Administration

Background Job Processing

VeloxFactory processes background work — thumbnail generation, nightly purge routines, print task cleanup — through a Redis-backed queue managed by [Laravel Horizon](). Horizon runs as a single supervised process and manages its own worker pool internally. It replaces the simple `queue:work` approach with dynamic scaling, real-time monitoring, and a built-in dashboard.

Prerequisites

Horizon has two hard requirements beyond the base VeloxFactory stack: a running Redis instance and the `php-redis` PHP extension. Neither is optional — Horizon will refuse to start without both.

Redis

Redis can be installed natively or run as a Docker container. Both work equally well; the choice depends on your infrastructure preferences.

Native installation:

```
apt install redis-server
systemctl enable redis-server
systemctl start redis-server
```

Docker container:

```
docker run -d \
  --name
  --restart
  -p
  redis:alpine
```

The container binds to `127.0.0.1` only — Redis is not exposed to the network, which is the correct default for a single-server deployment.

Whichever method you choose, verify connectivity before proceeding:

```
redis-cli                                                    ping
#                                  Expected:                 PONG
```

PHP Redis Extension

VeloxFactory is configured to use `phpredis` as the Redis client. The extension must be installed and active for PHP:

```
apt                                  install                                  php-redis
systemctl   restart   php8.2-fpm       #   adjust   version   to   match   your   PHP   installat
```

Confirm the extension is loaded:

```
php                  -m                  |                  grep                  redis
#                                  Expected:                                     redis
```

.env Configuration

With Redis running and the extension installed, update your `.env` to activate Redis as the queue driver:

```
QUEUE_CONNECTION=redis

REDIS_CLIENT=phpredis
REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379
```

Web Server

VeloxFactory requires a web server that routes all requests through Laravel's `public/index.php` entry point. Both Apache2 and nginx are supported. The Horizon dashboard at `/horizon` and the Reverb WebSocket endpoint require no special routing rules — they are handled by Laravel and PHP-FPM like any other request, with one exception: Reverb needs a WebSocket proxy pass.

Apache2

Enable `mod_rewrite` before configuring the vhost:

```
a2enmod rewrite proxy proxy_http proxy_wstunnel
systemctl restart apache2
```

Virtual host configuration:

```
<VirtualHost *:80>

    </Directory>

    #

    CustomLog ${APA
</VirtualHost>
```

Laravel's bundled `.htaccess` in `public/` handles the rewrite rules — no additional configuration needed for URL routing.

nginx

```
server {
```

```
    }





                                                                                                       f




    }


                                                                                                     #








    }





                                                                                           location


    }
}
```

ℹ **Adjust the PHP-FPM socket path to match your PHP version.** On systems with multiple PHP versions installed, the socket is typically at `/var/run/php/php8.2-fpm.sock`. Verify with `ls /var/run/php/`.

Queues

VeloxFactory uses two queues with distinct priorities:

Queue	Jobs	Notes

`high`	Thumbnail generation	Dispatched on-demand when a report is rendered. Processed with highest priority — workers on this queue are never blocked by maintenance routines.
`default`	Nightly purge jobs	Scheduled automatically at midnight. Cleans up expired Report History Records, orphaned files, and completed Print Tasks according to the configured retention policies.

> **i The two queues run in separate worker pools.** A long-running purge job on the `default` queue cannot delay thumbnail generation on the `high` queue — they never compete for the same worker.

Horizon Supervisors

Horizon manages workers through internal supervisors — process groups, each responsible for one queue. The system-level Supervisor (Supervisord) only ever manages the single Horizon master process; Horizon itself handles everything below that.

Supervisor	Queue	Balancing	Notes
`supervisor-rendering`	`high`	Auto	Scales worker processes dynamically based on queue depth — up to 10 in production. Job timeout: 120 seconds.
`supervisor-default`	`default`	Simple	Fixed worker count. Runs with lower CPU priority (`nice 10`) — purge jobs are maintenance work and should not compete with rendering for system resources. Job timeout: 300 seconds.

System Supervisor Configuration

Supervisord keeps the Horizon master process alive and restarts it automatically on failure. The following blocks replace the previous `queue:work`-based configuration entirely.

```
; Horizon — manages all VeloxFactory queue workers internally
[program:veloxfactory-horizon]
directory=/var/www/veloxfactory
command=/usr/bin/php artisan horizon
user=www-data
process_name=%(program_name)s
numprocs=1
autostart=true
autorestart=true
startretries=10
stopasgroup=true
killasgroup=true
stopsignal=TERM
startsecs=3
stopwaitsecs=600
redirect_stderr=true
stdout_logfile=/var/log/supervisor/veloxfactory-horizon.log
environment=HOME="/home/www-data",PATH="/usr/local/bin:/usr/bin:/bin"

; Reverb — WebSocket server for real-time events
[program:veloxfactory-reverb]
directory=/var/www/veloxfactory
command=/usr/bin/php artisan reverb:start
user=www-data
autostart=true
autorestart=true
startretries=10
stopasgroup=true
killasgroup=true
stopsignal=INT
startsecs=3
stopwaitsecs=60
redirect_stderr=true
stdout_logfile=/var/log/supervisor/veloxfactory-reverb.log
environment=HOME="/home/www-data",PATH="/usr/local/bin:/usr/bin:/bin"

; Scheduler — runs Laravel's task schedule every minute
```

```
[program:veloxfactory-schedule]
directory=/var/www/veloxfactory
command=/usr/bin/php artisan schedule:work
user=www-data
autostart=true
autorestart=true
startretries=10
stopasgroup=true
killasgroup=true
stopsignal=INT
startsecs=3
stopwaitsecs=60
redirect_stderr=true
stdout_logfile=/var/log/supervisor/veloxfactory-schedule.log
environment=HOME="/home/www-data",PATH="/usr/local/bin:/usr/bin:/bin"
```

> **i `stopsignal=TERM` is required for Horizon.** SIGTERM triggers a graceful shutdown — Horizon finishes any in-flight jobs before stopping its workers. Using SIGKILL or SIGINT instead will interrupt running jobs mid-execution and may leave Report History Records in an incomplete state.

After updating the configuration:

```
supervisorctl reread
supervisorctl update
supervisorctl status
```

The Dashboard

The Horizon dashboard is available at `/horizon`. It is restricted to users with the `global:admin` permission — the same permission required to manage users and system-wide settings.

The dashboard provides a real-time view of the entire queue system: pending and completed jobs, throughput metrics, worker counts per supervisor, and a full log of failed jobs with their stack traces. Failed jobs can be retried directly from the dashboard without any CLI access.

Graceful Shutdown & Deployments

When Horizon is stopped — for deployments, configuration changes, or server maintenance — it finishes any jobs currently in flight before exiting. The Supervisord configuration allows up to 600 seconds for this drain, which comfortably covers the longest expected job timeouts.

```
# Stop Horizon gracefully (in-flight jobs will finish first)
supervisorctl stop veloxfactory-horizon

# Restart after a deployment
supervisorctl restart veloxfactory-horizon
```

Never force-kill the Horizon process during a deployment. Always use `supervisorctl stop` or `supervisorctl restart` — both send SIGTERM and wait for the drain period.