliant artifacts, so that I can work within network-isolated production environments.
- to deploy workloads without needing to pull from public registries, so that I can maintain secure and reliable operations even without internet access.
- to configure deployments using standard artifacts (e.g., Compose files, static YAML), so that I can work with vendor-supplied configurations without modification.

Operational Visibility

As an Industrial Automation Engineer, I want:

- to see running container status, logs, and service activity in a unified operational view, so that I can monitor system health at the edge without jumping between tools.
- visibility into logs from all layers of the deployed system (host OS, containers, orchestration), so that I can investigate failures holistically.
- to display metrics and system statuses on a control-room dashboard, so that operational staff can monitor system performance in real time.
- to be notified when automated update jobs complete, fail, or skip, so that I can verify that scheduled activities have occurred as planned.
- the platform to display clear success/failure messages after system actions complete, so that I can respond appropriately during maintenance windows.

Update & Patch Management

As an Industrial Automation Engineer, I want:

- to define a group of devices to receive an update simultaneously, so that I can roll out patches safely in controlled batches.
- to define specific or limited subsets of a larger deployment for updates, so that I can test changes in a small group before pushing them fleet-wide.
- the option to schedule updates at specific times, so that I can avoid interfering with critical production schedules.
- to trigger updates from a central control point, so that I can manage large-scale environments without manually logging into each device.
- the ability to rollback updates, so that I can recover from a bad deployment with minimal production disruption.

Status Observation, Integration & Interface Management

As an Industrial Automation Engineer, I want:

- to observe the current status of all configured integrations, so that I can validate ongoing connectivity and detect failures early.
- alerts when a system loses connection to its configured integration (e.g. MQTT broker, historian, SCADA), so that I can trigger an investigation before process disruption occurs.
- to see the logs and states of individual integrations (e.g. data pipelines, system bridges), so that I can debug data flow or handoff issues.
- to configure secure interfaces between my edge container platform and upstream enterprise systems, so that I can transmit telemetry, alarms, and batch data reliably.
- to provide interface documentation and reference examples to vendors, so that third-party applications can integrate cleanly with my operational platform.

PORTAINER.iO