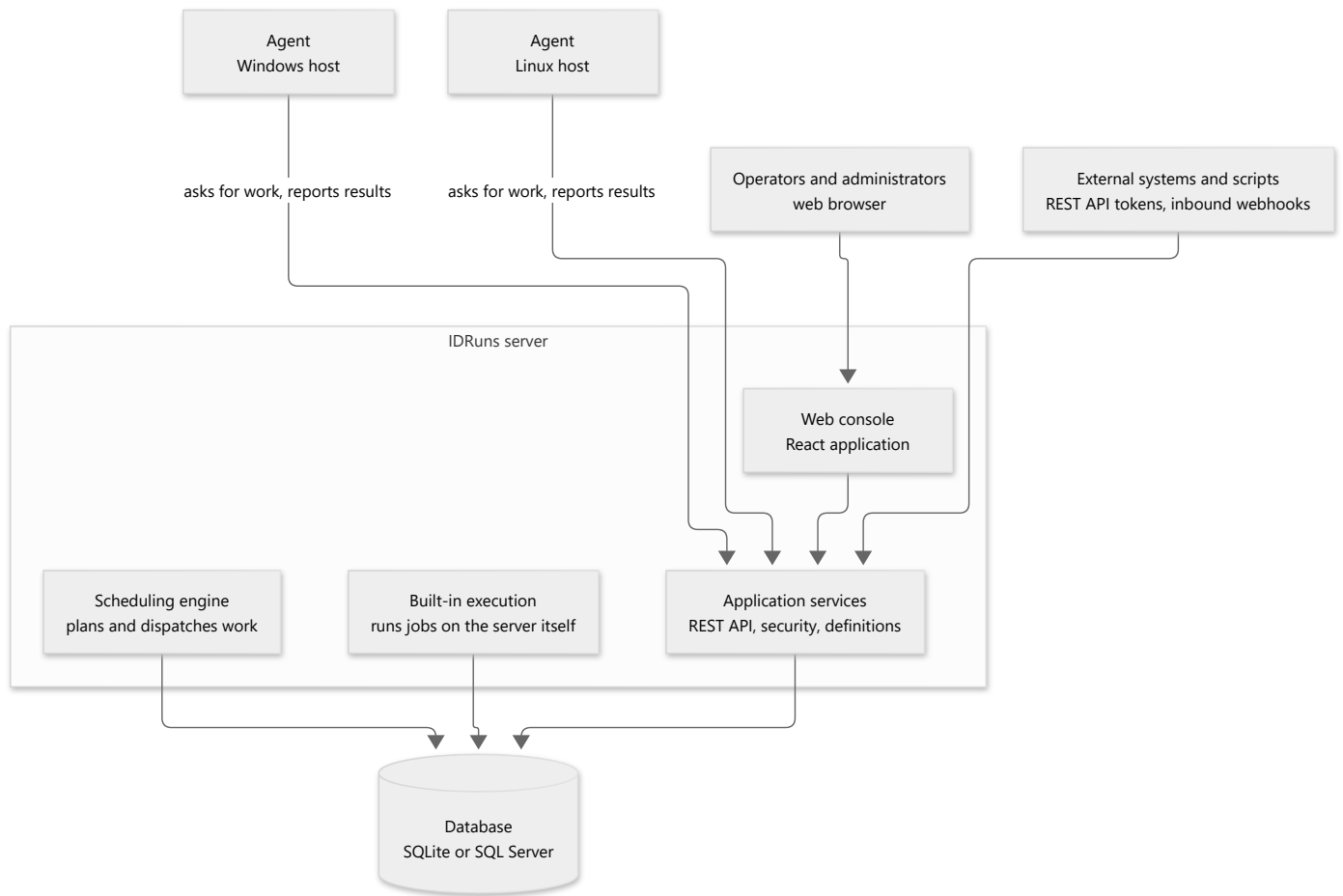


IDRuns — Architecture Overview

IDRuns (formerly NewWave) is a workload-automation platform. It schedules jobs, runs them on the right machines in the right order, and gives operators one console to watch and control everything. This overview covers the shape of the system. The full technical architecture document covers the internals.

The system at a glance



Main building blocks

Component	Role
Web console	Browser application for monitoring runs, defining jobs, and administering the system. Served by the server itself, so there is nothing separate to deploy.
Application services	The REST interface behind the console and behind automation. Handles sign-in, roles, and access to data.
Scheduling engine	Turns job definitions into planned runs, decides when each is eligible, and hands eligible runs to an agent.

Component	Role
Agents	Small services installed on the machines where work must happen. They run the commands and report the outcome.
Built-in execution	The server can run jobs on its own host, so a single-machine install needs no agent.
Orchestration engine	Runs pipelines over existing jobs: conditions, loops, approvals, retries, sub-runs, and values passed between steps. Built into the server; pipelines are versioned and can run on demand or on a cron schedule.
Event engine	Watches for job state changes, files appearing or changing on agents, variable changes, and scheduler conditions (such as a lost agent), and runs the actions or steps attached to each.
File Mover coordinator	Orchestrates multi-step file deliveries between agents (capture, relay, stage, verify, publish, service restarts, recovery).
Database	The single source of truth for definitions, the schedule, run history, and audit records.

