Timebars Ltd
Agile PM | Classic PM | PPM

Customer Installation Options

The Timebars suite is a production-ready project management platform covering three disciplines that every project-driven organization needs. All three applications are deployed from a single codebase and are runtime-configurable — one deployment supports all three products simultaneously.

Application	Discipline	Core Capability
Agilebars	Agile Sprint Planning	Backlog management, burndown charts, sprint scheduling, spreadsheet sync
Timebars	Resource Scheduling	Drag-and-drop resource allocation, multi-project scheduling, spreadsheet sync
Costbars	Project Portfolio Management	Business driver prioritization, portfolio analytics, project pipeline scheduling

This document outlines the two deployment options available. The key distinction is how much of the build and delivery pipeline you take ownership of.

The Two Options at a Glance

	Option 1: Managed Container Deployment	**Option 2: Own the Code**
What Timebars provides	Pre-built, versioned Docker images	Full source code + deployment pipeline
Customer responsibility	Infrastructure, deployment, operations	Development, build pipeline, infrastructure, operations
Customization	Configuration only	Unlimited — full code access
Updates	Minor fixes and stability improvements	Self-managed on your own schedule
Branding	Standard Timebars branding	Full white-label capability
Data location	Your infrastructure	Your infrastructure
Team required	DevOps / SysAdmin	Developer + DevOps + DBA

	Option 1: Managed Container Deployment	Option 2: Own the Code
Time to first deployment	Hours	Days to weeks

Air Gap and Vendor Independence

Both options are suitable for organizations operating in air-gapped environments — those with no internet access, restricted network perimeters, or strict policies against third-party vendor dependencies. Once the Docker images or source code have been delivered and your environment is configured, the Timebars suite runs entirely on your own infrastructure with no ongoing external connections required. There is no phone-home licensing mechanism, no cloud dependency, and no vendor system in the data path. Organizations with data residency obligations, government or defence network constraints, or internal policies requiring full infrastructure independence can deploy either option with confidence.

Option 1: Managed Container Deployment

Overview

Timebars Ltd. builds, tests, and publishes pre-built Docker images to Docker Hub. Your team pulls those images, deploys them to your infrastructure, and manages the running environment. The application code is maintained by Timebars; your team owns the infrastructure, configuration, and operations.

This is the right choice for organizations that need Timebars running on their own infrastructure quickly, without allocating a development team to the project.

What Timebars Provides

Upon purchase, Timebars Ltd. delivers:

- Docker Hub credentials (username/password or access token) to pull the official images
- Docker Compose configuration files for the complete stack
- Environment configuration templates
- Initial setup and configuration documentation
- Ongoing image updates covering minor fixes and stability improvements; these updates do not include new product features
- Custom feature development is available under a separate professional services engagement at an agreed fee — contact us to discuss your requirements

The Five Containers

The complete Timebars solution runs as five coordinated containers:

Container	Purpose
jimecox807/tbrun	The Timebars/Agilebars/Costbars client application
jimecox807/tbwwwp	Sales and support website with Cloud Dashboard

Container	Purpose
jimecox807/tbbe	Strapi backend services
jimecox807/tbfm	File management and CMS *(optional)*
jimecox807/tbcdn	CDN services for media delivery

System Requirements

Hardware (Minimum for Production)

- 4 CPU cores
- 16 GB RAM
- 100 GB SSD storage
- Stable internet connection for initial image download

Software

- Docker Engine 20.10 or higher
- Docker Compose 2.0 or higher
- Linux (Ubuntu 20.04+, CentOS 8+, or RHEL 8+), Windows Server 2019+, or macOS 11+
- PostgreSQL 13+ (deployable as a separate container or managed database service)

Network

- Open ports: 80 (HTTP), 443 (HTTPS), and any additional ports as configured
- Domain name(s) configured with DNS
- SSL/TLS certificates (Let's Encrypt is recommended for straightforward setup)

Team & Skills Required

Initial Deployment — 1 to 2 people, 4 to 8 hours

A **DevOps Engineer or System Administrator** with the following skills can handle initial deployment:

- Docker and Docker Compose — pulling images, running containers, managing volumes
- Networking basics — DNS configuration, SSL/TLS, port routing
- Linux or Windows Server administration
- Reverse proxy setup (Nginx, Apache, or Traefik)
- Environment variable configuration

A **Database Administrator** is optional but recommended for production environments, covering PostgreSQL setup, backup procedures, and initial performance configuration (2 to 4 hours).

