IFS Technical Insights

Bi-Weekly Update | Feb 2026

Understanding the IFS Cloud Technical Landscape

From Management Server to Kubernetes Runtime

IFS Cloud is built as an enterprise-grade, cloud-native platform. While this architecture enables scalability, flexibility, and continuous innovation, it also introduces complexity that can make day-to-day operations harder to reason about—especially for teams new to IFS Cloud.

This first article in the IFS Technical Insights series focuses on establishing a shared technical foundation. The intent is not to dive into commands or troubleshooting yet, but to explain how the major building blocks of IFS Cloud fit together and why that understanding matters in real operational environments.

Why Understanding the Architecture Matters

In many IFS Cloud environments, operational issues do not originate where symptoms appear. A slow user experience, failed background job, or intermittent error often stems from a different layer of the platform than where it is observed.

Without a clear mental model of the IFS Cloud technical landscape, teams often end up troubleshooting the wrong component first, leading to longer resolution times and recurring incidents.

Architectural Strategy of IFS Cloud

IFS Cloud is built on a common technology platform designed to embed innovation directly into business processes rather than layering it on top.

Key architectural principles include:

- Cloud-native design using containers and Kubernetes
- API-first architecture using REST and OData v4
- Consistency across SaaS and Remote installations

IFS Technical Insights

Bi-Weekly Update | Feb 2026

Multi-Tier Service-Oriented Architecture

IFS Cloud follows a clear three-tier architecture, with each tier having a well-defined responsibility.

1. Data Storage Tier

The data layer uses Oracle Database as the relational backend. Data access is strictly controlled through business logic APIs, ensuring consistency, security, and integrity.

2. Middle Tier

The middle tier hosts the core IFS services, including OData APIs, identity services, and application logic. These services run as Docker containers orchestrated by Kubernetes.

This layer is where many operational signals originate, such as pod restarts, resource pressure, and scheduling delays.

3. Presentation Tier

IFS Cloud uses a modern HTML5-based web client. All user interfaces consume the same business logic APIs, ensuring consistent behavior across devices.

Lifecycle Experience and the Evergreen Model

IFS Cloud follows an Evergreen delivery model, eliminating traditional upgrade projects. Release Updates (RUs) introduce new functionality, while Service Updates (SUs) deliver fixes and stability improvements.

Tailoring and Extensibility

IFS Cloud separates tailoring from core application logic to ensure that customizations survive updates without rework. This includes configuration, low-code extensions, and external integrations.

Deployment Models

IFS Cloud supports two primary deployment approaches:

- IFS Cloud (SaaS): Fully managed by IFS with standardized operations.
- Remote Installation: Customer-location installations where infrastructure is hosted at a customer-chosen site while lifecycle and platform standards remain consistent.

Why This Foundation Matters

Most recurring issues in IFS Cloud environments are not caused by platform instability, but by missed operational signals or gaps in architectural understanding.

This foundation enables more effective monitoring, faster troubleshooting, and better collaboration between application, database, and infrastructure teams.

Closing Note

This article is part of the IFS Technical Insights series, which focuses on sharing practical, experience-based technical knowledge related to IFS environments.

Subscribe for future updates:

https://ifstechservices.com/ifs-technical-insights