How work flows

1. **Define.** Users create jobs and job groups, with calendars, dependencies, and events that trigger follow-up actions.
2. **Plan.** The scheduling engine builds a rolling plan of runs from those definitions.
3. **Release.** When a run's date, time window, and dependencies are met and capacity is available (overall, and in any resource pool the job needs), it becomes ready for an agent. Higher-priority jobs are released first. An approval gate pauses the flow until a person approves it; a branch picks which follow-up job runs from the previous job's exit code.
4. **Execute.** The assigned agent picks the run up, executes it, and streams progress back.
5. **Record.** The server evaluates the result against the job's success criteria, records the outcome, and fires any events, alerts, or follow-up actions. Deadlines (SLAs) raise an alert when a run is at risk or late.

Work can also start from outside: an external system can trigger one job or group through its own webhook, optionally passing variable values for that run. It can start from an event too: a file arriving, changing or disappearing in a watched folder on an agent runs an ordered list of steps (move or copy the file, send it to another agent, run a command, start an orchestration), and system events such as a lost agent connection or a new production day run actions of their own.

Agents in brief

- **Agents call the server; the server never calls the agent.** Agents need no inbound network access, which makes them easy to place behind firewalls.
- Each agent has a name, sends a regular heartbeat, and asks for work on a short interval. A machine that stops reporting is marked offline, and its in-flight runs are flagged so nothing waits forever.
- Agents run on Windows and Linux, can run jobs as a chosen operating-system user, and can serve more than one IDRuns server.

- Agents also watch folders for file events, and let the console browse their own file system when you choose a file or folder path. Agents claim work in batches, so large bursts (thousands of jobs) are dispatched efficiently.
- If the server is unreachable when a job finishes, the agent keeps the result and delivers it later.
- File Mover is built into the agent: with its token set, an agent can take part in file deliveries (send, receive, back up, restart services) with a journal that lets it recover safely after a restart. The console shows which agents are ready for it.

Reliability and scale

- **State lives in the database**, so the server process can restart without losing the schedule or run history.
- **High availability** is optional: two servers share one database, one is active and the other takes over if it stops. Agents fail over automatically.
- **Capacity controls** limit how many jobs run at once, per agent and system-wide. **Resource pools** add named limits (for example "database connections: 4") that jobs and groups draw from, with priority deciding who goes first.
- **Service logs** are rotated and trimmed automatically (age and count), and a crash-looping service restarts at a throttled pace.
- **Licensing** is enforced by pausing scheduling only. Sign-in and viewing remain available.

Security

- Users sign in with local accounts or a directory (LDAP). Roles control what each person can change, and workgroups limit which jobs they can see.
- Scripts and integrations use revocable API tokens. Webhooks use separate tokens that can trigger exactly one job or group and nothing else; they are rate limited and audit logged.
- Agents can be required to authenticate with their own token. Administrators generate agent tokens in the console.
- File Mover transfers can be restricted to folders explicitly granted to a workgroup, with runtime users granted by name (an administrator setting).
- Connection secrets for external systems are stored encrypted.
- Changes and run status transitions are written to an audit log.

Extensibility and integrations

- **Job types:** operating-system commands, file transfer, plugin jobs, and control nodes (terminator, approval gate, branch).
- **Orchestrations:** visual or YAML pipelines over existing jobs, with a library of nearly 30 built-in file, network and service steps that run on Windows and Linux agents.

- **Events:** job, file, variable and system events run reusable actions (email, SMS, alert, log, set a variable, add a job to the schedule).
- **File Mover** plugin for copying or moving files of up to 10 GiB between agents, with deployment steps and automatic recovery.
- **AI assistance** (optional): describe a job in plain English to get a proposed definition, and get a short AI diagnosis attached to failure alerts.
- **Inbound webhooks** start a job or group from any system that can make an HTTP request.
- **Plugins** connect IDRuns to external systems (for example a SQL database or a REST service) through a common interface, without changes to the scheduler. The JDBC plugin reaches any database with a JDBC driver: administrators upload driver jars, and each run launches a small Java process that receives the request (including the password) on standard input and returns the result as JSON, so cancelling is just stopping the process and a faulty driver cannot affect the server. It needs a JDK on the server and runs in-process only.
- **Migration:** built-in import from, and export to, Tidal Enterprise Scheduler, plus import and export of definitions between IDRuns systems (by file or a live connection). Agents, agent lists, runtime users and workgroups are matched by name; named mapping sets (stored on the server and exchangeable as small files) translate names that differ, and an import review makes every unresolved name an explicit choice. Only names cross systems - never ids or passwords.

Technology

Area	Choice
Server	Python, FastAPI
Web console	React, TypeScript
Database	SQLite for evaluation and small installs; SQL Server for production and high availability
Agent	Python; Windows service or Linux systemd service
Hosting	One server process on Windows or Linux, one database, agents on the target machines
Standalone edition	A single executable (Windows or Linux) containing the server, web console, and database migrations, with a first-run setup wizard and a 30-day evaluation