Automating Tasks with Laravel Scheduler and Custom Commands

Laravel provides a powerful task scheduling system that allows you to automate various operations. Here's how to implement it effectively with cron jobs and custom commands in your Dockerized environment.

1. Setting Up the Scheduler

Create the Scheduler Kernel

All scheduled tasks are defined in `app/Console/Kernel.php`:

```
protected function schedule(Schedule $schedule)
{
    // Run database backups daily at midnight
    $schedule->command('db:backup')
             ->daily()
             ->onSuccess(function () {
                 Log::info('Database backup completed successfully');
             })
             ->onFailure(function () {
                 Log::error('Database backup failed');
             });

    // Process queued jobs every minute
    $schedule->command('queue:work --stop-when-empty')
             ->everyMinute()
             ->withoutOverlapping();

    // Send daily reports at 8 AM
    $schedule->job(new SendDailyReports)
             ->dailyAt('08:00')
             ->timezone('America/New_York');

    // Clear temp files weekly
    $schedule->exec('rm -rf storage/temp/*')
             ->weekly();

    // Custom maintenance tasks
    $schedule->command('maintenance:cleanup')
             ->hourly();
}
```

PROF

2. Creating Custom Commands

Generate a new command:

```
php artisan make:command DatabaseBackupCommand
```

Example Command Implementation:

```
<?php

namespace App\Console\Commands;

use Illuminate\Console\Command;
use Illuminate\Support\Facades\Storage;
use Carbon\Carbon;

class DatabaseBackupCommand extends Command
{
    protected $signature = 'db:backup';
    protected $description = 'Create a database backup';

    public function handle()
    {
        $filename = "backup-" . Carbon::now()->format('Y-m-d-H-i-s') .
".sql";
        $command = "mysqldump --user=" . env('DB_USERNAME') .
                   " --password=" . env('DB_PASSWORD') .
                   " --host=" . env('DB_HOST') .
                   " " . env('DB_DATABASE') .
                   " > " . storage_path() . "/app/backups/" . $filename;

        $returnVar = NULL;
        $output = NULL;
        exec($command, $output, $returnVar);

        if ($returnVar === 0) {
            $this->info("Database backup created: " . $filename);

            // Upload to cloud storage
            Storage::disk('s3')->put(
                'backups/'.$filename,

file_get_contents(storage_path('app/backups/'.$filename))
            );

            return 0;
        } else {
            $this->error("Database backup failed");
            return 1;
        }
    }
}
```

3. Docker Configuration

Update your `docker-compose.yml`:

```
services:
  # ... existing services ...

  scheduler:
    build:
      context: .
      dockerfile: docker/php/Dockerfile
    container_name: laravel_scheduler
    command: >
      bash -c "echo 'Starting scheduler...' &&
      while [ true ]
      do
        php artisan schedule:run --verbose --no-interaction &
        sleep 60
      done"
    volumes:
      - ./src:/var/www
    depends_on:
      - mysql
      - redis
```

Alternative using cron in Docker:

Create a `docker/cron/laravel-cron` file:

```
* * * * * cd /var/www && php artisan schedule:run >> /dev/null 2>&1
```

PROF

Update your `docker-compose.yml`:

```
services:
  # ... existing services ...

  cron:
    image: alpine:latest
    container_name: laravel_cron
    volumes:
      - ./src:/var/www
      - ./docker/cron:/etc/crontabs/root
    command: crond -f -l 8
    depends_on:
      - app
```

4. Advanced Scheduling Techniques

Task Hooks

```
$schedule->command('reports:generate')
    ->daily()
    ->before(function () {
        // Prepare system for report generation
    })
    ->after(function () {
        // Clean up after report generation
    });
```

Maintenance Mode Awareness

```
$schedule->command('import:data')
    ->hourly()
    ->evenInMaintenanceMode();
```

Timezone Configuration

```
$schedule->command('send:daily-notifications')
    ->dailyAt('09:00')
    ->timezone('America/Chicago');
```

Job Scheduling

PROF

```
$schedule->job(new ProcessPodcasts)
    ->everyFiveMinutes()
    ->onQueue('podcasts');
```