Ongoing Operations — Part-time commitment

Role	Responsibilities	Typical Weekly Commitment
System Administrator	Container health monitoring, log review, image updates, backup management, security patch coordination	2 to 5 hours

Role	Responsibilities	Typical Weekly Commitment
Database Administrator	Database backups, maintenance windows, performance checks, data integrity	1 to 3 hours

Deployment Steps

1. Authenticate to Docker Hub using provided credentials
2. Pull all five Docker images
3. Configure environment variables and application secrets
4. Set up the PostgreSQL database
5. Deploy the full stack using the provided Docker Compose configuration
6. Configure reverse proxy routing and SSL certificate installation
7. Verify all services are running and communicating correctly
8. Perform initial application configuration and user setup
9. Establish monitoring, alerting, and scheduled backup procedures

Key Considerations

- Application code and feature releases are managed by Timebars Ltd.
- Updates are applied by pulling new image versions and restarting the affected containers
- Customization is limited to configuration — the underlying application code is not accessible
- Requires ongoing Docker Hub access to receive updates

Option 2: Own the Code

Overview

The Own the Code option delivers the complete Timebars source code across six repositories, along with a documented CI/CD deployment pipeline. Your development team takes full ownership: they build the applications, manage the deployment pipeline, and operate the production environment.

This option is designed for organizations that need the ability to modify any part of the product — the scheduling logic, the data models, the UI, the integrations, or the reporting layer — and who want to operate entirely independently of Timebars Ltd.'s product roadmap.

What Timebars Ltd. Provides

Upon purchase, Timebars Ltd. delivers:

- Bitbucket repository credentials and access to all six source code repositories
- Docker Compose configuration files and deployment scripts (included in the dockeretc repository)
- Environment configuration templates for all application components
- Initial setup and configuration documentation
- CI/CD pipeline templates for GitHub Actions or Bitbucket Pipelines
- Ongoing repository updates covering minor fixes and stability improvements; new product features are not included in the standard acquisition

- Custom feature development is available under a separate professional services engagement at an agreed fee — contact us to discuss your requirements

The Six Source Code Repositories

Repository	Technology	Purpose
tbrunp	Vanilla JS, Parcel.js	Timebars/Agilebars/Costbars client applications
tbwwwp2	Next.js 14	Sales website, Notifications System and Dashboards
tbbe	Strapi v4	CMS and backend API services, PostgreSQL
cdn	PNG, MP4	Static content delivery *(optional)*
filemanager	Node.js	File upload component for CDN integration *(optional)*
dockeretc	MD, Bash	Docker Compose files, deployment scripts, and technical documentation

The Deployment Pipeline

The included deployment architecture is organized across three zones. This is a standard, cloud-agnostic pattern that any experienced developer will recognize immediately.

Zone 1 — Development Environment

Your development team works with the source code. They use their chosen editor and AI coding tools (see below) to build custom features, modify workflows, and adapt the applications to your specific requirements. Changes are committed to your own Git repository (GitHub, GitLab, or Bitbucket) and validated against a local staging environment before anything advances.

Zone 2 — CI Pipeline to Docker Hub

Every push to the repository triggers your CI pipeline — GitHub Actions or Bitbucket Pipelines, depending on your toolchain. The pipeline builds the Docker images, runs automated linting and tests, and pushes the tagged, versioned images to Docker Hub. This is the controlled handoff between development and production. Nothing reaches the production server without passing the pipeline.

Zone 3 — Production Environment

The production server pulls the current image from Docker Hub and runs the applications as separate containers. A reverse proxy (Nginx or Traefik) handles routing and SSL termination. A PostgreSQL database persists application data. The entire stack runs on whatever cloud provider or VPS your organization prefers — AWS, GCP, Azure, Hetzner, DigitalOcean, or on-premises bare metal. There are no proprietary dependencies and no vendor-controlled infrastructure in the path.

AI-Accelerated Development

The codebase is structured and documented to be navigable by AI coding tools, including Claude Code. This materially changes the economics of customization.

A development team working with Claude Code can:

- Orient to an unfamiliar codebase significantly faster than with traditional code review
- Propose and implement targeted changes to scheduling logic, UI, or data models in days rather than weeks
- Generate and maintain tests alongside feature development
- Accelerate integration work with internal systems and APIs

The practical result is that a small, capable team can have a working customized version of the product in days to a few weeks, depending on the depth of modification required — without needing deep expertise in every layer of the stack before they start.

System Requirements

Hardware (Minimum for Production)

- 4 CPU cores
- 16 GB RAM
- 150 GB SSD storage (additional space for source code, build artifacts, and build tooling)
- Stable internet connection

Software

- Node.js 18.x or higher
- NPM 9.x or higher
- Git 2.30 or higher
- Docker Engine 20.10+ and Docker Compose 2.0+
- PostgreSQL 13+
- Linux (Ubuntu 20.04+, CentOS 8+, or RHEL 8+), Windows Server 2019+, or macOS 11+

Build Tools (included in repositories)

- Parcel.js — for the tbrunp client applications
- Next.js build tooling — for the tbwwwp2 dashboard
- Strapi build dependencies — for the tbbe backend

Team & Skills Required

Initial Deployment — 2 to 4 people, 24 to 48 hours

Role	Required Skills	Estimated Effort
Full-Stack Developer	JavaScript/Node.js, Next.js, Strapi, jQuery, NPM, Parcel.js, Git	16 to 32 hours
DevOps Engineer	Server provisioning, CI/CD pipeline configuration, reverse proxy, SSL, Docker, environment management	12 to 24 hours
Database Administrator	PostgreSQL setup and configuration, schema management, backup and recovery, performance tuning	8 to 16 hours

