

IDRuns — Product Overview

Enterprise job scheduling without the enterprise price tag.

IDRuns is a workload-automation platform. It schedules jobs, runs them on the right machines in the right order, and gives operators one web console to watch and control everything: scripts, commands, file transfers, database tasks and API calls across Windows and Linux servers.

Teams adopt IDRuns as their first real scheduler, to replace cron sprawl, Windows Task Scheduler and home-grown wrapper scripts, or as a lower-cost replacement for Tidal Workload Automation (Cisco Tidal / Tidal Enterprise Scheduler). IDRuns was previously called NewWave; the engine is the same.

Why IDRuns

One place for all scheduled work	Every job, dependency, calendar and run history lives in one system with one console. No more hunting through crontabs and task schedulers on dozens of hosts.
Small footprint	One server process, one database, and a small agent on each machine that runs work. The web console is served by the server, so there is nothing else to install.
Minutes to first job	The standalone edition is a single executable with a setup wizard. No installer and no multi-day deployment project.
Firewall-friendly agents	Agents call the server; the server never calls the agent. Agents need outbound HTTP(S) only.
Assisted migration from Tidal	Our services team has tooling that speeds up moving your Tidal workload onto IDRuns.
Flat, predictable pricing	Every paid plan includes five agents. A free 30-day evaluation is available.

Feature overview

Define work

- **Jobs and nested job groups.** Groups pass agent, calendar, timezone, time window, variables, owner and runtime settings down to the jobs inside them. Any job can override what it inherits.
- **Job types.** Operating-system commands, FTP transfers, plugin jobs (SQL Server, REST API and others), File Mover deliveries, and control nodes: Terminator, Approval Gate and Branch.
- **Calendars and fiscal calendars.** Reusable run-date rules, business periods, effective-date ranges, offsets and time windows.

- **Dependencies, events and actions.** Prerequisites between jobs, triggers that fire on run outcomes, and reusable actions such as alerts and follow-up jobs.
- **Variables.** Global, group and local variables can be inserted into commands and other fields.
- **Job iteration.** One definition can fan out across a list of values.
- **Workflow Designer.** A visual canvas for building and reviewing dependencies, with critical-path highlighting and an estimated duration.
- **Version history.** Save named versions of a whole job group, compare them field by field, preview a restore, and roll back safely.
- **Describe your job in plain English** (optional AI). Type "Run backup.sh every weekday at 2am on DB01" and get a proposed definition to review before saving.

Control execution

- **Approval gates.** Pause a workflow until a named approver or workgroup approves or rejects it.
- **Branching.** Choose the next job from the previous job's exit code (for example " $= 0$ ", "between 1 and 4", " ≥ 8 ", or otherwise).
- **Resource pools and priority.** Named limits such as "database connections: 4" that jobs and groups draw from. Higher-priority work is admitted first.
- **Capacity limits.** Per-agent and system-wide caps on concurrent jobs.
- **SLA deadlines.** "Finish within N minutes" or "finish by 06:00". A warning is raised at 80% of the window and a critical alert when the deadline is missed.
- **Reruns, timeouts and success criteria.** Decide what counts as success, how long a job may run, and how it retries.
- **Run as a chosen user.** Jobs run under a configured operating-system identity on Windows and Linux.

Monitor and operate

- **Job Activity.** List, Flow and Sequence views of every run for a production date, with bulk actions, live output and a full run history.
- **Clear waiting reasons.** A waiting run says exactly why: scheduled start, time window, resources, dependencies, group, or approval.
- **Sequence check.** Flags runs that started out of dependency order, so overrides and unexpected reruns are easy to spot.
- **Alerts, logs and reports.** Acknowledge and close alerts, search diagnostic logs, and review workload trends by hour, status and job group.
- **AI failure diagnosis** (optional). Failure alerts can carry a short explanation of the likely cause.
- **Master Status and top-bar counters** show scheduler health and progress at a glance.
- **Schedule Simulator.** Advance virtual time over a snapshot of your definitions to see what would run, without touching production.
- **Works on any screen.** Three navigation layouts, a choice of color schemes, and a dedicated phone layout.

Move files reliably — File Mover

- Copy or move files of up to 10 GiB between agents.
- Multi-step deliveries: capture, relay, stage, verify, publish, back up and restart services.
- A journal on each agent lets an interrupted delivery recover safely after a restart.
- Path access is granted per workgroup and checked both when a job is defined and when it runs.