5. Monitoring Scheduled Tasks

Logging Task Output

```
$schedule->command('inventory:update')
    ->hourly()
    ->appendOutputTo(storage_path('logs/inventory.log'));
```

Email Notifications

```php
$schedule->command('reports:generate')
    ->daily()
    ->sendOutputTo(storage_path('logs/reports.log'))
    ->emailOutputTo('admin@example.com');
```

Slack Notifications

```php
$schedule->command('backup:run')
    ->daily()
    ->onSuccess(function (Stringable $output) {
        Slack::to('#backups')->send("Backup succeeded!\n" . $output);
    })
    ->onFailure(function (Stringable $output) {
        Slack::to('#alerts')->send("Backup failed!\n" . $output);
    });
```

6. Complex Scheduling Examples

Business Hours Processing

```php
$schedule->command('process:orders')
    ->everyFifteenMinutes()
    ->weekdays()
    ->between('8:00', '17:00');
```

Seasonal Tasks

PROF

```php
$schedule->command('send:holiday-promo')
    ->daily()
    ->when(function () {
        return now()->between(
            Carbon::parse('November 15'),
            Carbon::parse('December 31')
        );
    });
```

Chained Tasks

```php
$schedule->command('import:data')
    ->daily()
    ->then(function () {
        Artisan::call('process:imported-data');
```

```
    })
    ->thenPing('https://example.com/webhook/import-complete');
```

7. Testing Scheduled Tasks

Unit Testing Commands

```
public function test_backup_command()
{
    Storage::fake('s3');

    $this->artisan('db:backup')
         ->assertExitCode(0);

    Storage::disk('s3')->assertExists('backups/backup-*.sql');
}
```

Testing Schedule Definition

```
public function test_schedule_configuration()
{
    $schedule = app()->make(Schedule::class);

    $events = collect($schedule->events())
        ->filter(function ($event) {
            return stripos($event->command, 'db:backup');
        });

    $this->assertCount(1, $events);
    $this->assertEquals('0 0 * * *', $events->first()->expression);
}
```


8. Production Considerations

Supervisor Configuration

For reliable execution, use Supervisor to monitor the scheduler:

```
[program:laravel-scheduler]
process_name=%(program_name)s_%(process_num)02d
command=/bin/bash -c "php /var/www/artisan schedule:run --verbose --no-
interaction"
autostart=true
autorestart=true
user=www-data
```

```
numprocs=1
redirect_stderr=true
stdout_logfile=/var/www/storage/logs/scheduler.log
```

Locking to Prevent Overlaps

```
$schedule->command('long:process')
    ->daily()
    ->withoutOverlapping()
    ->onOneServer();
```

Error Handling

```
$schedule->command('critical:process')
    ->hourly()
    ->before(function () {
        $this->backupCurrentState();
    })
    ->onFailure(function () {
        $this->notifyAdmins();
        $this->revertToBackup();
    });
```

9. Custom Frequency Options

Extend the scheduler with custom frequencies in `AppServiceProvider`:

```
use Illuminate\Console\Scheduling\Event;

Event::macro('everyOddHour', function () {
    return $this->hourlyAt(01, 03, 05, 07, 09, 11, 13, 15, 17, 19, 21,
23);
});
```

Usage:

```
$schedule->command('sync:data')->everyOddHour();
```

10. Complete Docker Setup Example

Here's a complete Docker configuration for scheduling:

```yaml
version: '3.8'

services:
  # ... other services ...

  scheduler:
    build:
      context: .
      dockerfile: docker/php/Dockerfile
    container_name: laravel_scheduler
    command: >
      bash -c "php artisan schedule:work"
    volumes:
      - ./src:/var/www
    environment:
      - TZ=America/New_York
    depends_on:
      - mysql
      - redis
      - app

  queue:
    build:
      context: .
      dockerfile: docker/php/Dockerfile
    container_name: laravel_queue
    command: >
      bash -c "php artisan queue:work --tries=3 --sleep=3 --timeout=120"
    volumes:
      - ./src:/var/www
    depends_on:
      - mysql
      - redis
      - app
```

PROF

This implementation provides a robust task automation system with Laravel's scheduler, complete with Docker integration, monitoring, and advanced scheduling capabilities. The system will reliably execute your scheduled tasks while providing proper logging and error handling.