Configuring Redis Queues and Laravel Horizon for Efficient Background Job Processing

This guide will walk you through setting up Redis as your queue driver and configuring Laravel Horizon for monitoring and managing your queue workers.

1. Install Required Packages

First, install the necessary packages via Composer:

```
composer require laravel/horizon predis/predis
```

2. Configure Redis as Queue Driver

Update your `.env` file to use Redis as the queue driver:

```
QUEUE_CONNECTION=redis

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

3. Configure Queue Connections

In `config/queue.php`, ensure your Redis connection is properly configured:

PROF

```
'redis' => [
    'driver' => 'redis',
    'connection' => 'default',
    'queue' => env('REDIS_QUEUE', 'default'),
    'retry_after' => 90,
    'block_for' => null,
    'after_commit' => false,
],
```

4. Publish Horizon Assets

Publish Horizon's configuration and assets:

```
php artisan horizon:install
```

This will create:

- `config/horizon.php` (configuration file)
- `app/Providers/HorizonServiceProvider.php` (service provider)
- Dashboard view files

5. Configure Horizon

Edit `config/horizon.php` to configure your environments and worker settings:

```
'environments' => [
    'production' => [
        'supervisor-1' => [
            'connection' => 'redis',
            'queue' => ['default', 'notifications', 'emails'],
            'balance' => 'auto',
            'processes' => 10,
            'tries' => 3,
            'timeout' => 60,
            'memory' => 128,
        ],
    ],

    'local' => [
        'supervisor-1' => [
            'connection' => 'redis',
            'queue' => ['default'],
            'balance' => 'simple',
            'processes' => 3,
            'tries' => 3,
            'timeout' => 60,
            'memory' => 128,
        ],
    ],
],
```

PROF

6. Start Horizon

Run Horizon in your local environment:

```
php artisan horizon
```

For production, you should configure a process manager like Supervisor to keep Horizon running.

7. Configure Supervisor for Production (Optional but Recommended)

Create a Supervisor configuration file at `/etc/supervisor/conf.d/horizon.conf`:

```
[program:horizon]
process_name=%(program_name)s
command=php /path/to/your/project/artisan horizon
autostart=true
autorestart=true
user=forge
redirect_stderr=true
stdout_logfile=/path/to/your/project/storage/logs/horizon.log
stopwaitsecs=3600
```

Then update Supervisor:

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl start horizon
```

8. Monitoring and Metrics

Horizon provides a dashboard at `/horizon` (protect this route in production). You can also configure metrics:

```
// config/horizon.php
'metrics' => [
    'trim_snapshots' => [
        'job' => 24,
        'queue' => 24,
    ],
],
```

PROF

9. Queue Worker Configuration Tips

Balancing Strategies:

- `simple`: Process jobs in order
- `auto`: Balance based on queue workload
- `false`: Process all queues equally

Memory Limits:

Set appropriate memory limits based on your job requirements:

```
'memory_limit' => 64, // MB
```

Timeouts:

Configure timeouts to prevent stuck jobs:

```
'timeout' => 60, // seconds
```

10. Testing Your Setup

1. Create a test job:

```
php artisan make:job ProcessPodcast
```

2. Dispatch the job:

```
ProcessPodcast::dispatch()->onQueue('default');
```

3. Check Horizon dashboard to see the job processing.

11. Performance Optimization

1. **Prefetching**: In `config/horizon.php`, adjust the `balance_maxshift` and `balance_cooldown` values to optimize worker distribution.

2. **Prioritize Queues**: Use queue priorities in your supervisor configuration:

```
'queue' => ['high', 'default', 'low'],
```

PROF

3. **Monitor Performance**: Use Horizon's metrics to identify bottlenecks and adjust worker counts accordingly.

12. Security Considerations

1. Protect the Horizon dashboard by adding authentication in `app/Providers/HorizonServiceProvider.php`:

```
protected function gate()
{
    Gate::define('viewHorizon', function ($user) {
        return in_array($user->email, [
            'admin@example.com',
        ]);
    });
}
```

2. Use HTTPS for the Horizon dashboard in production.

13. Troubleshooting

Common issues and solutions:

1. **Jobs stuck in pending**:
   - Check if Horizon is running
   - Verify Redis connection
   - Check for queue name mismatches
2. **High memory usage**:
   - Reduce the number of processes
   - Lower the memory limit
   - Optimize your jobs
3. **Failed jobs**:
   - Configure retries appropriately
   - Set up failed job logging

By following this configuration, you'll have a robust queue processing system that can handle background jobs efficiently while providing visibility into your queue operations through Horizon's dashboard.