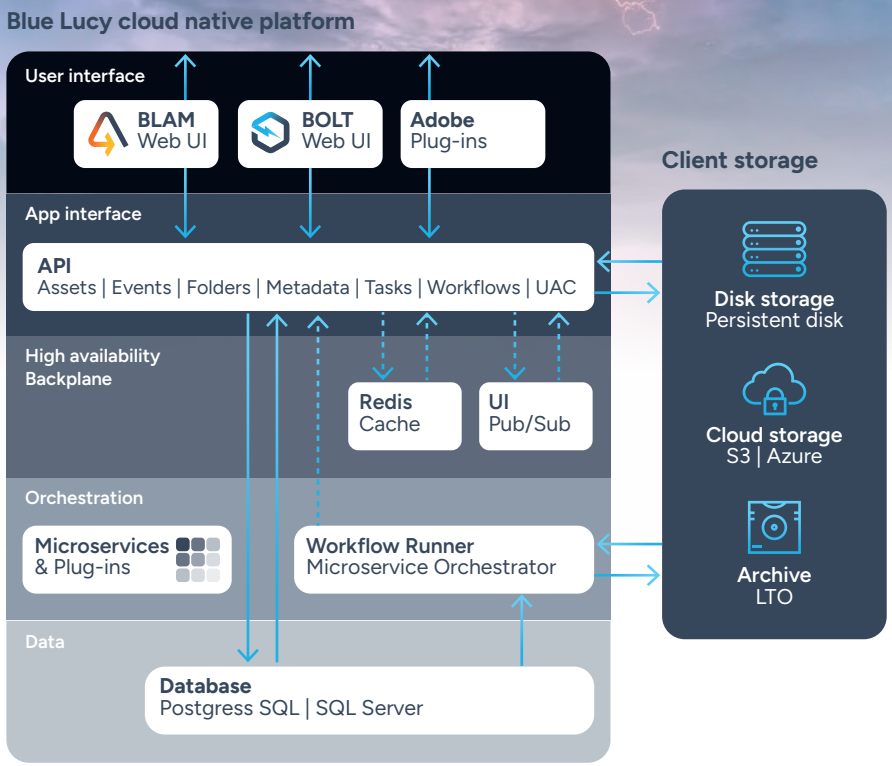
BLUE LUCY

# Technology

## Architecture

The platform follows a distributed microservices architecture, meaning the overall operational capability is structured as a collection of loosely coupled services. This architecture is robust, resilient and conforms to the separation of concerns paradigm. The singularity of purpose which is a key tenet of separation means BLAM is easy to maintain and extend. Equally there are opportunities for re-use of components which speed up our development so that as new business requirements come in – such as a new 3rd party system to connect to – we can implement the capability extremely quickly.




### Overview

The overall architecture comprises the database and two core Blue Lucy components: the API and the Workflow Runer (WFR). It is designed to be horizontally scalable.

The database is the single data repository of the system with the API and WFR being stateless services, meaning that scaling is achieved by simply provisioning more of them. Blue Lucy hosted services are deployed in a Highly Available (HA) configuration comprising two API engines and two WFRs arranged behind a load balancer.

### The API and WFR

The API and WFR are containerised services and will run in any compute infrastructure which means that BLAM is not only agnostic to the cloud environment in which it runs but may also be run 'on prem' in any suitable container orchestrator. This affords maximum operational flexibility including running cloud-ground hybrid systems – an important capability in an industry in which the majority of systems and media is currently located at the facility. 80% of BLAM deployments are cloud-ground hybrid. The distributed architecture also supports global operating models for globally distributed business operations, again we have a referenceable architecture for this.

### Microservices

Within the BLAM orchestration layer there is a further abstraction between the WFR and the microservices which perform specific functions at run time. The microservices are individual components in their own right and separate from the WFR itself. This enables the microservices to be developed independently of the WFR and provides an extra level of safety at run-time. This true microservice architecture gives the Blue Lucy business unparalleled development scale and also means that microservices may be developed by third parties. Microservices may be updated or new ones applied to live systems without any downtime or interruption, the new microservices are simply picked up by the WFR when called.

BLUE LUCY

# Technology

Our microservices run in the WFR and interact with the platform API and the API of the third-party services, directly enabling real time updates between the BLAM orchestrator and subsystems. There are currently around 500 microservices available off the shelf of which approximately 150 are integration connectors to media processing and business systems.

Alongside the microservices the platform also has a range of plugins which are similar in construct and are equally hot pluggable but are designed for the integration of event driven architectures and are deployed as listeners. Examples in use might be a plugin which is subscribed to a message queue listening for specific events, or an HTTP listener to extend the BLAM API.

## Integrate and extend

BLAM has an open REST API, which is the same API used by the platform's user interfaces (UI). The API programmatically is supported with embedded documentation created by Swagger and further documentation hosted at Blue Lucy Central and in the online knowledge portal which may be directly accessed within the applications. In addition, a full Software Development Kit (SDK) is available which allows developers to build microservices for the platform.

Available as a package from NuGet, the developer-friendly .NET SDK enables software engineers to code in their preferred IDE, such as Visual Studio, and provides helper features, such as IntelliSense, facilitating rapid and predictable development. The SDK supports rapid learning, with standard methods for accessing data, and provides a safe interface as the commands interact with the BLAM API rather than lower level. The SDK allows developers to use any .NET-compatible library which provides the freedom to integrate any 3rd party component.

Using the SDK is more powerful than simply calling the API, as it utilises the WFR service to perform any 3rd party function or interaction. This has the potential to extend the useful functionality of the platform way beyond the usual media and broadcast systems to drive more business value for operator. The public SDK which is available at https://www.nuget.org/packages/BLAM3.SDK/ is the same tool that we use internally so it is proven, robust and is regularly updated. An SDK is also available for Python and a handful of our standard microservices use it so the Python runtime environment is included with the WFR.

## Frameworks

The platform is underpinned by .NET 8.0, the latest and fastest .NET framework which affords a truly cross-platform, open source, common language run-time environment. .NET 8.0 affords long term supportability, an excellent security model and is optimised for containerised deployments providing true enterprise level robustness.

For the web-browser based user interface of BLAM we use the Angular 17 framework from Google and for BOLT we use React developed and maintained by Facebook. Both frameworks provide an optimal user experience within the operational use cases. SignalR is used extensively in the user interface to provide real-time status updates. SignalR eliminates the need for polling from the front-end to reduce chatter and provides users with instantaneous operational status updates.

For deployed platform observability we conform to industry standards, including supporting OpenTelemetry, to enable centralised logging, metrics and full tracing. You can even inject your own trace ID to fully integrate with upstream triggers or push the trace ID to downstream recipients to give an incredibly powerful and versatile full operational visibility.

## Platform updates

In development Blue Lucy follows continuous integration principals which drives a modular development approach and ensures robust and reliable operation.

Equally software updates, which are deployed automatically for Blue Lucy managed systems, follows a continuous delivery paradigm with updates to the platform core services available to customer managed environments at all times.

We are currently making major releases, typically pegged to specific new features or platform capabilities, between six and nine times a year. As the microservices are abstracted from the core these may be released to in production systems as required. In all cases updates may be deployed with zero, or near zero downtime.

---

Blue Lucy prefers to focus on the operational business value of our products rather than technology, but the overall architecture and our approach to software development is an important aspect of our value proposition.

If you would like more information about the Blue Lucy software architecture we are always happy to talk.