Integrate

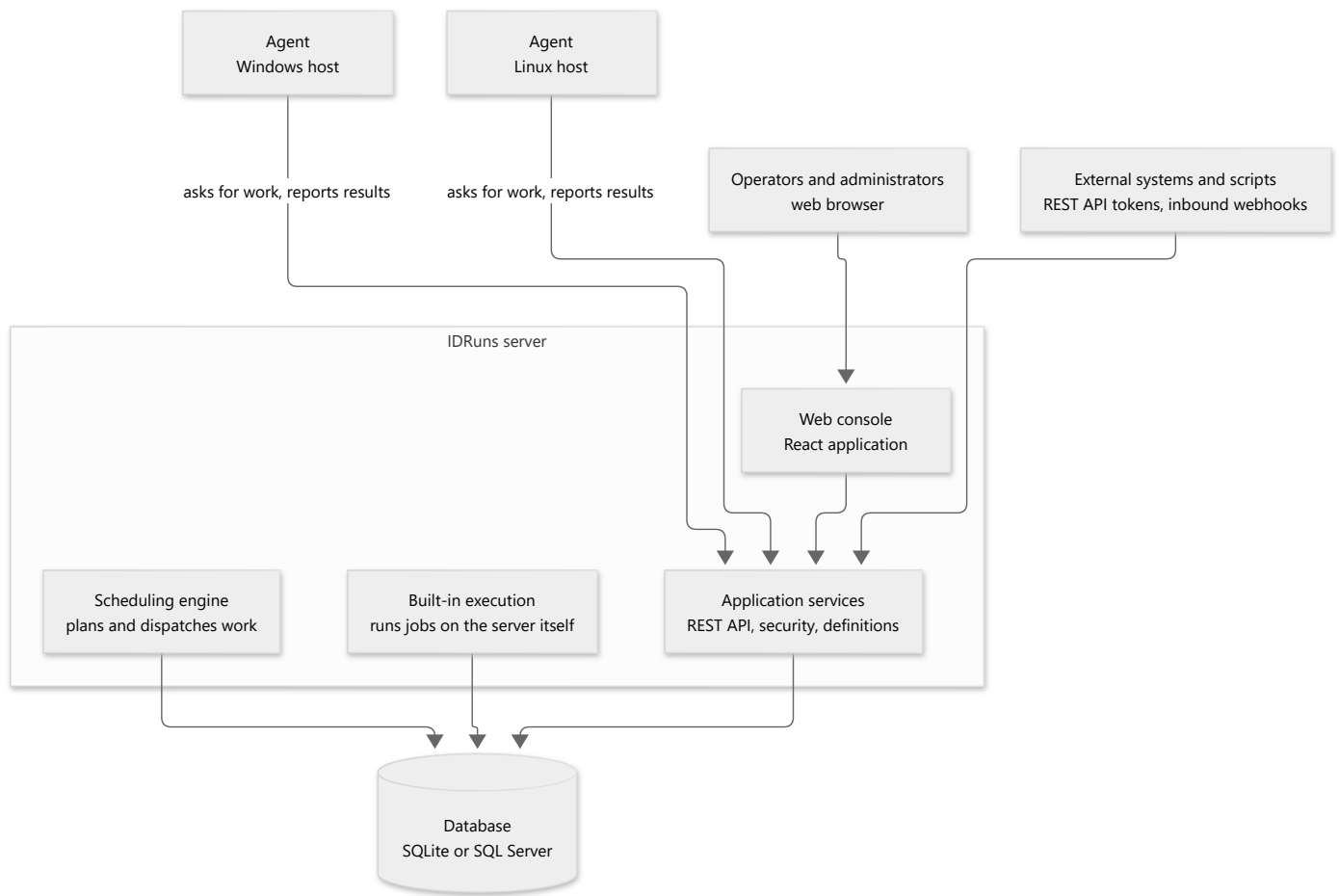
- **REST API** for everything the console does, using revocable API tokens.
- **Inbound webhooks.** Any system that can make an HTTP request can start one job or group, optionally passing variable values for that run. Each webhook token can trigger exactly one target and is rate-limited and audit-logged.
- **Plugins.** Named, tested connections to external systems such as SQL Server and REST services, through a common interface. Plugin jobs run on the server or on an agent.
- **Export and import.** Move definitions between IDRuns systems, for example from test to production.

Govern and secure

- Sign in with local accounts or LDAP / Active Directory.
 - Role-based access, with workgroups controlling who can see and change which jobs.
 - Field-level audit log of changes and run status transitions.
 - Connection secrets stored encrypted.
-

How IDRuns is built

Architecture



Component	Role
Web console	Browser application for monitoring, defining jobs and administration. Served by the server on a single port.
Application services	The REST interface behind the console and behind automation. Handles sign-in, roles and data access.
Scheduling engine	Turns definitions into a rolling plan of runs, decides when each is eligible, and hands it to an agent.
Built-in execution	The server can run jobs on its own host, so a single-machine install needs no agent.
Agents	Small services on the machines where work must happen. They run the work and report the outcome.
File Mover coordinator	Orchestrates multi-step file deliveries between agents.
Database	The single source of truth for definitions, schedule, run history and audit records.