Role	Required Skills	Estimated Effort
QA Specialist *(recommended)*	Web application testing, API testing, integration testing, browser compatibility	16 to 24 hours

Ongoing Operations

Role	Responsibilities	Typical Weekly Commitment
Full-Stack Developer	Custom feature development, bug resolution, dependency updates, integrations, build maintenance	10 to 40 hours (scales with customization workload)
DevOps Engineer	CI/CD pipeline maintenance, infrastructure monitoring, security updates, scaling, disaster recovery	5 to 15 hours
Database Administrator	Backup verification, schema migrations, performance monitoring, data integrity	3 to 8 hours

Deployment Steps

1. Receive Bitbucket repository credentials and clone all six repositories
2. Install Node.js, NPM, and all project dependencies
3. Configure environment variables for each application component
4. Set up and configure PostgreSQL
5. Build each component:
    - Build **tbrunp** using Parcel.js
    - Build **tbwwwp2** using Next.js build commands
    - Build the **tbbe** Strapi backend
    - Configure the **cdn** static site *(optional)*
    - Configure **filemanager** *(optional)*
6. Deploy built applications to containers or directly to web servers
7. Configure reverse proxy routing and SSL certificates
8. Set up your CI/CD pipeline for automated builds and deployments
9. Conduct comprehensive application testing across all three products
10. Configure monitoring, logging, alerting, and backup systems

Key Benefits

Complete Control Every aspect of the product is modifiable — data models, scheduling logic, UI, reporting, integrations. There are no vendor-controlled feature flags, no API rate limits, and no roadmap dependency.

One-Time Acquisition Cost No per-seat fees. No subscriptions that grow with headcount. The acquisition is a capital investment; ongoing cost is your own development team's time.

Data Sovereignty The application runs entirely on your infrastructure. Data never flows through a third-party SaaS platform. This is relevant for organizations with compliance obligations, data residency

requirements, or internal policy constraints on cloud data storage.

White-Label and Rebrand Ready The codebase can be rebranded for internal deployment, client-facing use, or integration into a broader product offering. The licence does not restrict this.

No Vendor Sunset Risk When you own the source code, the product cannot be discontinued beneath you. This is not a theoretical concern — Microsoft retired Project Server on-premises support and is sunsetting Project Online, forcing thousands of organizations into expensive, disruptive migrations to platforms they did not choose, on timelines they did not control. Once you acquire the Timebars codebase, that scenario is permanently off the table. Pricing changes, acquisitions, or discontinuation decisions at the vendor level have zero impact on your deployment. The product exists independently of Timebars Ltd.'s ongoing business decisions and will continue running as long as your team chooses to run it.

Built on Technologies That Will Not Be Deprecated The Timebars suite is built on the open web stack — JavaScript, HTML, and CSS running in the browser. These are not proprietary runtimes, vendor-controlled frameworks, or platform-specific technologies. JavaScript is the most widely used programming language in the world, browsers are a permanent fixture of computing infrastructure, and the standards underpinning them are governed by open industry bodies. There is no realistic scenario in which these technologies are deprecated or discontinued. Your investment in this codebase is not dependent on the survival of a particular vendor, framework, or cloud platform.

PostgreSQL: An Enterprise Database With No Licensing Risk The application uses PostgreSQL as its production database — an open-source relational database with over 35 years of active development, originating from a UC Berkeley research project. PostgreSQL is governed by an independent community and the PostgreSQL Community Association, a non-profit body — no single company owns it, and no vendor can change its licensing terms or discontinue it. It consistently ranks among the top four databases globally and has been named the most admired and desired database by developers in the Stack Overflow Developer Survey for multiple consecutive years. It is used in production by Apple, Instagram, Spotify, and thousands of enterprises worldwide. There are no per-core fees, no enterprise edition paywalls, and a large, experienced global talent pool available to support it.

A Proven Foundation The scheduling engine, data structures, and spreadsheet integration are production-tested. Your team is customizing a working product, not speculating about whether a complex scheduling system can be built.

Key Considerations

- Your team is responsible for all application updates, security patches, and dependency maintenance
- Build expertise across the full stack (Node.js, Next.js, Strapi, Docker) is required
- Initial setup is more involved than Option 1 — budget time accordingly for the CI/CD pipeline and testing phases
- The depth of customization required directly determines the timeline; straightforward deployments move faster than those requiring significant workflow modifications

---

Support and Documentation

Both options include:

- Comprehensive deployment documentation

- Configuration templates and environment examples
- Technical support during initial setup
- Access to knowledge base resources
- Migration assistance between options (if your requirements change)

For technical questions, onboarding support, or to discuss your specific deployment requirements:

Jim Cox Timebars Ltd. **Phone:** (613) 255-5374 **Email:** jcox@tbcox.com

---

Document Version: 1.2 **Last Updated:** May 2026