How work flows

1. **Define.** Users create jobs and groups with calendars, dependencies and events.
2. **Plan.** The engine compiles a rolling schedule of runs from those definitions.
3. **Release.** When a run's date, time window and dependencies are met and capacity is available, it becomes ready. Higher priority goes first.
4. **Execute.** The assigned agent picks the run up, executes it and streams progress back every few seconds.
5. **Record.** The server checks the result against the job's success criteria, records the outcome, and fires events, alerts and follow-up work.

Technology stack

Layer	Technology
Server	Python 3.11+, FastAPI, uvicorn
Persistence	SQLAlchemy 2 ORM with versioned Alembic migrations
Database	SQLite (evaluation and single-node) or Microsoft SQL Server (production and high availability)
Web console	React 19, TypeScript, Vite, TanStack Query / Table / Virtual, React Flow
Agent	Python 3.11+, asyncio, httpx; Windows service or Linux systemd service
Licensing	Ed25519-signed license files
AI (optional)	Anthropic Claude, with your own API key
Standalone edition	One self-contained executable for Windows or Linux

Agents and the agent protocol

- All traffic is JSON over HTTP(S), started by the agent. Agents need no inbound ports, which makes them easy to place behind firewalls and in DMZs.
- An agent sends a heartbeat every 30 seconds and asks for work every 5 seconds when idle. A machine that stops reporting is marked offline, and its in-flight runs are flagged so nothing waits forever.
- If the server is unreachable when a job finishes, the agent keeps the result and delivers it later.
- One agent can serve more than one IDRuns server, and agent lists spread work and provide failover across machines.
- Agents can be required to authenticate with their own token, generated in the console.

Reliability and high availability

- **All state lives in the database,** so the server can restart without losing the schedule or run history.
- **Active/standby high availability.** Two or more servers share one SQL Server database. A database lease decides which one schedules. If it stops, the standby takes over within about 30 seconds, and agents fail over to it on their own.
- **Self-healing loops.** Background scheduling loops record failures and back off, and a slow task cannot stall the console or agent traffic.

- **Safety nets every tick.** Stalled runs are reaped, maximum runtimes are enforced, and cancellations are delivered to agents.
- **Housekeeping.** Run history, audit and service logs are purged and rotated automatically on configurable retention.
- **License enforcement pauses scheduling only.** Sign-in and viewing always remain available.

Security

Concern	How IDRuns handles it
Sign-in	Local accounts (PBKDF2-HMAC-SHA256, 600,000 iterations, per-user salt) or LDAP; signed session cookie with a short default lifetime
Authorization	Roles plus workgroup ownership of definitions
API access	Revocable bearer tokens; only a hash is stored, and the token is shown once
Webhooks	Per-endpoint tokens limited to one target; rate limits, body-size cap, uniform responses on failure, audit logged
Agents	Bearer-token authentication with constant-time comparison; always required for File Mover
File Mover	Workgroup path grants checked at authoring and at dispatch, and again on the agent; symlinks and junctions refused
Linux run-as	A small, separate helper holds the privilege to switch users; the server itself stays unprivileged
Secrets	Connection secrets encrypted at rest
Audit	Field-level audit trail of changes and run status transitions

We recommend running the agent link over HTTPS and turning on agent authentication once every agent has its token.

Deployment options

Option	Best for	What you install
Standalone edition	Evaluation, small teams, single server	One executable. A first-run wizard sets the admin account, listen address and database, then opens the console in your browser. It can register itself as a service.
Server install	Production	The server as a Windows service or on Linux, backed by SQL Server, with agents on target hosts
High availability	Business-critical scheduling	Two or more servers sharing one SQL Server database, with automatic failover

Requirements at a glance

- Server: Windows or Linux.
- Database: SQLite built in, or Microsoft SQL Server.
- Agents: Windows or Linux, with outbound HTTP(S) to the server.
- Clients: any modern browser, including phones and tablets.

Migrating from Tidal

Moving from Tidal Enterprise Scheduler is a services engagement. Our services team has purpose-built tooling that speeds up the conversion of Tidal job groups, calendars, agent lists, dependencies and variables, and works with your team through a phased migration:

1. Run IDRuns alongside your existing Tidal environment.
2. Our services team converts your workload one job group at a time.
3. Together we verify each group in IDRuns, then retire it in Tidal.
4. Cut the Tidal license once the last group has moved.

Because Tidal and IDRuns share the same core concepts, most definitions carry across with little rework.

Getting started

- **Try the live demo.** It runs the real application in your browser.
- **Download the standalone edition** for a free 30-day non-production evaluation. No installer and no admin rights on Windows.
- **Talk to us** for a quote. Every paid plan includes five agents.

idruns.com